NIST Special Publication 800-85

PIV Middleware and PIV Card Application Conformance Test Guidelines (SP800-73 compliance)

National Institute of Standards and Technology
Technology Administration
U.S. Department of Commerce

Ramaswamy Chandramouli
Levent Eyuboglu
Ketan Mehta

INFORMATION SECURITY

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD, 20899-8930

October 2005

U.S. Department of Commerce
Carlos M. Gutierrez, Secretary

Technology Administration
Michelle O'Neill, Acting Under Secretary of Commerce for Technology

National Institute of Standards and Technology
William A. Jeffrey, Director

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of non-national security-related information in federal information systems. This special publication 800-series reports on ITL's research, guidelines, and outreach efforts in information system security, and its collaborative activities with industry, government, and academic organizations.

Acknowledgements

The authors (Ramaswamy Chandramouli of NIST, Levent Eyuboglu and Ketan Mehta of Booz Allen Hamilton) wish to thank their colleagues who reviewed drafts of this document and contributed to its development. Special thanks to the General Services Administration and Federal Identity Credentialing Committee for their contribution to the development of this document. The authors also gratefully acknowledge and appreciate the many contributions from the public and private sectors whose thoughtful and constructive comments improved the quality and accuracy of this publication.

Table of Contents

1. INTRODUCTION 1
1.1 AUTHORITY 1
1.2 PURPOSE 2
1.3 SCOPE 2
1.4 TARGET AUDIENCE 3
1.5 DOCUMENT OVERVIEW 3
2. SYSTEM OVERVIEW 5
2.1 TEST PLAN 6
2.2 TEST SET-UP 6
2.3 TEST SYSTEM CONFIGURATION 6
2.3.1 PIV Middleware Test Configuration 7
2.3.2 PIV Card Application Test Configuration 8
3. TEST SUITE ELEMENTS 10
3.1 PIV MIDDLEWARE TESTS 10
3.2 PIV CARD APPLICATION TESTS 10
3.2.1 PIV Card Application Card Command Interface Tests 11
3.2.2 PIV Data Object Representation Tests 12
4. DERIVED TEST REQUIREMENTS 14
5. TEST ASSERTIONS 16
5.1 MAPPING FROM TEST CATEGORIES TO TEST ASSERTIONS 16
5.2 PIV CLIENT API TEST ASSERTIONS 17
5.3 PIV CARD COMMAND INTERFACE TEST ASSERTIONS 18
5.4 PIV DATA OBJECT REPRESENTATION TEST ASSERTIONS 18
5.5 PIV AUTHENTICATION USE CASE TEST ASSERTIONS 19
6. TEST AND COMPLIANCE DOCUMENTATION 20
7. ACCEPTANCE CRITERIA 21
7.1 ACCEPTANCE CRITERIA FOR THE PIV MIDDLEWARE TEST 21
7.2 ACCEPTANCE CRITERIA FOR THE PIV CARD APPLICATION TESTS 21
8. TEST AND COMPLIANCE PROCESS 22
8.1 FAILURE REVIEW 23
APPENDIX A— DERIVED TEST REQUIREMENTS A-1
APPENDIX B— PIV CLIENT API TEST ASSERTIONS B-1
APPENDIX C— CARD COMMAND INTERFACE TEST ASSERTIONS C-1
APPENDIX D— PIV DATA OBJECTS REPRESENTATION TEST ASSERTIONS D-1
APPENDIX E— PIV AUTHENTICATION USE CASE TEST ASSERTIONS E-1
APPENDIX F— TEST REPORTS F-1

APPENDIX G— ACRONYMS G-1

List of Figures

Figure 1: PIV Conformance Test Architecture 5
Figure 2: Test System Configuration 7
Figure 3: Middleware Test Configuration 7
Figure 4: PIV Card Application Test Configuration 9

List of Tables

Table 5-1. Cross-referencing Guide 16
Table 5-2. List of Commands in Client API and Number of Test Cases 17
Table 5-3. List of Commands in Card Application Command and Number of Test Cases 18
Table A-1 PIV Command Mapping A-9
Table D-1. Encoding of Length Field D-1

1. Introduction

The Personal Identity Verification (PIV) of Federal Employees and Contractors, Federal Information Processing Standard 201 (FIPS 201) was developed to establish standards for identity credentials and contains two parts. Part 1 (PIV-I) describes the minimum requirements for a federal personal identification system that meets the control and security objectives of Homeland Security Presidential Directive (HSPD) 12. Part 2 (PIV-II) gives the technical specifications of components and processes required for the interoperability of PIV Cards[1] with the access control and PIV card management systems throughout the Federal Government. FIPS 201 is accompanied by three documents:

+ National Institute of Standards and Technology Special Publication 800-73 (NIST SP 800-73), specifies interface requirements for retrieving and using the identity credentials from the PIV Card. It also defines a PIV data model, which details the structure and format of the information stored on the PIV Card.
+ NIST SP 800-76 Draft contains technical specifications for biometric data mandated in FIPS 201.
+ NIST SP 800-78 specifies the cryptographic algorithms and key sizes for performing cryptographic operations on PIV data objects defined as part of the PIV data model.

This test guidance document specifies the test plan, processes, derived test requirements, and the detailed test assertions/conformance tests for testing the following PIV software components:

+ PIV middleware (implements PIV Client API)
+ PIV card application.

1.1 Authority

This document has been developed by the National Institute of Standards and Technology (NIST) in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets, but such standards and guidelines shall not apply to national security systems. This recommendation is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), “Securing Agency Information Systems,” as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided A-130, Appendix III.

This recommendation has been prepared for use by federal agencies. It may be used by non-governmental organizations on a voluntary basis and is not subject to copyright. Nothing in this document should be taken to contradict standards and guidelines made mandatory and binding on federal agencies by the Secretary of Commerce under statutory authority. Nor should this

[1] The term PIV Card in the context of this document refers to a smart card loaded with a PIV card application.

recommendation be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of OMB, or any other federal official.

1.2 Purpose

The objective of this document is to provide test requirements and test assertions that could be used to validate the compliance/conformance of two PIV components—*PIV middleware* and *PIV card application* with the specifications in NIST SP 800-73. Because NIST SP 800-73 specifications were developed for meeting interoperability goals of FIPS 201, the conformance tests in this document provide the assurance that the set of PIV middleware and PIV card applications that have passed these tests are interoperable. This in turn facilitates marketing and procurement of FIPS 201-conformant products that meet the goals of HSPD-12.

1.3 Scope

This document provides guidelines for running conformance tests for the following four classes of specifications in NIST SP 800-73:

- End-Point Client-Application Programming Interface (Chapter 6 of SP 800-73).
- End-Point PIV Card Application Card Command Interface (Chapter 7 of SP800-73).
- End-Point Data Objects (Chapter 4 of SP 800-73) and End-Point Data Types and their Representations (Chapter 5 of SP 800-73).
- PIV Authentication Use Cases (C.1.2 and C.1.4 of Appendix C of SP 800-73)

The functions specified in the Client-Application Programming Interface are to be supported by PIV middleware. The commands specified in the PIV Card Application Card Command Interface are to be supported by PIV card application. The data objects associated with the PIV card application have to be tested for correct representation. The PIV authentication use cases are performed using a group of commands provided by the PIV card application using the contents of the data objects. Hence, the four classes of conformance tests listed above span the following two main PIV components: PIV middleware and PIV card application. The test suite provided in this document is divided into the following two broad categories of tests:

- PIV middleware tests
- PIV card application tests.

The above tests are developed through the following two-step process:

- **Derived test requirements (DTR).** These are constructed from the ‘Shall’ statements in SP 800-73 specifications. Because the PIV authentication use cases involve a group of commands whose behavior is covered under DTRs, there are no separate DTRs for PIV authentication use cases.
- **Test assertions.** These provide the tests that need to be performed to test each of the requirements under DTRs as well as tests with appropriate execution conditions for each

of the commands in the interface to realize each of the return/response status codes specified in SP 800-73.

This document does not provide conformance tests for any other software used in the PIV system such as the back-end access control software, card issuance software, card reader/biometric reader drivers, and specialized service provider software such as Cryptographic Service Provider modules and Biometric Service Provider Modules. This document does not address or provide conformance tests for SP800-76 or FIPS 201.

1.4 Target Audience

This document is intended to:

+ Enable developers of PIV middleware and the PIV card application to develop their software modules to be testable for interface requirements specified in SP 800-73.
+ Enable developers of PIV middleware and the PIV card application to develop self-tests as part of the development effort.
+ Enable testing laboratories authorized to perform conformance tests on PIV middleware and the PIV card application to develop tests that cover the test suite provided in this document.

1.5 Document Overview

The document is organized as follows:

+ Section 2 provides a conceptual software overview of a typical PIV system and introduces the PIV test components.
+ Section 3 lists the various elements of the test suite under the two broad categories of tests (PIV middleware tests and PIV card application tests) provided in this document.
+ Section 4 provides an overview of the DTR construction process.
+ Section 5 gives a brief description of the test assertion for each of the four specification classes covered by this document (refer to Section 1.3).
+ Section 6 explains the documentation required from both the component owners and test labs for conducting the testing process.
+ Section 7 details the acceptance criteria for each type of test.
+ Section 8 explains the test compliance process and failure review.
+ Appendix A includes derived test requirements based on specifications in SP 800-73.
+ Appendix B includes client Application Programming Interface (API) test assertions.

- Appendix C includes PIV card command interface test assertions.
- Appendix D includes PIV data objects representation test assertions.
- Appendix E includes PIV authentication use cases test assertions.
- Appendix F includes sample test reports.
- Appendix G, Acronyms, describes the textual representations used in the document.

2. System Overview

The conceptual architecture involving the two PIV software components for which conformance tests are given in this document is shown in Figure 1. The conformance test in this document applies to the areas highlighted with dashed lines in the Figure 1.

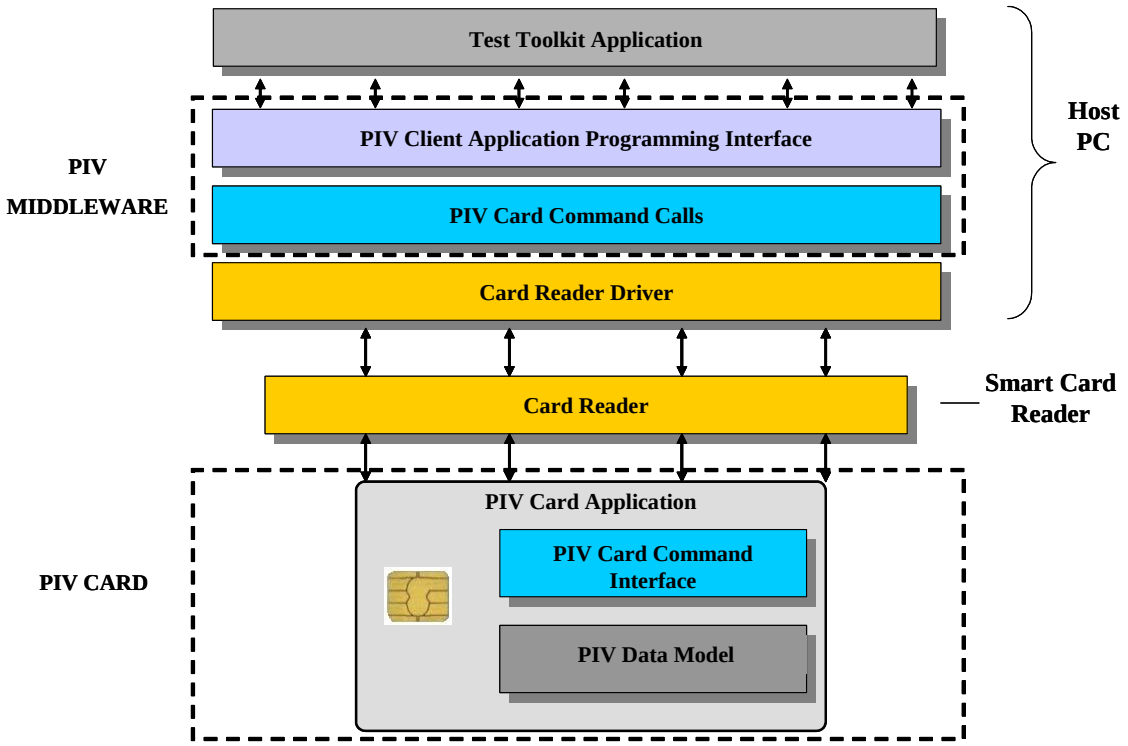

Figure 1: PIV Conformance Test Architecture

+ PIV middleware is a software application that is the interface between an agency's PIV implementation and the PIV card application. It allows the agency applications to remain independent of the underlying operating system platform. The PIV middleware has the following two functions.

    1. It implements the commands in the PIV Client Application Programming Interface (section 6 of SP 800-73).

    2. It generates the appropriate commands (also called Application Protocol Data Units or APDUs) in PIV Card Command Interface (card edge interface – section 7 of SP 800-73) and thus communicates with the PIV card application.

+ The PIV card application resides on the card, implements the commands in the PIV Card Command Interface (section 7 of SP 800-73) and provides access to objects of the PIV Data Model. The PIV Data model defines the logical use of the on-card application space

including the SP800-73 required data objects, and data elements, along with the size and structure of each object.

2.1 Test Plan

The test plan identifies the tasks/artifacts required for testing the PIV middleware and PIV card application. These artifacts include the following: PIV middleware and smart card populated with PIV card application; the test toolkit (or test scripts), which implements the test assertions; and the various infrastructure devices needed to interface with the card, such as the card acceptance device (called the card reader). The components involved in the test plan and the elements of the test configuration for the two broad categories of tests presented in this document are discussed in the next two subsections.

2.2 Test Set-up

The test system consists of the following components:[2]

+ A test toolkit application software that resides on a personal computer (PC)
+ Smart card readers
  - An ISO 7816 and PC/SC-compliant contact-based smart card reader and
  - An ISO 14443 and PC/SC-compliant contactless smart card reader

  or
  - A dual interface reader
+ A mechanism to input personal identification number (PIN) (e.g. a PIN pad or a keyboard) that can be transmitted to the SC Reader.
+ A set of test PIV Cards, loaded with PIV card application, with a contact interface that is compliant with ISO/IEC 7816 and a contactless interface that is compliant with ISO/IEC 14443 or a test PIV card emulator
+ PIV Middleware application

These components will be used in different configurations based on the type of test being conducted in the test bed.

2.3 Test System Configuration

The test system shown in Figure 2, will be configured in both a middleware test and a card application test to accommodate the different components to be tested, as explained in Section 3.

[2] Compliance of the readers and input devices with an external standard such as ISO/IEC 7816 is not addressed in this document.

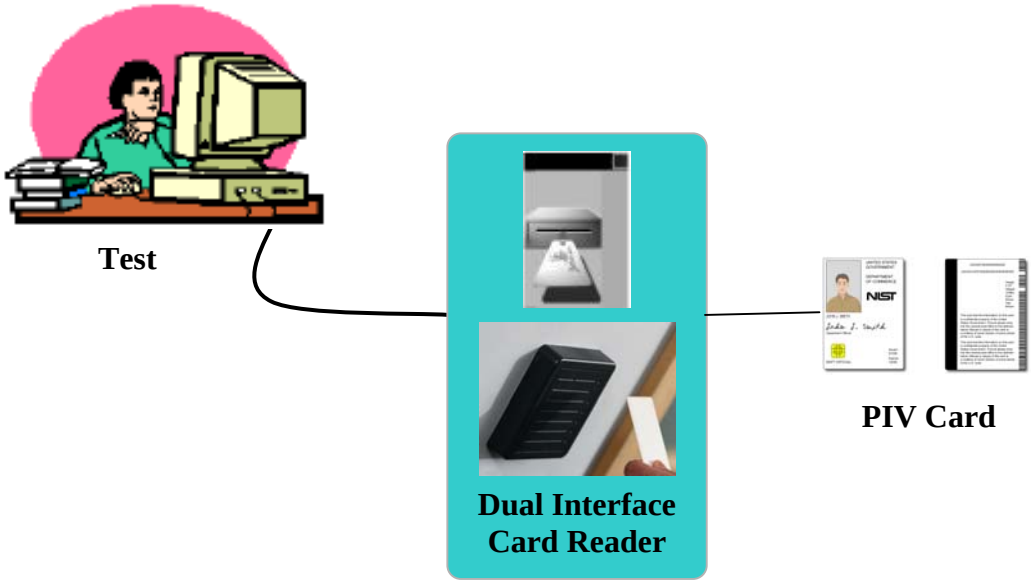

Figure 2: Test System Configuration

2.3.1 PIV Middleware Test Configuration

The middleware test configuration is used to test a vendor's middleware software application that implements the PIV client API and generates the appropriate commands in PIV card command interface (refer Table A-1 for mapping between client API and card command interface). The middleware test configuration is depicted in Figure 3.

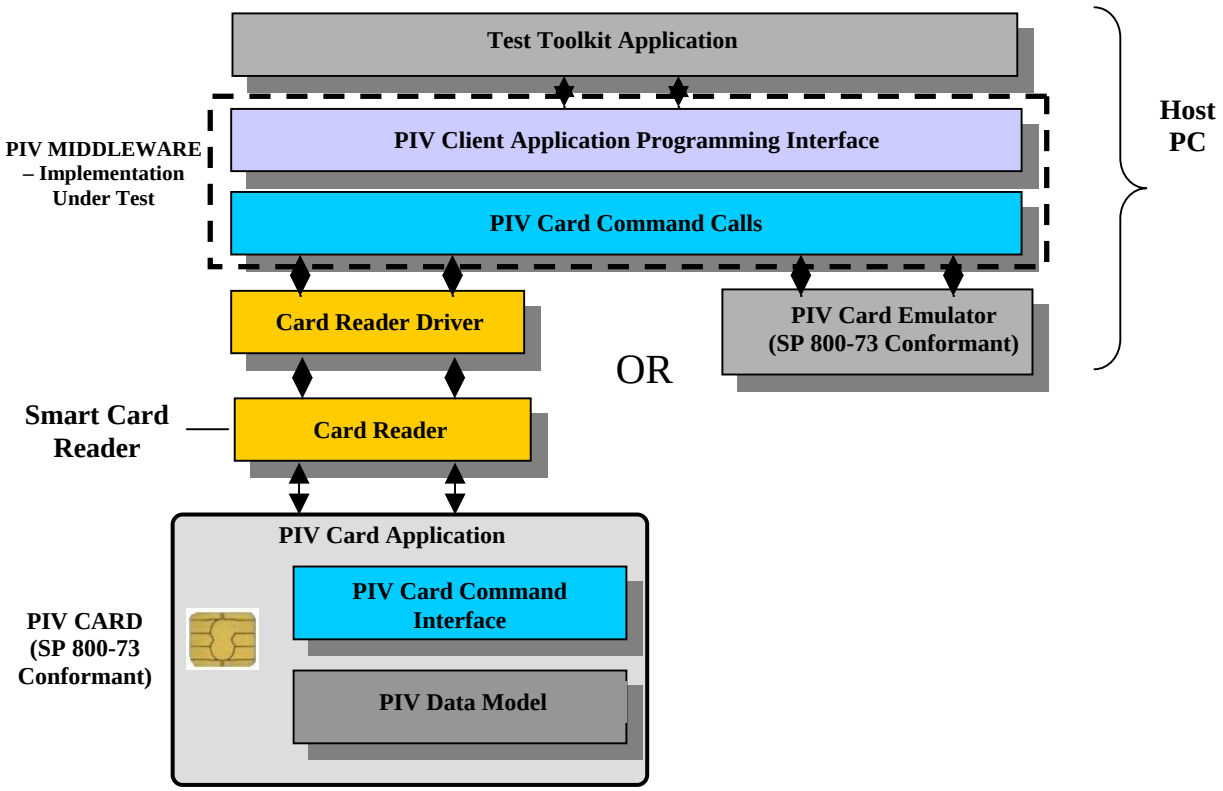

Figure 3: Middleware Test Configuration

The following list shows the test system components included in this configuration:

+ A test toolkit application software
+ Vendor provided PIV middleware which is the subject of this test (also called as Implementation Under Test or IUT)

One of the following combinations

+ Contact and contactless smart card readers or a dual interface reader together with
+ A PIN input mechanism together with
+ A dual interface FIPS 201 conformant PIV Card loaded with "SP 800-73 conformant PIV card application" (for definition refer section 7.2)

OR

+ A PIV card emulator that emulates the behavior of PIV card application.
+ A printer for reporting and documenting the test results

The test toolkit application software resides on the Test Computer, and facilitates the execution and management of both test suites explained in Section 3. For the PIV Middleware Test, the test system (Figure 2) will be configured so that the vendor provided PIV middleware under test is also installed on the Test Computer and interacts with the FIPS 201 conformant test cards via the card reader(s).

2.3.2 PIV Card Application Test Configuration

The card application test configuration is used to test any PIV card application through commands of the PIV card command interface defined in SP 800-73. The following list shows the test system components included in this configuration:

+ A test toolkit application software
+ Contact and a contactless smart card readers or a dual interface reader
+ A PIN input mechanism
+ A PIV Card loaded with PIV card application that supports contact and contactless interface and is the subject of this test. (also called Implementation Under Test or IUT)

For the PIV Card Application Test, the test system shown in Figure 2 will be configured such that the test toolkit application software directly interacts with the PIV Card under test via the card reader(s). The PIV Card Application Test configuration is depicted in Figure 4.

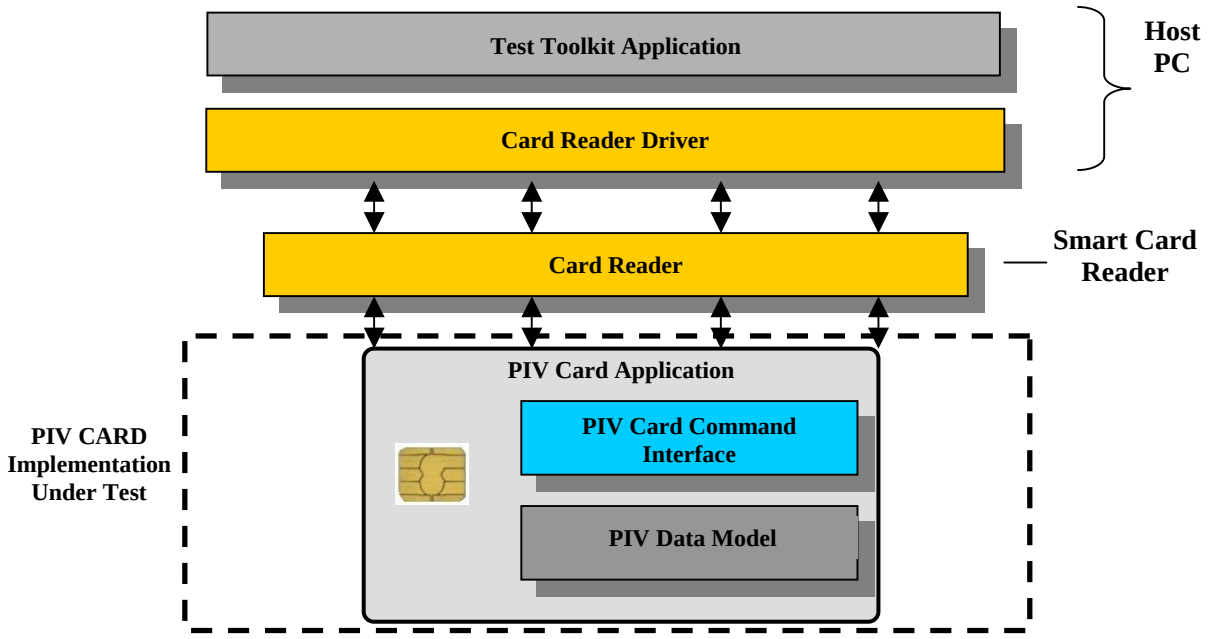

Figure 4: PIV Card Application Test Configuration

3. Test Suite Elements

Based on the conceptual software architecture shown in Figure 1, the PIV software components that are involved in and are subject to testing are as follows:

+ PIV middleware that implements the commands in the PIV Client API and interfaces with PIV card application (resident on the card) by generating commands (APDUs) in PIV card command interface.
+ PIV card application that implements the PIV application card command interface, accesses and modifies the content of PIV data objects, and facilitates realization of PIV authentication use cases.

3.1 PIV Middleware Tests

These tests will validate that the PIV middleware conforms to the specification in section 6 of SP 800-73. Conformance criteria includes correct implementation of all commands in PIV client API, generation of appropriate commands in the PIV card command interface to communicate with the PIV card application and return the prescribed response codes to the calling agency application. This test, however, does not validate the functional requirements or the testing of the FIPS 201-mandated card application parameters, which are covered under the PIV Card test.

The following client API functions are tested for conformance:

+ pivConnect
+ pivDisconnect
+ pivSelectCardApplication
+ pivLogIntoCardApplication
+ pivGetData
+ pivLogoutOfCardApplication
+ pivCrypt
+ pivPutData
+ pivGenerateKeyPair.

These functions will be tested for their response to both the valid and the error conditions as defined by this document. To conduct these tests, a smart card with an “SP 800-73- conformant PIV card application” (refer section 7.2 for definition) must be accessible.

3.2 PIV Card Application Tests

PIV card application tests cover the following:

- The PIV card application card command interface as per chapter 7 of SP800-73
- PIV data objects representation as specified in Chapters 4 and 5 of SP 800-73
- PIV authentication use cases as specified in C.1.2 and C.1.4 of Appendix C of SP 800-73.

The tests are performed through test scripts communicating directly with a PIV Card through the API of the driver that comes with the card reader .

3.2.1 PIV Card Application Card Command Interface Tests

As defined in SP 800-73, Chapter 7, the PIV card application shall expose the end-point PIV card application command interface to be accessible through PIV middleware, which as previously mentioned is the PIV client application. These tests will validate that the card under test can successfully execute the commands in the PIV card command interface. Successful execution constitutes the card responding with appropriate data and response status codes to the commands sent by the test system. It also involves setting of certain state variables within the PIV Card. For example, the criteria for successful execution of the SELECT command involves the following:

- The Response Status Code returned is ”90 00”.
- The application property template is returned with the right format and content.
- The PIV card application is the currently selected application on the card.

The card command interface test suite includes conformance tests for the following PIV card application commands:

- Data access commands
  - SELECT
  - GET DATA
- Card authentication commands
  - VERIFY
  - CHANGE REFERENCE DATA
  - RESET RETRY COUNTER
  - GENERAL AUTHENTICATE
- Credential initialization and administration commands
  - PUT DATA
  - GENERATE ASYMMETRIC KEY PAIR

The card edge commands will be validated against the following conditions:

- Card interface type (contact vs. contactless)

- Precondition for use (PIN verified, Currently Selected Application value)
- Expected Response status codes
- Right Content and Encoding for returned data
- Appropriate State Variables set in the card.

3.2.2 PIV Data Object Representation Tests

These tests validate the data objects resident on the PIV Card for compliance with the PIV data model whose elements and types are specified in Chapters 4 and 5 of SP 800-73. Specifically the tests check whether the data objects implemented on the PIV Card have used the right Basic Encoding Rules Tag-Length-Value (BER-TLV) tags and associated length and in the same order as defined in the PIV data model. The list of criteria used is given in subsection 3.2.2.1.

The testing covers the following data objects

- The six mandatory data objects as defined in Appendix A of SP 800-73
  - Card capabilities container
  - Cardholder unique identifier (CHUID)
  - X.509 certificate for PIV authentication
  - Cardholder fingerprint I
  - Cardholder fingerprint II
  - Security object
- The five optional data objects, also defined in the Appendix A of SP 800-73
  - On-card printed information
  - Cardholder facial image
  - X.509 certificate for PIV digital signature
  - X.509 certificate for card authentication
  - X.509 certificate for PIV key management.

3.2.2.1 Validation of Mandatory and Optional Data Objects

Each on-card data container will be validated against the following requirements as per Appendix A of SP 800-73:

- Correct container ID
- Data container size restrictions
- Data container access rule (PIN vs. no PIN)
- Data container access type (contact vs. contactless)
- Data element tag codes

+ Data element type and size requirements.

The validation process consists of selecting the PIV card application on the card, reading the contents of each data container through the GET DATA command (after satisfying the required security condition), parsing the string of bits returned based on expected TLV tag codes and length values.

3.2.3 PIV Authentication Use Cases Tests

The PIV authentication use cases specified in Appendix C of SP 800-73 specifies the sequence of PIV Card user activities and PIV card commands needed to implement each of the authentication mechanisms in Section 6 of FIPS 201. The following use cases in Appendix C of SP 800-73 are not covered under the tests given in this document.

+ **Authentication using PIV visual credentials (C.1.1).** Because the interaction is not electronic
+ **Authentication using PIV biometrics (C.1.3).** Because the specification of the biometric data object for PIV Card has not been finalized.

The validated use cases and an overview of the validation process are given below:

+ **Authentication using PIV CHUID (C.1.2).** Parse CHUID data and validate its integrity through its signature verification. Also, check whether the PIV Card is currently valid by checking the Expiration Date.
+ **Authentication using PIV authentication key (C.1.4).** Retrieve PIV Authentication Certificate, Validate Certificate, Send a Challenge to Card, and Validate the Response to Challenge.

4. Derived Test Requirements

DTRs show the type of tests required based on the specifications in SP 800-73. These specifications cover expected command behavior (in the case of interface specification), data object representation (in the case of PIV data model) and data contents (in the case of PIV authentication use cases).

Each DTR consists of the following:

+ Actual condition statements taken/derived from the SP 800-73 specification – these include conditions for successful command execution for each command as well as exception behaviors explicitly called out through 'shall' statements in SP 800-73. Those exception behaviors that are implicit in SP 800-73 through listing of error codes associated with each command are tested only through Test Assertions (Appendices B, C and D) and are not part of the DTR condition statements. The condition statements are identified by codes starting with 'AS' followed by a running sequence that denotes the section in this document where they occur.

+ Required Vendor Information – these include information that the vendors are mandated to provide in their documentation. The Required Vendor Information is identified by codes starting with 'VE' followed by a running sequence that denotes the section in this document where they occur.Required Test Procedures – these are actions that the tester has to perform in order to satisfy the requirements stated in actual condition statements. These include verifying the information mandated in the "Required Vendor Information" for the condition as well as performing software-based tests. It must be mentioned however that some of the required test procedures will not call out explicitly for verification of information in the associated "Required Vendor Information". In these instances it is implicitly assumed that such information is provided by the vendor and verified by the tester. The Required Test Procedures are identified by codes starting with TE followed by a running sequence that denotes the section in this document where they occur.

Validations of some DTRs are not covered by the test assertions provided in this document. These DTRs require compliance of the component with an external specification or standard such as ISO/IEC 7816 or ISO/IEC 14443. No required test procedures are provided for these DTRs, and a note is added to indicate that "this assertion is externally tested." The tester checks the vendor documentation for claimed compliance with such requirement or the presence of an external test/compliance certificate obtained from the related standards testing body, when applicable.

Some DTRs cannot be validated through the test tools provided in this document. For example, the test tool cannot access the asymmetric private keys generated and stored on the card. Therefore, a note is added to indicate that "this assertion is not separately tested" for these DTRs. The same note is added for DTRs that make general statements on the nature of the PIV Card and are validated as a result of the validation of many other DTRs. For example, the statement "each command that appears on the card command interface shall be implemented by a card

application that is resident in the Integrated Circuit Card (ICC)" is validated through the entire card command interface test and does not require an individual test assertion.

5. Test Assertions

Test assertions are statements of behavior, action, or condition that can be measured or tested. They provide the procedures to guide the tester in executing and managing the test. They include purpose of the test, starting conditions and prerequisites, success criteria, and post-test conditions, when applicable. A list of test assertions can be seen in Appendices B to E.

The following four sets of test assertions are included in this document:

- PIV client API test assertions (Section 3.1 for overview)
- PIV card command interface test assertions (per Section 3.2.1)
- PIV data object representation test assertions (per Section 3.2.2)
- PIV authentication use case test assertions (per Section 3.2.3)

An overview of each of the above classes of test assertions is given in Sections 5.2 through 5.5.

5.1 Mapping from Test Categories to Test Assertions

All the DTRs in Appendix A conceptually come under one of the two broad categories of tests stated in Section 3, i.e., PIV middleware tests and PIV card application tests. Similarly, each test assertion makes specific references to the related sections in SP 800-73 or the related DTRs. However, overall there is many-to-many mapping from the test suite elements (individual tests) under each of these two broad categories of tests to the DTRs (i.e., one test can map to many DTRs and one DTR can map to many tests). A similar type of relationship exists between DTRs and test assertions. To narrow the search space for cross references, Table 5-1 presents a cross-referencing guide showing the relevant DTR sections and test assertion sections with respect to test classes in the two broad categories of tests.

Category/Classes of Test	DTR Section(s)	Test Assertion Section(s)
(1) PIV Middleware Tests (Section 3.1)	A.4 End-Point Client API	Appendix B—PIV Client API Test Assertion
(2a) PIV Card Application Tests—PIV Card Application Card Command Interface Tests (Section 3.2.1)	(1) A.1 End-Point Concepts and Constructs (2) A.5 End-Point PIV Card Application Card Command Interface (3) A.6—PIV Data Model	Appendix C—PIV Card Command Interface Test Assertions
(2b) PIV Card Application Tests—PIV Data Object Representation Tests (Section 3.2.2)	(1) A.2 End-Point Data Objects (2) A.3—End-Point Data Types and Their Representation (3) A.6—PIV Data Model	Appendix D—PIV Data Object Representation Test Assertions
(2c) PIV Card Application Tests—PIV Authentication Use Cases Tests (Section 3.2.3)	NONE—covered by group of DTRs under A.5	Appendix E—PIV Authentication Use Case Test Assertions

Table 5-1. Cross-referencing Guide

5.2 PIV Client API Test Assertions

This section provides conformance tests in the form of test assertions for the functions specified in Chapter 6 of SP 800-73 (called client API) that the PIV middleware is expected to support. The test assertions are described through a test assertions template. The template provides placeholders for describing the purpose of the test, the preconditions required to exercise the test, the parameter values used in test invocation, and the expected results as well the state of the PIV system (value of state variables), if any, that will be affected by the test run (post-condition).

The conformance tests are run against the PIV middleware that in turn interacts with the PIV card application resident on the PIV Card. Hence, there are two pieces of software (PIV middleware and PIV Card application) that determine the outcome of each test run. Because the focus of the tests is the behavior of the PIV middleware, the test configuration assumes the presence of a validated PIV card application.

The list of functions and the number of test cases in Appendix B are summarized in Table 5-2.

Client API Command	Number of Test Cases (Appendix B)
pivConnect	6
pivDisconnect	5
picSelectCardApplication	4
pivLogIntoCardApplication	5
pivLogOutOfCardApplication	3
pivGetData[3]	21
pivPutData	26
pivGenerateKeyPair	5
pivCrypt	6

Table 5-2. List of Commands in Client API and Number of Test Cases

The PIV client API test cases are based on the following assumptions:

- There is a PIV Card with a validated PIV card application.
- A valid connection description is provided for the card application.
- A valid physical connection exists between an instance of the PIV card reader and the host where the PIV middleware (client application) resides.
- No other application is currently connected to the PIV card application.

[3] The number of test cases is based on the assumption that mandatory and optional PIV data objects are implemented. This applies to pivGetData and pivPutData.

5.3 PIV Card Command Interface Test Assertions

This section provides conformance tests in the form of test assertions for the command set that is specified in Chapter 7 of SP 800-73 (Card Application Card Command Interface) that the PIV card application is required to support. The test assertions are described through a test assertions template. The template provides placeholders for describing the purpose of the test, the preconditions required to exercise the test, the parameter values used in test invocation, and the expected results as well as the state of the PIV system (value of state variables), if any, that will be affected by the test run (post- condition).

The conformance tests are run to validate the PIV card application. Interaction with the PIV card application takes place through the API of the driver that comes with the card reader.

The list of commands in the card application card command interface and the number of test cases in Appendix C are summarized in Table 5-3.

Card Application Command	Number of Test Cases (Appendix C)
SELECT	5
GET DATA[4]	31
VERIFY	5
CHANGE REFERENCE DATA	5
RESET RETRY COUNTER	5
GENERAL AUTHENTICATE	6
PUT DATA	34
GENERATE ASYMMETRIC KEY PAIR	7

Table 5-3. List of Commands in Card Application Command and Number of Test Cases

The following assumptions have been made with regard to PIV Card command interface test cases:

- A valid PIV Card is inserted into the contact reader or placed near a contactless reader.
- A valid PC/SC connection exists between the test system and an instance of the reader.
- No application is currently connected to the PIV card application.
- No other contactless card is within the proximity of the contactless reader.

5.4 PIV Data Object Representation Test Assertions

The following assumptions have been made with respect to PIV data object representation test assertions:

- A PIV card application with a valid Application Identifier (AID) is resident on the card.

[4] The number of test cases is based on the assumption that mandatory and optional PIV data objects are implemented. This applies to GET DATA and PUT DATA commands.

- The PIV card application is expected to have implemented all the six mandatory PIV data objects of the PIV data model on the card.
- The presence of any one or more of the five optional PIV data objects on the PIV Card can be known from the vendor documentation.

These test assertions are described in Appendix D.

5.5 PIV Authentication Use Case Test Assertions

The PIV authentication use case assertions are used to validate the PIV Card authentication mechanisms. Because the test system does not include a real access control subsystem, the verification of data received from the card cannot be accomplished against a back-end database. Instead the data read from the PIV Card is compared against a configuration file set up in the test system. These test assertions are described in Appendix E.

6. Test and Compliance Documentation

There are two sets of compliance documentation: vendor required and test facility generated.

The vendor-required documents consist of the following:

+ **For PIV middleware tests, component installation, and execution instructions.** The vendor provides technical instructions and other documentation to aid the testing personnel in installing and using the PIV middleware under test. The PIV middleware must include a user interface that allows testers to execute individual API functions. The PIV middleware must be able to generate and execute appropriate commands in the PIV card command interface in response to a specific API function call executed through the client API interface.

+ **For component technical documentation.** The vendor-supplied component technical documentation must include the detailed technical description and the design of the component to be tested. This document includes, at a minimum, all the required vendor information from the DTRs necessary for testers to execute the tests.

+ **For card testing.** Security information such as the PIV card application PIN, the supported cryptographic algorithm IDs, and their associated key references are required from the vendor.

The test facility-generated documents are required for performing and reporting the test process. The following are some of the examples:

+ **Checklists.** Checklists provide the tester with a list of actions and requirements to complete before the test starts. Information required in the preconditions section of the assertions is included in the checklists

+ **Test logs.** A test log is kept for each test run on any component and is used to summarize the results of all the tests run.

+ **Test reports.** These provide the background (environmental information) for each of the test cases as well as summary of outcomes from test runs (from test logs) associated with each test case.

A test case is a sequence of command/function invocations that pertain to a given execution condition for the 'command/function under test'. For example, if the GET DATA command is the command/function under test, then the execution condition 'Invocation of this function after PIN verification' will consist of the following sequence of command/function invocations – SELECT,VERIFY, GET DATA, and collectively constitutes a test case. There may be many test runs for this test case. The function invocations returning the expected return codes for a test case in all test runs indicates that the command/function has been implemented correctly.

7. Acceptance Criteria

Acceptance criteria are based on the compliance of the item under test with the requirements defined in FIPS 201 and the accompanying special publication documents. The criteria are further specified in the following sections, based on the type of test being conducted.

7.1 Acceptance Criteria for the PIV Middleware Test

The PIV middleware test acceptance criteria will be based on the middleware application under test passing all the PIV client API test assertions. The middleware should return appropriate return codes in response to executing the client API functions as defined in Section 6 of SP 800-73. The middleware should also be able to send the correct card commands to and interpret the responses received from the "SP 800-73 conformant PIV card application" (refer to section 7.2 for definition). The test assertions detail the pass/fail criteria defined for each test case that is designed to test a certain condition being tested.

7.2 Acceptance Criteria for the PIV Card Application Tests

Acceptance criteria for the PIV card application tests are based on the PIV card application passing the following three classes of tests: PIV card application card command interface tests, PIV data objects representation tests, and PIV authentication use cases tests. The PIV Card application that has passed these classes of tests is called "SP 800-73 conformant PIV Card application."

For PIV card application card command tests, the PIV card application should send the appropriate response status codes and application data in response to commands. It should also set or reset certain card state variables and thus fulfill the test post conditions.

For the PIV data object representation tests, the PIV card application should show the presence of all mandatory PIV data objects and published optional PIV data objects implemented with correct BER-TLV tag codes and size requirements and in the same order in which it is outlined in Appendix A of SP 800-73.

For the PIV authentication use case tests, the data returned from the PIV Card at the various stages of the sequence of interactions should exhibit certain properties (expiration date later than the current date, signature and the data objects match, etc).

The acceptance criteria for the testing of PIV Card functionalities, for which FIPS 201 makes reference to external documents (such as digital signature formats), is based on visual verification of vendor- provided documents and test/compliance certificates.

8. Test and Compliance Process

The PIV software component that passes all applicable tests, as explained in this document, will be considered conformant. This document provides the technical details for the testing of the two PIV software components. In this context, compliance means—

- Passing the related test assertions explained in this document
- Passing the inspection/verification of the required vendor documentation.

The certified and/or accredited test laboratory that will conduct the testing has the following responsibilities—

- Prepare and provide the test application forms and the documentation
- Receive and configure the PIV software component to be tested
- Conduct the test with a testing toolkit
- Review the test results and report failures
- Inspects the vendor documentation
- Communicate the results.

Upon vendor's submission of the request for PIV component certification, the required documentation, and the PIV software components to be tested, the test laboratory configures the test system, records all preconditions, and runs the applicable suite of tests for the submitted PIV component. After conducting the tests, the test laboratory evaluates the test results and communicates the Test Results Summary (TSR) and Test Run Details (TRD) to the vendor.

The Test Results Summary provides the overall environmental information (date and time the tests were conducted, the tester name and vendor product identifier etc) as well as the summary conclusion for tests associated with that particular class. The format of the summary report will vary depending upon the test classes. The sample format for each of the above types of summary reports is given in Appendix F. The TSR associated with each of the four classes are:

- PIV Client API Test Summary
- Card Command Interface Test Summary
- PIV Data Objects Representation Test Summary
- PIV Authentication Use Case Test Summary

Test Run Details (TRD) are used to log the details of each test run associated each of the four classes in the test suite. They provide the details of the outcome of each test run for various execution conditions. This detail report will enable the product vendor to make the necessary logic changes to the implementation of the various commands/interfaces and data object representations in order to become fully conformant.

8.1 Failure Review

The test will be repeated once for components that do not pass the tests. After the retest, the tester prepares for each failure a discrepancy report that summarizes the purpose of the test, the progression of steps, and the responses received from the tested components. The discrepancy report will be internally reviewed and discussed by the test lab before an official response is sent to the vendor. Vendors who object to the results presented in the discrepancy report must explain their reason for the objection. If the reason necessitates another retest, the test laboratory may consider repeating the test. Otherwise, the test lab will seek the guidance of the NIST personnel on the failure, before the component is returned to the vendor to be corrected.

Appendix A—Derived Test Requirements

A.1 End-Point Concepts and Constructs

A.1.1 Platform Requirements

AS01.01: The PIV Card Application shall place the following requirements on the ICC platform on which it is implemented or installed:

- **Global security status that includes the security status of a global cardholder PIN**
- **Application selection using a truncated AID**
- **Ability to reset the security status of an individual application**
- **Indication to applications as to which physical communication interface – contact versus contactless- is in use**
- **Support for the default selection of an application upon warm or cold reset.**

Note: This assertion is not separately tested.

A.1.2 Card Applications

AS01.02: Each command that appears on the card command interface shall be implemented by a card application that is resident in the ICC.

Note: This assertion is not separately tested – collection of DTRs for all commands implicitly tests this assertion.

AS01.03: Each card application shall have a globally unique name called its AID [ISO/IEC 7816, Part 4].

Note: This assertion is tested as part of the AS05.05 through AS05.10.

AS01.04: Access to the card commands and data objects of a card application shall be gained by selecting the card application using its application identifier.

Note: This assertion is tested as part of AS05.11.

AS01.05: The Proprietary Identifier eXtension (PIX) of the AID shall contain an encoding of the version of the card application.

Note: This assertion is tested as part of the AS05.05 through AS05.10.

A.1.2.1 Personal Identity Verification Card Application

AS01.06: The AID of the Personal Identity Verification card application (PIV Card Application) shall be: 'A0 00 00 03 08 00 00 10 00 01 00'

Note: This assertion is tested as part of the AS05.05 through AS05.10.

AS01.07: For the first version of the PIV Card Application, the AID shall consist of the NIST Registered application provider IDentifier (RID) 'A0 00 00 03 08' followed by the application portion of the NIST PIX indicating the PIV Card Application '00 00 10 00' and then the version portion of the NIST PIX '01 00' for the first version of the PIV Card Application.

Note: This assertion is tested as part of the AS05.05 through AS05.10.

A.1.2.2 Default Selected Card Application

AS01.08: The card platform shall support a default selected card application. In other words, there shall be a currently selected application immediately after a cold or warm rest.

Required Vendor Information

VE01.08.01: The vendor shall specify in its documentation the default selected card application.

Required Test Procedures

TE01.08.01: The tester shall validate that there is a default selected card application which is the one specified by the vendor in VE01.08.01.

A.1.3 Security Architecture

A.1.3.1 Access Control Rule

AS01.09: The access control rule shall consist of an access mode and a security condition.

Note: This assertion is not separately tested.

AS01.10: The action described by the access mode can be performed on the data object if and only if the security condition evaluates to TRUE for the current values of the security status.

Note: This assertion is not separately tested.

AS01.11: If there is no access control rule with an access mode describing a particular action, then that action shall never be performed on the data object.

Note: This assertion is not separately tested.

A.1.3.2 Security Status

AS01.12: Associated with each authenticatable entity shall be a set of one or more Boolean variables each called a security status indicator of the authenticatable entity.

Note: The security status indicators will be tested indirectly through the functional testing.

AS01.13: The security status indicator of an authenticatable entity shall be TRUE if the credentials associated with the security status indicator of the authenticatable entity has been authenticated and FALSE otherwise.

Note: The security status indicators will be tested indirectly through the functional testing.

AS01.14: The successful execution of an authentication protocol shall set the security status indicator associated with the credentials that were verified by the protocol to TRUE.

Note: The security status indicators will be tested indirectly through the functional testing.

AS01.15: A security status indicator shall be said to be a global security status indicator if it not changed when the currently selected application changes from one application to another.

Note: This assertion is not separately tested.

AS01.16: A security status indicator is said to be an application security status indicator if it is set to FALSE when the currently selected application changes from one application to another.

Required Vendor Information

VE01.16.01: The vendor shall specify in its documentation that the application security status indicators will be set to FALSE when the currently selected application changes from one application to another.

Required Test Procedures

TE01.16.01: The tester shall visually validate that the vendor documentation contains the requirement stated in VE01.16.01.

A.1.3.3 Authentication of an Individual

AS01.17: Personal identification numbers presented to the card command interface shall be 8 bytes long.

Note: This assertion is tested as part of AS05.13.

AS01.18: If the actual PIN length is less than 8 bytes, it shall be padded to 8 bytes with 'FF' and appended to the actual PIN. The bytes comprising the PIN shall not include 'FF'.

Note: This assertion is tested as part of AS05.14.

A.1.4 PIV Card Application Status Variables

AS01.19: When the PIV Card Application is the currently selected application, the following status variables be associated with it.

- **Status Variable: Global Security Status Indicators– must always be defined. Can be used by all applications on the card platform. Maintained by: card platform.**
- **Status Variable: Currently selected application – must always be defined. The platform shall support the selection of a card application using a possibly right-truncated application identifier and there shall always be a currently selected application. Maintained by: card platform.**
- **Status Variable: Application security status Indicators – must always be defined. These indicators are local to the PIV Card Application. Maintained by: PIV Card Application.**

Note: This assertion is not separately tested.

A.2 End-Point Data Objects

A.2.1 PIV Card Application Data Objects

AS02.01: A PIV Card Application shall contain six mandatory data objects and five optional data object for interoperable use.

- **The six mandatory data objects are the following: 1. Card Capability Container 2. Card Holder Unique Identifier 3. X.509 Certificate for PIV Authentication 4. Card Holder Fingerprint I 5. Card Holder Fingerprint II 6. Security Object**
- **The five optional data objects for interoperable use are the following: 1. Card Holder Facial Image 2. Printed Information 3. X.509 Certificate for PIV Digital Signature 4. X.509 Certificate for PIV Key Management 5. X.509 Certificate for Card Authentication**

Note: This assertion is not separately tested.

A.2.2 OIDs and Tags of PIV Card Application Data Objects

AS02.02: For the purpose of constructing PIV Card Application data object names in the CardApplicationURL in Card Capability Container (CCC) of the PIV Card Application, the NIST RID ('A0 00 00 03 08') shall be used and the card application type shall be set to '00'.

Required Test Procedures

Note: This assertion is tested as part of AS02.03.

AS02.03: For all data objects present on the card, the object identifiers (OIDs) used by PIV Client Application to refer to them, and associated BER-TLV tags used by PIV Card Command Interface shall conform to the entries in Table 6 of SP 800-73.

Required Vendor Information

VE05.18.01: The vendor shall state in its documentation the list of all the data objects present on the card along with the OIDs and BER-TLV tags associated with them.

Required Test Procedures

TE02.03.01: The tester shall validate that the OIDs and BER-TLV tags of all the data objects present on the card conforms to the table A-1, and accurately represent the actual data objects observed by the tester as being implemented on the card.

A.3 End-Point Data Types and Their Representations

A.3.1 Algorithm Identifier

AS03.01: The algorithm identifiers for the cryptographic algorithms implemented on the card shall conform to entries in Table 7 of SP 800-73 along with default algorithm – 3 key Triple DES –ECB with algorithm identifier '00'.

Required Vendor Information

VE03.01.01: The vendor shall state the identifiers associated with all the algorithms supported by the card.

Required Test Procedures

TE03.01.01: The tester shall validate the presence of all the mandatory and optional algorithm identifiers (if implemented) on the vendor documentation and the card, and that they comply with Table A-2.

AS03.02: The default cryptographic algorithm for the PIV Card Application with algorithm identifier '00' is 3 Key Triple DES – ECB.

Required Vendor Information

VE03.02.01: The vendor shall state in its documentation that the 3 Key Triple Data Encryption Standard Electronic Code Book (DES-ECB) is the default cryptographic algorithm for the PIV Card Application and its algorithm identifier is ’00.’

Required Test Procedures

TE03.02.01: The tester shall validate that the default cryptographic algorithm is the 3 Key Triple DES-ECB with the identifier ’00.’

A.3.2 Application Property Template

AS03.03: Upon selection, the PIV Card Application shall return the application property template described in tables 8 and 9 of SP 800-73.

Required Vendor Information

VE03.03.01: The vendor shall provide in its documentation the PIV card application property template along with their TLVs.

Required Test Procedures

TE03.03.01: The tester shall visually validate that the information provided in response to VE03.03.01 is in conformance with Tables 8 and 9 of SP 800-73 .

TE03.03.02: The tester shall validate that the information provided in VE03.03.01, is actually implemented by the card.

A.3.3 Authenticator

AS03.04: The authenticator BER-TLV used on the PIV client-application programming interface shall have the structure described in Table 10 of SP 800-73.

Required Vendor Information

VE03.04.01: The vendor shall provide a list of all the authenticators along with their tags and possible values, when applicable.

Required Test Procedures

TE03.04.01: The tester shall visually validate that the vendor documentation states the correct tags for the "reference data" and "Key Reference" as shown in Table A-5.

TE03.04.02: The tester shall validate that the card returns the correct tags and values in the authenticator data object as specified in Table A-5.

A.3.4 Connection Description

AS03.05: The connection description BER-TLV used on the PIV client-application programming interface shall have the structure described in table 11 of SP 800-73.

Required Vendor Information

VE03.05.01: The vendor shall provide in its documentation the format and content of the Connection Description Templates implemented by the card.

Required Test Procedures

TE03.05.01: The tester shall validate the presence of the information provided in VE03.05.01 and that the Connection Description Template sent to the card conforms to Table A-6.

AS03.06: At most one selection from the '8x' series and one selection from the '9x' series shall appear in the connection description template as specified in AS03.05.

Note: This assertion is tested as part of AS03.05.

A.3.5 Key References

AS03.07: The key reference, when represented as a byte, occupies b8 and b5-b1 while b7 and b6 shall be set to 0.

Note: This assertion is not separately tested.

AS03.08: The key references used on all PIV interfaces shall be from the list found in Table 12 of SP 800-73 along with those in the document titled "Namespace Management for Personal Identity Verification (PIV) Applications and Data Objects" that is posted on the NIST Web Page.

Note: This assertion is not separately tested.

AS03.09: The card holder global PIN's value shall not be changed nor shall its retry counter be reset while the PIV Card Application is the currently selected application.

Required Vendor Information

VE03.09.01: The vendor shall specify in its documentation that the card conforms to the assertion stated in AS03.09.

Required Test Procedures

TE03.09.01: The tester shall select the PIV Card Application and attempt to change global PINand validate that the global PIN value did not change and that its retry counter was not reset.

TE03.09.02: The tester shall visually validate the presence of the information required in VE03.09.01

A.3.6 Status Words

AS03.10: A status word shall be a 2-byte value returned by an entry point on the client-application programming interface or a card command at the card edge.

Note: This assertion is not separately tested – since it is part of the process of testing status words (return codes) for every command in PIV Client API and PIV card command interfaces.

AS03.11: Recognized values of all SW1-SW2 pairs used as return values on both the PIV client-application programming and PIV card command interfaces shall be from the list provided in Table 13 of SP 800-73.

Note: This assertion is not separately tested.

AS03.12: A data object shall be identified on the PIV client-application programming interface using its OID.

Note: This assertion is not separately tested.

AS03.13: An object identifier on the PIV client-application programming interface shall be a dot delimited string of the integer components of the OID.

Note: This assertion is not separately tested.

AS03.010: A data object shall be identified on the PIV Card Application card command interface using its BER-TLV tag.

Note: This assertion is not separately tested.

A.4 End-Point Client-Application Programming Interface

AS04.01: Entry points on the PIV client-application programming interface shall include all functions listed in Table 14 of SP 800-73.

Note: This assertion is tested as part of AS04.02 through AS04.10.

A.4.1 Entry Points for Communication

Required Vendor Information & Required Test Procedures

To test the entry points or commands that should be supported by client application, the only information that the vendor has to provide is the PIV Card Application version that the client application supports. All parameter values for exercising the commands have to be obtained from the PIV Card Application vendor documentation, using the mapping of client application entry point commands to the PIV Card Application card commands. This mapping is given in the Table A-1 below. Hence this section contains only tester requirements in terms of Required Test Procedures.

Client Application Entry Points	PIV Card Application Card Command[5]	Mapping Description
pivConnect	No Equivalent command	For establishing a connection session with card reader.
pivDisconnect	No Equivalent command	For disconnecting a connection session with the card reader
pivSelectCardApplication	SELECT	Passes the AID value. Sets the value for 'Currently Selected Application' on the PIV card
pivLogIntoCardApplication	VERIFY	Provides the key reference for PIV Card Application PIN as well as the PIN string and passes. Sets the PIV Card Application Security Status on the card
pivLogOutOfCardApplication	RESET (not specified in SP 800-73)	Resets the PIV Card Application Security Status on the card
pivGetData	GET DATA	Maps the Object Identifier (OID) to BER-TLV tag for the selected object
pivPutData	PUT DATA	Maps the OID to BER-TLV tag for the selected object
pivGenerateKeyPair	GENERATE ASSYMMETRIC KEY PAIR	Passes the Key Reference and Cryptographic Mechanism identifier values
pivCrypt	GENERAL AUTHENTICATE	Passes the Key Reference, Cryptographic Algorithm reference and the string to be acted upon.

Table A-1 PIV Command Mapping

A.4.1.1 pivConnect

AS04.02: The pivConnect's purpose is to connect the client-application programming interface and hence the client application itself to the PIV Card Application on a specific ICC.

[5] It is assumed that some of these functions will use GET RESPONSE and chaining to accomplish the read or write to the card.

TE04.02.01: The tester shall validate that the client application implements the pivConnect as per SP800-73.

A.4.1.2 pivDisconnect

AS04.03: The piv Disconnect's purpose is to disconnect the PIV application programming interface from the PIV Card Application and the ICC containing the PIV Card Application.

TE04.03.01: The tester shall validate that the client application implements the pivDisconnect as per SP800-73.

A.4.2 Entry Points for Data Access

A.4.2.1 pivSelectCardApplication

AS04.04: The pivSelectCardApplication sets the currently selected card application.

TE04.04.01: The tester shall validate that the client application implements the pivSelectCardApplication as per SP800-73.

A.4.2.2 pivLogIntoCardApplication

AS04.05: The pivLogIntoCardApplication establishes application security status within the PIV Card Application.

TE04.05.01: The tester shall validate that the client application implements the pivLogIntoCardApplication as per SP800-73.

A.4.2.3 pivGetData

AS04.06: The pivGetData returns the entire data content of the named data object.

TE04.06.01: The tester shall validate that the client application implements the pivGetData as per SP800-73.

A.4.2.4 pivLogoutOfCardApplication

AS04.07: The pivLogoutOfCardApplication resets the application security status of the PIV Card Application. The currently selected application after successful return from this entry point is platform-dependent.

TE04.07.01: The tester shall validate that the client application implements the pivLogoutOfCardApplication as per SP800-73.

A.4.3 Entry Points for Cryptographic Operations

A.4.3.1 pivCrypt

AS04.08: pivCrypt perform a cryptographic operation such as encryption or signing on a sequence of bytes.

TE04.08.01: The tester shall validate that the client application implements the pivCrypt as per SP800-73.

A.4.4 Entry Points for Credential Initialization and Administration

A.4.4.1 pivPutData

AS04.09: The pivPutData replaces the entire data content of the named data object with the provided data.

TE04.09.01: The tester shall validate that the client application implements the pivPutData as per SP800-73.

A.4.4.2 pivGenerateKeyPair

AS04.10: The pivGenerateKeyPair generates an asymmetric key pair in the currently selected application.

TE04.10.01: The tester shall validate that the client application implements the pivGenerateKeyPair as per SP800-73.

A.5 End-Point PIV Card Application Card Command Interface

AS05.01: All PIV Card Application card commands listed in Table 15 of SP 800-73 (interpreted in conjunction with associated Errata published in NIST Web page) shall be supported by a PIV Card Application.

Required Vendor Information

VE05.01.01: The vendor shall provide the list of all PIV Card Application card commands, along with the interface(s) (contact or contactless) they support, the security condition(s) they are subject to and their support for command chaining as implemented by the card.

Required Test Procedures

TE05.01.01: The tester shall validate that the information presented in response to VE05.01.01 by the vendor complies with Table 15 of SP 800-73..

TE05.01.02: The tester shall validate that the card implements all the commands as required in Table 15 of SP 800-73

TE05.01.03: The tester shall validate that the commands are implemented only through the interfaces allowed as shown in Table 15 of SP 800-73

TE05.01.04: The tester shall validate that the commands are implemented only after the security condition associated wit hem are satisfied, as shown in the table, via the specified interface.

TE05.01.05: The tester shall validate that only the commands as indicated in the table are allowed for chaining via the interface supported after the security condition is satisfied.

AS05.02: Card commands indicated with a 'Yes' in the Command Chaining column shall support command chaining for transmitting a data string too long for a single command as defined in ISO/IEC 7816-4 [1].

Note: This assertion is tested as part of AS05.01.

AS05.03: The PIV Card Application shall return the status word of ‘6A81’ (Function not supported) when it receives a card command on the contactless interface marked “No” in the Contactless Interface column in the table in AS05.01.

Note: This assertion is tested as part of AS05.01.

AS05.04: Cryptographic protocols using asymmetric keys that require PIN shall not be used on the contactless interface.

Note: This assertion is not separately tested.

A.5.1 PIV Card Application Card Commands for Data Access

A.5.1.1 SELECT Card Command

AS05.05: The PIV Card Application shall be selected by providing its application identifier ‘A0 00 00 03 08 00 00 10 00 vv vv’ in the data field of the SELECT command where 'vv vv' is the version of the PIV Card Application to be made the currently selected application.

Required Vendor Information

VE05.05.01: The vendor shall specify in its documentation the PIV Card Application Identifier.

Required Test Procedures

TE05.05.01: The tester shall validate that the PIV Card Application is selected by providing its application identifier as specified in VE05.05.01.

AS05.06: There shall be at most one PIV Card Application on any ICC.

Required Vendor Information

VE05.06.01: The vendor shall state in its documentation that there is only one PIV Card Application on the ICC.

Required Test Procedures

TE05.06.01: The tester shall visually validate the vendor documentation for the information provided in VE05.06.01.

AS05.07: The PIV Card Application can also be made the currently selected application by providing a right-truncated version; that is, without the two-byte version number, 'vv vv'; in the data field of the SELECT command 'A0 00 00 03 08 00 00 10 00'

Required Vendor Information

VE05.07.01: The vendor shall specify in its documentation whether the card implements the application selection by the right-truncated version.

Required Test Procedures

TE05.07.01: The tester shall visually validate that the information in VE05.07.01 is present in the vendor documentation..

TE05.07.02: If the card implements the short version application selection, the tester shall validate that the PIV application is selectable by the right-truncated SELECT command.

AS05.08: The complete AID, including the two-byte version, of the PIV Card Application that became the currently selected application upon successful execution of the SELECT command shall be returned in the application property template.

Note: This assertion is tested as part of AS03.03.

AS05.09: If the currently selected application is the PIV Card Application when the SELECT APPLICATION command is given and the AID in the data field of the SELECT APPLICATION is either the AID of the PIV Card Application or a right-truncated version thereof, then the PIV Card Application shall continue to be the currently selected application and the setting of all security status indicators in the PIV Card Application shall be unchanged.

Required Vendor Information

VE05.18.01: The vendor shall provide information in its documentation validating the compliance with the statement in AS05.09

Required Test Procedures

TE05.09.01: The tester shall validate that when the currently selected application is the PIV Card Application, if the SELECT APPLICATION command is sent with an AID that is either the AID of the PIV Card Application or its right-truncated version, then the PIV Card Application shall continue to be the currently selected application and the setting of all security status indicators in the PIV Card Application shall be unchanged

AS05.10: If the currently selected application is the PIV Card Application when the SELECT APPLICATION command is given and the AID in the data field of the SELECT APPLICATION is neither the AID of the PIV Card Application nor a right-truncated version thereof, then the PIV Card Application shall continue to be the currently selected application and the setting of all security status indicators in the PIV Card Application shall be unchanged

Required Vendor Information

VE05.10.01: The vendor shall provide information in its documentation validating the compliance with the statement in AS05.10

Required Test Procedures

TE05.10.01: The tester shall validate that when the currently selected application is the PIV Card Application, if the SELECT APPLICATION command is sent with an AID that is neither the AID of the PIV Card Application nor its right-truncated version, the PIV Card Application continues to be the currently selected application.

A.5.1.2 GET DATA Card Command

AS05.11: The GET DATA card command retrieves the data content of the single data object whose tag is given in the data field.

Note: This assertion is tested as part of AS05.01

AS05.11A: The GET DATA card command retrieves the data content of the data object only after the access rule associated with the data object (Appendix A of SP 800-73) evaluates to TRUE.

Required Vendor Information

VE05.11A.01: The vendor shall specify in its documentation the access rule for each of the data objects or make a reference to the table in Appendix A of SP 800-73.

Required Test Procedures

TE05.11A.01: The Tester shall validate that all data objects that require the PIV card application PIN shall only be accessible after a successful validation of that PIN (through VERIFY command).

TE05.11A.02: The Tester shall validate that all data objects whose access rule is ”Always Read ” shall only be accessible with or without PIV Card Application PIN validation.

A.5.2 PIV Card Application Card Commands for Authentication

A.5.2.1 VERIFY Card Command

AS05.12: Only key references specific to the PIV Card Application; i.e. local key references, shall be verified by the PIV Card Application VERIFY command.

Note: This assertion is not separately tested.

AS05.13: If the current value of the retry counter associated with the key reference is zero, then the comparison of the value of PIN input with the PIN associated with key reference shall not be made and the PIV Card Application shall return the status word '69 83'.

Required Vendor Information

VE05.13.01: The vendor shall specify in its documentation the reset value of the retry counters associated with all the key references.

Required Test Procedures

TE05.13.01: The tester shall validate that the PIV Card Application returns ’69 83’ in response to the VERIFY command, when the retry counter associated with the key reference is zero.

AS05.14: If the card command succeeds, then the security status of the key reference shall be set to TRUE and the retry counter associated with the key reference shall be set to the reset retry value associated with the key reference.

Required Vendor Information

VE05.14.01: The vendor shall provide in its documentation the reset retry values associated with each key reference supported by the card.

Required Test Procedures

TE05.14.01: The tester shall validate that the retry counter associated with the key reference shall be set to the reset retry value specified by the vendor in VE05.14.01 (not decremented), when the VERIFY command succeeds.

AS05.15: If the card command fails, then the security status of the key reference shall be set to FALSE and the retry counter associated with the key reference shall be decremented by one.

Required Vendor Information

VE05.15.01: The vendor shall provide in its documentation the reset retry values associated with each key reference implemented by the card.

Required Test Procedures

TE05.15.01: The tester shall validate that when the VERIFY command fails, the retry counter associated with the key reference is decremented by one.

A.5.2.2 CHANGE REFERENCE DATA Card Command

AS05.16: If the current value of the retry counter associated with the key reference is zero, then the reference data associated with the key reference shall not be changed and the PIV Card Application shall return the status word '69 83'

Required Vendor Information

VE05.16.01: The vendor shall state in its documentation that the card supports the assertion made in AS05.016.

Required Test Procedures

TE05.16.01: The tester shall validate that when the current value of the retry counter associated with the key reference is zero, the reference data associated with the key reference does not change and the PIV Card Application returns '69 83'

AS05.17: If the card command succeeds, then the security status of the key reference shall be set to TRUE and the retry counter associated with the key reference shall be set to the reset retry value associated with the key reference

Required Vendor Information

VE05.17.01: The vendor shall state in its documentation that the card supports the assertion made in AS05.17.

Required Test Procedures

TE05.17.01: The tester shall validate that the vendor documentation states the required information in VE05.17.01 and the retry counter associated with the key reference shall be set to the reset retry value associated with the key reference when the command succeeds.

AS05.18: If the card command fails, then the security status of the key reference shall be set to FALSE and the retry counter associated with the key reference shall be decremented by one

Required Vendor Information

VE05.18.01: The vendor shall state in its documentation that the card supports the assertion made in AS05.18.

Required Test Procedures

TE05.18.01: The tester shall validate that the vendor documentation contains the information required in VE05.18.01 and the retry counter associated with the key reference shall be decremented by one if the card command fails.

AS05.19: If the either the current reference data or the new reference data in the command field of the command does not satisfy the criteria in Section 3.5.3 of SP 800-73, the PIV Card Application shall not change the reference data associated with the key reference and shall return the status word '6A 80'.

Required Vendor Information

VE05.19.01: The vendor shall state in its documentation that the card supports the assertion made in AS05.19.

Required Test Procedures

TE05.19.01: The tester shall validate that the vendor documentation contains the information required in VE05.19.01 and the old PIN is not changed and the card returns status word ‘6A 80’, when either of the PIN information in the reference data field of the command is not padded to 8 bytes,

A.5.2.3 RESET RETRY COUNTER Card Command

AS05.20: Only retry counters associated with key references specific to the PIV Card Application; i.e. local key references, shall be reset by the PIV Card Application RESET RETRY COUNTER command.

Note: This assertion is not tested separately.

AS05.21: If the current value of the reset counter associated with the key reference is zero, then retry counter associated with the key reference shall not be reset and the PIV Card Application shall return the status word '69 83'.

Required Vendor Information

VE05.21.01: The vendor shall provide in its documentation the reset values associated with each key reference supported by the card.

VE05.21.02: The vendor shall specify in its documentation that the RESET RETRY COUNTER card command will not reset the retry counter and the card will return '69 83', when the reset counter associated with the key reference is zero.

Required Test Procedures

TE05.21.01: The tester shall validate that the information requested in VE05.21.02 and VE05.21.01 are present in the vendor documentation. (NOTE: Testing this condition will leaves the card unusable for further tests since since the reset counter is zero).

AS05.22: If the card command succeeds, then the retry counter associated with the key reference shall be set to the reset retry value associated with the key reference. Neither the security status of the key reference or the reset counter shall be changed.

Required Vendor Information

VE05.22.01: The vendor shall specify in its documentation a list of reset retry values of the retry counters associated with each key reference on the card.

VE05.22.02: The vendor shall specify in its documentation that the card supports the assertion made in AS05.22

Required Test Procedures

TE05.22.01: The tester shall validate that when the card command succeeds, PIN retry counter is set to PIN reset retry value specified in VE05.22.01, and neither the security status of the key reference or the reset counter is changed.

AS05.23: If the card command fails, then the security status of the key reference shall be set to FALSE and the reset counter associated with the key reference shall be decremented by one.

Required Vendor Information

VE05.23.01: The vendor shall state in its documentation that card supports the assertion made in AS05.023

Required Test Procedures

TE05.23.01: The tester shall validate that the information requested in VE05.23.01 is present in the vendor documentation and, the security status of the key reference is set to FALSE and the reset counter is decremented by one

AS05.24: If the either the reset retry counter reference data (PUK) or the new reference data (PIN) in the command field of the command does not satisfy the criteria in Section 3.5.3 of SP 800-73, the PIV Card Application shall not reset the retry counter associated with the key reference and shall return the status word '6A 80'.

Required Vendor Information

VE05.24.01: The vendor shall state in its documentation that the card supports the assertion made in AS05.24

Required Test Procedures

TE05.24.01: The tester shall validate that the vendor documentation includes the information required in VE05.24.01 and that when either the PUK or the PIN of the command does not satisfy the criteria in Section 3.5.3, the retry counter is not reset and the card returns ‘6A 80.’

A.5.2.4 GENERAL AUTHENTICATE Card Command

AS05.25: The GENERAL AUTHENTICATE command shall be used to authenticate the card or a card application to the client-application (INTERNAL AUTHENTICATE), to authenticate an entity to the card (EXTERNAL AUTHENTICATE), and to perform a mutual authentication between the card and an entity external to the card (MUTUAL AUTHENTICATE).

Required Vendor Information

VE05.25.01: The vendor shall specify in its documentation the types of authentications supported by the card.

Required Test Procedures

TE05.25.01: The tester shall validate that the GENERAL AUTHENTICATE command is implemented to authenticate the Card to the client application.

TE05.25.02: The tester shall validate that the GENERAL AUTHENTICATE command is implementd to authenticate an entity to the card.

TE05.25.03: The tester shall validate that the GENERAL AUTHENTICATE command is implemented to mutually authenticate the Card to an entity and the entity to the card.

AS05.26: The GENERAL AUTHENTICATE command shall be implementd to realize the signing functionality on the PIV client-application programming interface.

Note: This assertion is not separately tested.

AS05.27: If a card command other than the GENERAL AUTHENTICATE command is received by the PIV Card Application before the termination of a GENERAL AUTHENTICATE chain, the PIV Card Application shall rollback to the state it was in immediately prior to the reception of the first command in the interrupted chain.

Required Vendor Information

VE05.18.01: The vendor shall specify in its documentation that the card supports the assertion made in AS05.27.

Required Test Procedures

TE05.27.01: The tester shall validate that the PIV Card Application reverts back to the state it was in if a command other than GENERAL AUTHENTICATE is received before the termination of a GENERAL AUTHENTICATE chain.

A.5.3 PIV Card Application Card Commands for Credential Initialization and Administration

A.5.3.1 PUT DATA Card Command

AS05.28: The PUT DATA card command completely replaces the data content of a single data object in the PIV Card Application with new content.

Required Vendor Information

VE05.28.01: The vendor shall specify in its documentation the format, encoding and the parameters of the PUT DATA command supported by the card.

Required Test Procedures

TE05.28.01: The tester shall validate that the card complies with the PUT DATA command as defined in SP800-73.

A.5.3.2 GENERATE ASYMMETRIC KEY PAIR Card Command

AS05.29: The GENERATE ASYMMETRIC KEY PAIR card command initiates the generation and storing in the card of the reference data of an asymmetric key pair, i.e., a public key and a private key, and the command returns the public key.

Required Vendor Information

VE05.29.01: The vendor shall specify in its documentation the cryptographic module identifiers (from Table 20 of SP 800-73) that have been implemented on the card.

Required Test Procedures

TE05.29.01: The tester shall validate that the card implements the algorithms associated with identifiers specified as part of VE05.29.01 requirement and that the public key returned is formatted based on data object tags specified in Table 21 of SP 800-73.

AS05.30: The public key of the generated key pair is returned as the response to the command.

Note: This assertion is tested as part of AS05.29.

AS05.31: If there is reference data currently associated with the key reference, it is replaced in full by the generated data.

Required Vendor Information

VE05.31.01: The vendor shall provide the contents of the public key data on the card.

Required Test Procedures

TE05.31.01: The tester shall validate that the initial contents of the public key data is replaced in full by the generated data, following a GENERATE ASYMMETRIC KEY PAIR command.

A.6 PIV Data Model

AS06.01: Part 3 conformant cards shall return all the TLV elements of a container in the physical order listed for that container in this data model.

Required Vendor Information

VE06.01.01: The vendor shall specify in its documentation the format (TLV) and the content of all the elements in each data container on the card.

VE06.01.02: The vendor shall specify in its documentation that the information provided in VE06.01.01 conforms to SP800-73.

Required Test Procedures

TE06.01.01: The tester shall validate that the formatting, encoding and the content of all the elements in each data container conforms to SP800-73.

AS06.02: Both single-chip/dual-interface and dual-chip implementations shall be feasible.

Note: This assertion is not separately tested.

AS06.03: In the single-chip/dual-interface configuration, the PIV Card Application shall be provided the information regarding which interface is in use.

Required Vendor Information

VE06.03.01: The card operating system should inform the PIV Card Application the communication interface in use.

Required Test Procedures

TE06.03.01: The tester shall validate that the card platform informs the PIV Card Application of the interface being used.

TE06.03.02: The tester shall validate that the PIV Card Application checks that a contact interface is being used for contact-only APDUs.

AS06.04: In the dual-chip configuration, a separate PIV Card Application shall be loaded on each chip.

Note: This assertion is not separately tested.

Appendix B—PIV Client API Test Assertions

Test Assertion Template

Purpose	A quick description of the test and why it is being run
Target	The pivFunction being tested
Reference(s)	References to the SP800-73 or other relevant publications
Precondition(s)	Anything that must be done or known prior to executing the scenario
Test Steps	`Sequence of steps for making a function call`
Expected Result(s)	What the expected execution path yields in terms of progress and values
Post Condition(s)	A description of both client and card application state once the test scenario completes

B.1 Connection Test Assertions

B.1.1 Valid Path Test Assertions

B.1.1.1 Initiate Standalone Connection

Purpose	Confirms that a single connection can be obtained by a calling application to the PIV Card Application on a specific ICC
Target	`pivConnect`
Reference(s)	1. SP800-73, Section 6.1.1 2. AS03.05, AS04.01, AS04.02
Precondition(s)	1. A valid connection description is provided for the card application 2. There exists a valid physical connection between an instance of the PIV Card and the host of the calling application 3. No application is currently connected to the PIV Card Application

Test Steps	`1. Set sharedConnection := false`
	`2. Set connectionDescription := <<valid connection>>`
	`3. Create cardHandle reference`
	`4. Call pivConnect w/`
	• *(IN)* `sharedConnection`
	• *(INOUT)* `connectionDescription`
	• *(OUT)* `cardHandle`

Expected Result(s)	Call returns with status_word of PIV_OK and initialized cardHandle
Post Condition(s)	Client Application is Connected to PIV Card

B.1.1.2 Initiate Shared Connection

Purpose	Confirms that a shared connection can be established by two distinct calling applications to the PIV Card with a specific ICC
Target	`pivConnect`
Reference(s)	1. SP800-73, Section 6.1.1 2. AS04.02
Precondition(s)	1. A valid connection description is provided for the card application 2. There exists valid physical connection between an instance of the PIV Card and the host of the calling application 3. Another client application is currently connected via a shared connection to the PIV Card Application.

Test Steps	`1. Set sharedConnection := true`
	`2. Set connectionDescription := <<valid description>>`
	`3. Create cardHandle reference`
	`4. Call pivConnect w/`
	• *(IN)* `sharedConnection`
	• *(INOUT)* `connectionDescription`
	• *(OUT)* `cardHandle`

Expected Result(s)	Call returns with status_word := PIV_OK and initialized cardHandle
Post Condition(s)	Both client applications are connected through the same connection to PIV Card Application.

B.1.2 Test Assertions for Error Conditions

B.1.2.1 Malformed Connection Description

Purpose	Confirms that the correct status word is returned when a malformed connection description is used
Target	`pivConnect`
Reference(s)	1. SP800-73, Section 6.1.1 2. AS04.02
Precondition(s)	1. An invalid connection description is provided for the card application 2. There exists valid physical connection between an instance of the PIV Card and the host of the calling application

Test Steps	`1. Set sharedConnection := true | false`
	`2. Set connectionDescription := <<invalid connection>>`
	`3. Create cardHandle reference`
	`4. Call pivConnect w/`
	• *(IN)* `sharedConnection`
	• *(INOUT)* `connectionDescription`
	• *(OUT)* `cardHandle`

Expected Result(s)	Call returns with status_word := `PIV_CONNECTION_DESCRIPTION_MALFORMED`
Post Condition(s)	1. The cardHandle variable is not initialized

	2. The Client Application is not connected to the PIV Card Application

B.1.2.2 Sharing a Locked Connection

Purpose	Ensure that when a connection is initially established unshared that no additional connections can be established
Target	`pivConnect`
Reference(s)	1. SP800-73, Section 6.1.1 2. AS04.02
Precondition(s)	1. A valid connection description is provided for the card application 2. There exists valid physical connection between an instance of the PIV Card and the host of the calling application 3. An application currently owns an exclusive connection (`sharedConnection := false`)

Test Steps	`1. Set sharedConnection := true | false`
	`2. Set connectionDescription := <<valid connection>>`
	`3. Create cardHandle reference`
	`4. Call pivConnect w/`
	• *(IN)* `sharedConnection`
	• *(INOUT)* `connectionDescription`
	• *(OUT)* `cardHandle`

Expected Result(s)	Call returns with status_word := `PIV_CONNECTION_LOCKED`
Post Condition(s)	1. The Client Application previously connected remains connected 2. The cardHandle variable is not initialized 3. The newly requesting Client Application is not connected to the PIV Card Application

B.1.2.3 Attempting to Lock a Shared Connection

Purpose	Ensure that a Client Application cannot lock a PIV application connection that currently has open shared connections
Target	`pivConnect`
Reference(s)	1. SP800-73, Section 6.1.1 2. AS04.02
Precondition(s)	1. A valid connection description is provided for the card application 2. There exists valid physical connection between an instance of the PIV Card and the host of the calling application 3. An application currently owns a shared connection (`sharedConnection := true`)

Test Steps	`1. Set sharedConnection := false`
	`2. Set connectionDescription := <<valid connection>>`
	`3. Create cardHandle reference`
	`4. Call pivConnect w/`

	• *(IN)*sharedConnection • *(INOUT)* connectionDescription • *(OUT)* cardHandle
Expected Result(s)	Call returns with status_word := PIV_CONNECTION_FAILURE
Post Condition(s)	1. The Client Application previously connected remains connected 2. The cardHandle variable is not initialized 3. The newly requesting Client Application is not connected to the PIV Card Application

B.1.2.4 Attempting to Open an Unsupported Connection

Purpose	Confirms that the correct status word is returned when an unsupported connection mode is attempted.
Target	pivConnect
Reference(s)	1. SP800-73, Section 6.1.1 2. AS04.02
Precondition(s)	1. An invalid connection mode (e.g. Integrated Services Digital Network (ISDN)) is attempted 2. There exists valid physical connection between an instance of the PIV Card and the host of the calling application
Test Steps	1. Set sharedConnection := true | false 2. Set connectionDescription := <<valid ISDN connection string>> 3. Create cardHandle reference 4. Call pivConnect w/ • *(IN)* sharedConnection • *(INOUT)* connectionDescription • *(OUT)* cardHandle
Expected Result(s)	Call returns with status_word := PIV_CONNECTION_FAILURE
Post Condition(s)	1. The cardHandle variable is not initialized 2. The Client Application is not connected to the PIV Card

B.2 Disconnection Test Assertions

B.2.1 Valid Test Assertions

B.2.1.1 Disconnect a Standalone Connection

Purpose	Ensure that a Client Application can close a currently open exclusive PIV application connection
Target	pivDisconnect
Reference(s)	1. SP800-73, Section 6.1.2 2. AS03.05, AS04.01, AS04.03
Precondition(s)	1. There exists a valid physical and logical connection between an instance of the PIV Card and the host of the calling application 2. A client application currently has a connection accessible through

	`cardHandle`
Test Steps	`1. Call pivDisconnect w/ arguments` • *`(IN)`* `cardHandle`
Expected Result(s)	Call returns with status_word := `PIV_OK`
Post Condition(s)	1. The client application is no longer connected to the PIV card application 2. PIV Card Application is no longer aware of the Client Application

B.2.1.2 Disconnect a Shared Connection

Purpose	Ensure that a Client Application can close a currently open and shared PIV Card Application connection without impacting other Client Application's connections to that same PIV card application
Target	`pivDisconnect`
Reference(s)	1. SP800-73, Section 6.1.2 2. AS04.03
Precondition(s)	1. There exists a valid logical and physical connection between an instance of the PIV Card and the host of the calling application 2. At least two distinct client applications (having two distinct `CardHandle references`) are connected to the PIV card application
Test Steps	`1. Call pivDisconnect w/ arguments` • *`(IN)`* `cardHandle`
Expected Result(s)	Call returns with status_word := `PIV_OK`
Post Condition(s)	1. The Client Application is no longer connected to the PIV card application 2. All other Client Applications maintain their previously valid connections 3. PIV Card Application is no longer aware of that particular Client Application but remains aware of all other Client Applications.

B.2.2 Test Assertions for Error Cases

B.2.2.1 Logically disconnecting a Client Application from a physically disconnected Card Application

Purpose	Checks the response to the case where a physical disconnection precedes a logical disconnection between Client Application and Card Application
Target	`pivDisconnect`
Reference(s)	1. SP800-73, Section 6.1.2 2. AS04.03
Precondition(s)	1. There exists a client application with a valid and open `cardHandle:CardHandle` to a PIV Card Application

	2. The card is not currently physically connected to the card reader
Test Steps	1. `Call pivDisconnect w/ arguments` • *(IN)* `cardHandle`
Expected Result(s)	Call returns with status_word := `PIV_CONNECTION_ERROR`
Post Condition(s)	The Client Application is no longer connected to the PIV card application

B.2.2.2 Attempt Disconnect with Invalid Card Handle

Purpose	Ensure that the Client Application can detect an invalid cardHandle argument.
Target	`pivDisconnect`
Reference(s)	1. SP800-73, Section 6.1.2 2. AS04.03
Precondition(s)	1. There exists a valid physical and logical connection between an instance of the PIV Card and the host of the calling application 2. A client application currently has a connection accessible through `cardHandle`
Test Steps	1. `Set cardHandle := <<invalid cardHandle>>` 2. `Call pivDisconnect w/` • *(IN)* `cardHandle`
Expected Result(s)	Call returns with status_word := `PIV_INVALID_CARD_HANDLE`
Post Condition(s)	The client application remains connected to the PIV card application

B.2.2.3 Disconnecting a previously disconnected Client Application

Purpose	Ensure that a Client Application can close a previously closed PIV Card Application connection without impacting other Client Application's connections to that same PIV card application
Target	`pivDisconnect`
Reference(s)	1. SP800-73, Section 6.1.2 2. AS04.03
Precondition(s)	1. There exists a client application with a valid and open `cardHandle:CardHandle` to a PIV Card Application 2. The subject connection was previously closed 3. The card is physically connected to the card reader
Test Steps	1. `Call pivDisconnect w/ arguments` • *(IN)* `cardHandle`
Expected Result(s)	Call returns with status_word := `PIV_CONNECTION_ERROR`
Post Condition(s)	1. The Client Application is no longer connected to the PIV card application 2. PIV Card Application is no longer aware of that particular Client Application

B.3 pivSelectCardApplication

B.3.1 Valid Test Assertions

B.3.1.1 Select a Card Application with a full AID

Purpose	Ensure that a Client Application can locate and select a valid Card Application, store its properties, and return a reference to the applicationProperties.
Target	`pivSelectCardApplication`
Reference(s)	1. SP800-73, Section 6.2.1 2. AS03.05, AS04.01, AS04.04
Precondition(s)	1. The Client Application currently owns a connection accessible through cardHandle.
Test Steps	1. Set cardHandle := <<valid cardHandle>> 2. Set applicationID := <<valid applicationID>> 3. Create applicationProperties reference 4. Call pivSelectCardApplication w/ • *(IN)* cardHandle • *(IN)* applicationID • *(OUT)* applicationProperties
Expected Result(s)	Call returns with status_word of `PIV_OK` and initialized applicationProperties reference
Post Condition(s)	The "CurrentlySelectedApplication" in PIV Card is the PIV Card Application

B.3.1.2 Use a right truncated AID to Select a Card Application

Purpose	Ensure that a Client Application is able to locate and select a valid Card Application that is identified by a right truncated AID, store its properties, and return a reference to the applicationProperties.
Target	`pivSelectCardApplication`
Reference(s)	1. SP800-73, Section 6.2.1 2. AS04.04
Precondition(s)	1. The Client Application currently owns a connection accessible through cardHandle.
Test Steps	1. Set cardHandle := <<valid cardHandle>> 2. Set applicationID := <<valid right truncated applicationID>> 3. Create applicationProperties reference 4. Call pivSelectCardApplication w/ • *(IN)* cardHandle • *(IN)* applicationID • *(OUT)* applicationProperties
Expected Result(s)	1. Call returns with status_word of `PIV_OK` and initialized

	applicationProperties reference
Post Condition(s)	The "CurrentlySelectedApplication" in PIV Card is the PIV Card Application

B.3.2 Test Assertions for Error Conditions

B.3.2.1 Detect and handle an invalid cardHandle reference.

Purpose	Ensure that a Client Application can detect and gracefully exit when passed an invalid cardHandle.
Target	`pivSelectCardApplication`
Reference(s)	1. SP800-73, Section 6.2.1 2. AS04.04
Precondition(s)	1. An invalid cardHandle is passed to the client application. 2. The applicationAID is assumed to be valid.
Test Steps	1. Set cardHandle := <<invalid cardHandle>> 2. Set applicationID := <<valid applicationID>> 3. Create applicationProperties reference 4. Call pivSelectCardApplication w/ • *(IN)* cardHandle • *(IN)* applicationID • *(OUT)* applicationProperties
Expected Result(s)	Call returns with status_word of `PIV_INVALID_CARD_HANDLE` and does not initialize applicationProperties reference
Post Condition(s)	The Client Application returns to the state it had prior to calling `pivSelectCardApplication`.

B.3.2.2 Detect and handle an invalid applicationAID.

Purpose	Ensure that a Client Application can detect and gracefully exit when passed an invalid applicationAID.
Target	`pivSelectCardApplication`
Reference(s)	1. SP800-73, Section 6.2.1 2. AS04.04
Precondition(s)	1. A correctly formatted but invalid applicationAID is passed to the client application. 2. The cardHandle is assumed to be valid.
Test Steps	1. Set cardHandle := <<valid cardHandle>> 2. Set applicationID := <<invalid applicationID>> 3. Create applicationProperties reference 4. Call pivSelectCardApplication w/ • *(IN)* cardHandle • *(IN)* applicationID • *(OUT)* applicationProperties
Expected Result(s)	Call returns with status_word of `PIV_CARD_APPLICATION_NOT_FOUND` and does not initialize applicationProperties reference

Post Condition(s)	The Client Application returns to the state it had prior to calling `pivSelectCardApplication`.

B.4 pivLogIntoCardApplication

B.4.1 Valid Test Assertions

B.4.1.1 Log on to the Card Application

Purpose	Validate that the Client Application can establish application security status with the selected PIV Card Application.
Target	`pivLogIntoCardApplication`
Reference(s)	1. SP800-73, Section 6.2.2 2. AS03.05, AS04.01, AS04.05
Precondition(s)	1. The card has established a connection to the client. 2. The cardHandle was properly initialized by pivConnect. 3. The client application has successfully executed the pivSelectCardApplication command.

Test Steps	`1. Set cardHandle := <<a valid cardHandle>>`
	`2. Set authenticators := <<valid authenticators byte sequence>>`
	`3. Call pivLogIntoCardApplication w/`
	• *(IN)* `authenticators`
	• *(IN)* `cardHandle`

Expected Result(s)	Call returns with status_word of `PIV_OK`
Post Condition(s)	Security context is established and the Client Application can now perform read / write operations on the data objects controlled by PIV Card Application. The client is thus logged into Card Application.

B.4.2 Test Assertions for Error Conditions

B.4.2.1 Attempt Logon without an invalid cardHandle.

Purpose	Ensure a Client Application can detect and process an invalid card handle.
Target	`pivLogIntoCardApplication`
Reference(s)	1. SP800-73, Section 6.2.2 2. AS04.05
Precondition(s)	1. The card has established a connection to the client. 2. The cardHandle was properly initialized by pivConnect. 3. The client application has successfully executed the pivSelectCardApplication command.
Test Steps	`1. Set cardHandle := <<an invalid cardHandle>>`

	2. Set authenticators := <<valid authenticators byte sequence>> 3. Call pivLogIntoCardApplication w/ • *(IN)* authenticators • *(IN)* cardHandle
Expected Result(s)	Call returns with status_word := PIV_INVALID_CARD_HANDLE
Post Condition(s)	The Client Application is not logged into the Card Application

B.4.2.2 Attempt Logon with a malformed authenticator.

Purpose	Ensure a Client Application can detect and process a malformed authenticator byte sequence.
Target	pivLogIntoCardApplication
Reference(s)	1. SP800-73, Section 6.2.2 2. AS04.05
Precondition(s)	1. The card has established a connection to the client. 2. The cardHandle was properly initialized by pivConnect. 3. The client application has successfully executed the pivSelectCardApplication command.
Test Steps	1. Set cardHandle := <<a valid cardHandle>> 2. Set authenticators := <<a malformed authenticators byte sequence>> 3. Call pivLogIntoCardApplication w/ • *(IN)* authenticators • *(IN)* cardHandle
Expected Result(s)	Call returns with status_word := PIV_AUTHENTICATOR_MALFORMED
Post Condition(s)	The Client Application is not logged into the Card Application.

B.4.2.3 Attempt Logon with invalid authenticator

Purpose	Ensure a Client Application can detect and process an authenticator which has the correct format but does not result in a valid security permission/context.
Target	pivLogIntoCardApplication
Reference(s)	1. SP800-73, Section 6.2.2 2. AS04.05
Precondition(s)	1. The card has established a connection to the client. 2. The cardHandle was properly initialized by pivConnect. 3. The client application has successfully executed the pivSelectCardApplication command.
Test Steps	1. Set cardHandle := <<a valid cardHandle>> 2. Set authenticators := <<a well formed authenticators byte sequence containing an invalid PIN and/or KEY_REFERENCE value>> 3. Call pivLogIntoCardApplication w/ • *(IN)* authenticators

	• *(IN)* cardHandle
Expected Result(s)	Call returns with status_word := PIV_AUTHENTICATION_FAILURE
Post Condition(s)	The Client Application has is not logged into the Card Application.

B.4.2.4 Attempt Logon after a failed pivSelectCardApplication

Purpose	Ensure a Client Application properly respond to a failed pivSelectCardApplication method call.
Target	pivLogIntoCardApplication
Reference(s)	1. SP800-73, Section 6.2.2 2. AS04.05
Precondition(s)	1. The card has established a connection to the client. 2. The cardHandle was properly initialized by pivConnect. 3. The client application has un successfully executed the pivSelectCardApplication command.
Test Steps	1. Set cardHandle := <<a valid cardHandle>> 2. Set authenticators := <<valid authenticators byte sequence>> 3. pivSelectCardApplication status_word != "PIV_OK" 4. Call pivLogIntoCardApplication w/ • *(IN)* authenticators • *(IN)* cardHandle
Expected Result(s)	Call returns with status_word := PIV_CARD_APPLICATION_NO_LONGER_AVAILABLE
Post Condition(s)	The Client Application is not logged into the Card Application.

B.5 pivLogoutOfCardApplication

B.5.1 Valid Test Assertions

B.5.1.1 Log out of the Card Application

Purpose	Reset security context of the card application.
Target	pivLogoutOfCardApplication
Reference(s)	1. SP800-73, Section 6.2.4 2. AS03.05, AS04.01, AS04.07
Precondition(s)	1. The client has established a connection to the card. 2. The client is logged into the card application. 3. The client has established an "application security status".
Test Steps	1. Set cardHandle := <<a valid cardHandle>> 2. Call pivLogoutOfCardApplication w/ *(IN)*cardHandle
Expected Result(s)	Call returns with status_word := PIV_OK and the Client Application is logged off of the Card Application
Post Condition(s)	1. The Client Application is logged off of the Card Application. All data except "free read" data cannot be read or processed. 2. The cardHandle remains valid.

	3. The connection remains open.

B.5.2 Test Assertions for Error Conditions

B.5.2.1 Attempt Log out with Invalid Cardhandle

Purpose	Ensure the method can detect and handle an invalid cardHandle.
Target	`pivLogoutOfCardApplication`
Reference(s)	1. SP800-73, Section 6.2.4 2. AS04.07
Precondition(s)	1. The client has established a connection to the card. 2. The client is logged into the card application. 3. The client has established an "application security status".
Test Steps	1. `Set cardHandle := <<an invalid cardHandle>>` 2. `Call pivLogoutOfCardApplication w/(IN)cardHandle`
Expected Result(s)	Call returns with status_word :=`PIV_INVALID_CARD_HANDLE`
Post Condition(s)	The precondition states remain unchanged.

B.5.2.2 Attempt Log out with no security context

Purpose	Ensure the method can detect and handle an invalid security context.
Target	`pivLogoutOfCardApplication`
Reference(s)	1. SP800-73, Section 6.2.4 2. AS04.07
Precondition(s)	1. The client has established a connection to the card. 2. The client has successfully executed the pivSelectCardApplication method. 3. The client was not able to successfully execute the pivLogIntoCardApplication command
Test Steps	1. `Set cardHandle := <<avalid cardHandle>>` 2. `Call pivLogoutOfCardApplication w/(IN)cardHandle`
Expected Result(s)	Call returns with status_word := `PIV_NO_CURRENT_CONTEXT`
Post Condition(s)	The precondition states remain unchanged.

B.6 pivGetData

B.6.1 Valid Test Assertions

B.6.1.1 Get a reference to data on a Client Application data object

Purpose	Ensure the Card Application can read dataset from an object on the Client Application.
Target	`pivGetData`
Reference(s)	1. SP800-73, Section 6.2.3 2. AS03.05, AS04.01, AS04.06
Precondition(s)	1. The Client Application currently owns a connection to the Card

	Application accessible through cardHandle. 2. The client is logged into the Card Application.
Test Steps	1. Set cardHandle := <<valid cardHandle>> 2. Set OID := <<valid OID>> 3. Create data reference 4. Call pivGetData w/ (test for each implemented data object) • *(IN)* cardHandle • *(IN)* OID • *(OUT)* data
Expected Result(s)	Call returns with status_word of PIV_OK and an initialized reference to data
Post Condition(s)	The Card Application has access to the entire dataset of the selected Client Application object.

B.6.2 Test Assertions for Error Conditions

B.6.2.1 Handle an invalid cardHandle

Purpose	Ensure the Client Application can recognize and handle an invalid cardHandle.
Target	pivGetData
Reference(s)	1. SP800-73, Section 6.2.3 2. AS04.06
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The Client Application is logged into the Card Application.
Test Steps	1. Set cardHandle := <<invalid cardHandle>> 2. Set OID := <<valid OID>> 3. Create data reference 4. Call pivGetData w/ • *(IN)* cardHandle • *(IN)* OID • *(OUT)* data
Expected Result(s)	Call returns with status_word := PIV_INVALID_CARD_HANDLE and does not initialize data reference
Post Condition(s)	The client application returns to the state it had before the call.

B.6.2.2 Handle an invalid Object Identifier

Purpose	Ensure the Client Application can recognize and handle an invalid OID.
Target	pivGetData
Reference(s)	1. SP800-73, Section 6.2.3 2. AS04.06
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle.

	2. The Client Application is logged into the Card Application.
Test Steps	1. `Set cardHandle := <<valid cardHandle>>` 2. `Set OID := <<invalid OID>>` (Improper syntax or not found in Table 6 of SP 800-73) 3. `Create data reference` 4. `Call pivGetData w/` • *(IN)* `cardHandle` • *(IN)* `OID` • *(OUT)* `data`
Expected Result(s)	Call returns with status_word := `PIV_INVALID_OID` and does not initialize data reference
Post Condition(s)	The client application returns to the state it had before the call.

B.6.2.3 The Client Application can handle missing data object

Purpose	Ensure the Client Application can recognize and handle a missing OID.
Target	`pivGetData`
Reference(s)	1. SP800-73, Section 6.2.3 2. AS04.06
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The client is logged into the Card Application.
Test Steps	1. `Set cardHandle := <<valid cardHandle>>` 2. `Set OID := <<valid OID>>` (Found in Table 6 of SP 800-73 but not implemented on the PIV Card application.) 3. `Create data reference` 4. `Call pivGetData w/` • *(IN)* `cardHandle` • *(IN)* `OID` • *(OUT)* `data`
Expected Result(s)	Call returns with status_word := `PIV_DATA_OBJECT_NOT_FOUND` and does not initialize data reference
Post Condition(s)	The client application returns to the state it had before the call.

B.6.2.4 Security Conditions are enforced for Secured Objects

Purpose	Ensure that Security Conditions are enforced for Retrieving Data from Secured Applications
Target	`pivGetData`
Reference(s)	1. SP800-73, Section 6.2.3 2. AS04.06
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The client has successfully selected the PIV Card Application

	3. The client is not logged into the Card Application.
Test Steps	1. Set cardHandle := <<valid cardHandle>> 2. Set OID := <<valid OID for each of the following objects; PIVAuthCert, Fingerprint1, Fingerprint2, Facial Image, Printed Information, Digital Signature Key, Key Management Key >> 3. Create data reference 4. Set authenticators := <<invalid authenticators byte sequence for PIN>> 5. Call pivGetData w/ (for all implemented objects that require PIN) • *(IN)* cardHandle • *(IN)* OID • *(OUT)* data
Expected Result(s)	Call returns with status_word := PIV_SECURITY_CONDITION_NOT_SATISFIED and does not initialize data reference
Post Condition(s)	The client application returns to the state it had before the call.

B.7 pivPutData

B.7.1 Valid Test Assertions

B.7.1.1 Write data to an object on the Client Application

Purpose	Ensure the Client Application can write the entire data content to an object on the Card Application.
Target	pivPutData
Reference(s)	1. SP800-73, Section 6.4.1 2. AS03.05, AS04.01, AS04.09
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The PIV Card Application has authenticated the PIV Card Application Administrator.
Test Steps	1. Set cardHandle := <<valid cardHandle>> 2. Set OID := <<valid OID>> 3. Set data := <<a correctly formatted byte sequence> 4. Call pivPutData w/ (for all implemented objects) • *(IN)* cardHandle • *(IN)* OID • *(IN)* data
Expected Result(s)	Call returns with status_word of PIV_OK
Post Condition(s)	Validate that the Card Application has written the entire dataset of the selected object on the Client Application by issuing pivGetData

B.7.2 Test Assertions for Error Conditions

B.7.2.1 Identify and handle an invalid card handle

Purpose	Ensure the Client Application can identify and respond to an invalid card handle.
Target	`pivPutData`
Reference(s)	1. SP800-73, Section 6.4.1 2. AS04.09
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The PIV Card Application has authenticated the PIV Card Application Administrator.
Test Steps	1. Set cardHandle := <<invalid cardHandle>> 2. Set OID := <<valid OID>> 3. Set data := <<a correctly formatted byte sequence> 4. Call pivPutData w/ • *(IN)* cardHandle • *(IN)* OID • *(IN)* Data
Expected Result(s)	Call returns with status_word of `PIV_INVALID_CARD_HANDLE`
Post Condition(s)	1. The Card Application returns to the state it had prior to the pivPutData method call. 2. The precondition states remain unchanged.

B.7.2.2 Identify and handle an invalid Object Identifier (OID)

Purpose	Ensure the Client Application can identify and handle an invalid OID.
Target	`pivPutData`
Reference(s)	1. SP800-73, Section 6.4.1 2. AS04.09
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The PIV Card has authenticated the PIV Card Application Administrator.
Test Steps	1. Set cardHandle := <<valid cardHandle>> 2. Set OID := <<invalid OID>> (Improper syntax or not found in Table 6 of SP 800-73) 3. Set data := <<a correctly formatted byte sequence> 4. Call pivPutData w/ • *(IN)* cardHandle • *(IN)* OID • *(IN)* Data
Expected Result(s)	Call returns with status_word of `PIV_INVALID_OID`. Note that the PIV middleware will detect the invalid OID and return error without

	interrogating the card.
Post Condition(s)	1. The Card Application returns to the state it had prior to the pivPutData method call. 2. The precondition states remain unchanged.

B.7.2.3 Client Application can handle missing data object

Purpose	Ensure the Client Application can identify and handle an invalid data (parameter) byte sequence format.
Target	`pivPutData`
Reference(s)	1. SP800-73, Section 6.4.1 2. AS04.09
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. 2. The PIV Card Application has authenticated the PIV Card Application Administrator.
Test Steps	1. `Set cardHandle := <<valid cardHandle>>` 2. `Set OID := <<valid OID>>` (Found in Table 6 of SP 800-73 but not implemented on the PIV card application.) 3. `Set data := <<length exceeds available memory on the card>` 4. `Call pivPutData w/` • *(IN)* `cardHandle` • *(IN)* `OID` • *(IN)* `data`
Expected Result(s)	Call returns with status_word of `PIV_DATA_OBJECT_NOT_FOUND` based on the response received from the card
Post Condition(s)	1. The Card Application returns to the state it had prior to the pivPutData method call. 2. The precondition states remain unchanged.

B.7.2.4 Identify and handle an excessively large data parameter

Purpose	Ensure that the Client Application can identify and handle a byte sequence that is larger than the data parameter's definition.
Target	`pivPutData`
Reference(s)	1. SP800-73, Section 6.4.1 2. AS04.09
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The PIV Card Application has authenticated the PIV Card Application Administrator.
Test Steps	1. `Set cardHandle := <<valid cardHandle>>` 2. `Set OID := <<valid OID>>` 3. `Set data := <<a byte sequence with size greater than the data parameter's definition>`

	4. Call pivPutData w/ • *(IN)* cardHandle • *(IN)* OID • *(IN)* data
Expected Result(s)	Call returns with status_word of PIV_OFFSET_BEYOND_END_OF_DATA_CONTENT
Post Condition(s)	1. The Card Application returns to the state it had prior to the pivPutData method call. 2. The precondition states remain unchanged.

B.7.2.5 Security Conditions are enforced for Secured Objects

Purpose	Ensure that Security Conditions are enforced for Writing Data to Secured Objects
Target	pivPutData
Reference(s)	1. SP800-73, Section 6.4.1 2. AS04.09
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The Client Application has successfully selected the PIV Card Application. 3. The PIV Card Application has not authenticated the PIV Card Application Administrator.
Test Steps	1. Set cardHandle := <<valid cardHandle>> 2. Set OID := <<valid OID>> 3. Create data reference 4. Call pivPutData w/ (for all implemented objects) • *(IN)* cardHandle • *(IN)* OID • *(IN)* data
Expected Result(s)	Call returns with status_word := PIV_SECURITY_CONDITION_NOT_SATISFIED and does not initialize data reference
Post Condition(s)	The client application returns to the state it had before the call.

B.8 pivGenerateKeyPair

B.8.1 Valid Test Assertions

B.8.1.1 Generate an asymmetric key pair

Purpose	Ensure the Card Application can generate an asymmetric key pair.
Target	pivGenerateKeyPair
Reference(s)	1. SP800-73, Section 6.4.2 2. AS03.05, AS04.01, AS04.10

Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The Client Application has successfully selected the PIV Card Application. 3. The PIV Card Application has authenticated the PIV Card Application Administrator.
Test Steps	1. Set cardHandle := <<valid cardHandle>> 2. Set keyReference := <<an existing key reference suitable for use with the specified cryptographicMechanism >> 3. Set cryptographicMechanism := <<a recognized Cryptographic Mechanism Identifier>> 4. Create publicKey reference 5. Call pivGenerateKeyPair w/ • *(IN)* cardHandle • *(IN)* keyReference • *(IN)* cryptographicMechanism • *(OUT)* publicKey
Expected Result(s)	Call returns with status_word of PIV_OK and a reference to publicKey
Post Condition(s)	The Client Application creates a reference to a public key / private key pair which is accessible to the Card Application.

B.8.2 Test Assertions for Error Conditions

B.8.2.1 Identify and handle an invalid card handle

Purpose	Ensure the Card Application can catch invalid card handles.
Target	pivGenerateKeyPair
Reference(s)	1. SP800-73, Section 6.4.2 2. AS04.10
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The Client Application has successfully selected the PIV Card Application. 3. The PIV Card Application has authenticated the PIV Card Application Administrator.
Test Steps	1. Set cardHandle := <<invalid cardHandle>> 2. Set keyReference := <<an existing key reference suitable for use with the specified cryptographicMechanism >> 3. Set cryptographicMechanism := <<a recognized Cryptographic Mechanism Identifier>> 4. Create publicKey reference 5. Call pivGenerateKeyPair w/ • *(IN)* cardHandle • *(IN)* keyReference • *(IN)* cryptographicMechanism • *(OUT)* publicKey
Expected Result(s)	Call returns with status_word of PIV_INVALID_CARD_HANDLE

Post Condition(s)	1. The Card Application returns to the state it had prior to the pivGenerateKeyPair method call. 2. The precondition states are unaffected.

B.8.2.2 Identify and handle an invalid keyReference

Purpose	Ensure that the Card Application can identify an invalid keyReference.
Target	`pivGenerateKeyPair`
Reference(s)	1. SP800-73, Section 6.4.2 2. AS04.10
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The Client Application has successfully selected the PIV Card Application. 3. The PIV Card Application has authenticated the PIV Card Application Administrator.
Test Steps	1. Set cardHandle := <<valid cardHandle>> 2. Set keyReference := <<a key reference not found in the standard>> 3. Set cryptographicMechanism := <<a recognized Cryptographic Mechanism Identifier>> 4. Create publicKey reference 5. Call pivGenerateKeyPair w/ • *(IN)* cardHandle • *(IN)* keyReference • *(IN)* cryptographicMechanism • *(OUT)* publicKey
Expected Result(s)	Call returns with status_word of `PIV_INVALID_KEY_REFERENCE`
Post Condition(s)	1. The Card Application returns to the state it had prior to the pivGenerateKeyPair method call. 2. The precondition states are unaffected.

B.8.2.3 Identify and handle an invalid cryptographicMechanism

Purpose	Ensure that the Card Application can identify unsupported cryptographicMechanisms.
Target	`pivGenerateKeyPair`
Reference(s)	1. SP800-73, Section 6.4.2 2. AS04.10
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The Client Application has successfully selected the PIV Card Application. 3. The PIV Card Application has authenticated the PIV Card

	Application Administrator.
Test Steps	1. Set cardHandle := <<valid cardHandle>> 2. Set keyReference := <<a valid key reference>> 3. Set cryptographicMechanism := <<an unrecognized Cryptographic Mechanism Identifier>> 4. Create publicKey reference 5. Call pivGenerateKeyPair w/ • *(IN)* cardHandle • *(IN)* keyReference • *(IN)* cryptographicMechanism • *(OUT)* publicKey
Expected Result(s)	Call returns with status_word of PIV_UNSUPPORTED_CRYPTOGRAPHIC_MECHANISM
Post Condition(s)	1. The Card Application returns to the state it had prior to the pivGenerateKeyPair method call. 2. The precondition states are unaffected.

B.8.2.4 Identify and handle an invalid keyReference/cryptographicMechanism combinations

Purpose	Ensure that the Card Application can identify and handle invalid combinations of valid keyReferences and valid cryptographicMechanisms.
Target	pivGenerateKeyPair
Reference(s)	1. SP800-73, Section 6.4.2 2. AS04.10
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The Client Application has successfully selected the PIV Card Application. 3. The PIV Card Application has authenticated the PIV Card Application Administrator.
Test Steps	1. Set cardHandle := <<valid cardHandle>> 2. Set cryptographicMechanism := <<a recognized Cryptographic Mechanism Identifier>> 3. Set keyReference := <<a reference to a valid key that is not associated with the selected cryptographicMechanism >> 4. Create publicKey reference 5. Call pivGenerateKeyPair w/ • *(IN)* cardHandle • *(IN)* keyReference • *(IN)* cryptographicMechanism • *(OUT)* publicKey
Expected Result(s)	Call returns with status_word of PIV_GENERATED_KEY_PAIR_INCOMPATIBLE_WITH_EXISTING_KEY_PAIR
Post Condition(s)	1. The Card Application returns to the state it had prior to the pivGenerateKeyPair method call.

	2. The precondition states are unaffected.

B.9 pivCrypt

B.9.1 Valid Test Assertions

B.9.1.1 Encrypt a sequence of bytes

Purpose	Encrypt a sequence of bytes supplied by the Card Application.
Target	`pivCrypt`
Reference(s)	1. SP800-73, Section 6.3.1 2. AS03.05, AS04.01, AS04.08
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The client is logged into the Card Application.
Test Steps	1. Set cardHandle := <<valid cardHandle>> 2. Set keyReference := <<a recognized key reference>> 3. Set algorithmIdentifier := <<a recognized Algorithm Identifier>> 4. Set algorithmInput := <<byte sequence compatible with the chosen algorithm identifier AND keyReference>> 5. Create algorithmOutput reference 6. Call pivCrypt w/ • *(IN)* cardHandle • *(IN)* keyReference • *(IN)* algorithmIdentifier • *(IN)* algorithmInput • *(OUT)* algorithmOutput
Expected Result(s)	Call returns with status_word of `PIV_OK` and a reference to algorithmOutput
Post Condition(s)	In the client application the reference to the variable algorithmOutput points to a data object.

B.9.2 Test Assertions for Error Conditions

B.9.2.1 Identify and handle invalid card handles

Purpose	Ensure the Card Application can catch invalid card handles.
Target	`pivCrypt`
Reference(s)	1. SP800-73, Section 6.3.1 2. AS04.08
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The client is logged into the Card Application.
Test Steps	1. Set cardHandle := <<an invalid cardHandle>> 2. Set keyReference := <<a recognized key reference>> 3. Set algorithmIdentifier := <<a recognized Algorithm Identifier>> 4. Set algorithmInput := <<byte sequence compatible with the

	chosen algorithm identifier AND keyReference>> 5. Create algorithmOutput reference 6. Call pivCrypt w/ • *(IN)* cardHandle • *(IN)* keyReference • *(IN)* algorithmIdentifier • *(IN)* algorithmInput • *(OUT)* algorithmOutput
Expected Result(s)	Call returns with status_word of PIV_INVALID_CARD_HANDLE
Post Condition(s)	1. The Card Application returns to the state it had prior to the pivCrypt method call. 2. The precondition states are unaffected.

B.9.2.2 Identify and handle invalid key references

Purpose	Ensure the Client Application can identify and handle key references that are not compatible with the requested algorithm.
Target	pivCrypt
Reference(s)	1. SP800-73, Section 6.3.1 2. AS04.08
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The client is logged into the Card Application.
Test Steps	1. Set cardHandle := <<a valid cardHandle>> 2. Set keyReference := <<a key reference not found in the standard>> 3. Set algorithmIdentifier := <<a recognized Algorithm Identifier>> 4. Set algorithmInput := <<byte sequence compatible with the chosen algorithm identifier AND keyReference>> 5. Create algorithmOutput reference 6. Call pivCrypt w/ • *(IN)* cardHandle • *(IN)* keyReference • *(IN)* algorithmIdentifier • *(IN)* algorithmInput • *(OUT)* algorithmOutput
Expected Result(s)	Call returns with status_word of PIV_INVALID_KEY_REFERENCE
Post Condition(s)	1. The Card Application returns to the state it had prior to the pivCrypt method call. 2. The precondition states are unaffected.

B.9.2.3 Identify and handle invalid input data

Purpose	Ensure that the Client Application can identify and handle input data (algorithmInput) that is not compatible with the requested algorithm/key combination.

Target	pivCrypt
Reference(s)	1. SP800-73, Section 6.3.1 2. AS04.08
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The client is logged into the Card Application.
Test Steps	1. Set cardHandle := <<a valid cardHandle>> 2. Set keyReference := <<a key reference compatible with the algorithmIdentifier input value>> 3. Set algorithmIdentifier := <<a recognized Algorithm Identifier>> 4. Set algorithmInput := <<a byte sequence not compatible with the chosen algorithmIdentifier AND keyReference combination>> 5. Create algorithmOutput reference 6. Call pivCrypt w/ • *(IN)* cardHandle • *(IN)* keyReference • *(IN)* algorithmIdentifier • *(IN)* algorithmInput • *(OUT)* algorithmOutput
Expected Result(s)	Call returns with status_word of PIV_INPUT_BYTES_MALFORMED
Post Condition(s)	1. The Card Application returns to the state it had prior to the pivCrypt method call. 2. The precondition states are unaffected.

B.9.2.4 Identify and handle invalid keyReference/algorithmIdentifier combinations

Purpose	Ensure that the Card Application can identify and handle invalid combinations of valid keyReferences an valid algorithmIdentifiers.
Target	pivCrypt
Reference(s)	1. SP800-73, Section 6.3.1 2. AS04.08
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The client is logged into the Card Application.
Test Steps	1. Set cardHandle := <<valid cardHandle>> 2. Set keyReference := <<a valid key reference>> 3. Set algorithmIdentifier := <<a recognized Algorithm Identifier that is not associated with the keyReference>> 4. Create publicKey reference 5. Call pivCrypt w/ • *(IN)* cardHandle • *(IN)* keyReference • *(IN)* algorithmIdentifier • *(IN)* algorithmInput • *(OUT)* algorithmOutput
Expected Result(s)	Call returns with status_word of PIV_OTHER_CARD_ERROR. This error

	code could also result due to other error conditions.
Post Condition(s)	1. The Card Application returns to the state it had prior to the pivCrypt method call. 2. The precondition states are unaffected.

B.9.2.5 Identify and handle pivGenerateKeyPair method errors

Purpose	Ensure that the Card Application can properly respond to a failure of the pivGenerateKeyPair method.
Target	`pivCrypt`
Reference(s)	1. SP800-73, Section 6.3.1 2. AS04.08
Precondition(s)	1. The Client Application currently owns a connection to the Card Application accessible through cardHandle. 2. The client is logged into the Card Application. 3. pivGenerateKeyPair fails to return a valid publicKey reference
Test Steps	1. Set cardHandle := <<valid cardHandle>> 2. Set keyReference := <<a valid key reference>> 3. Set algorithmIdentifier := <<a recognized Algorithm Identifier>> 4. Create invalid publicKey reference 5. Call pivCrypt w/ • *(IN)* cardHandle • *(IN)* keyReference • *(IN)* algorithmIdentifier • *(IN)* algorithmInput • *(OUT)* algorithmOutput
Expected Result(s)	Call returns with status_word of `PIV_REFERENCE_DATA_NOT_FOUND`. (due to the key generation algorithm combination not implemented on the card)
Post Condition(s)	1. The Card Application returns to the state it had prior to the pivCrypt method call. 2. The precondition states are unaffected.

Appendix C—Card Command Interface Test Assertions

Test Assertion Template

Purpose	A quick description of the test and why it is being run
Reference(s)	1. References to the SP800-73 or other relevant publications 2. References to DTRs
Precondition(s)	Anything that must be done or known prior to executing the scenario
Test Scenario	`Sequence of APDU calls`
Expected Result(s)	What the expected execution path yields in terms of progress and values
Post Condition(s)	A description of the card application state once the test scenario completes

C.1 Card Commands for Data Access

C.1.1 SELECT Card Command

C.1.1.1 Contact Interface

Purpose	Validates that the PIV Card executes the SELECT card command through the contact interface for the following conditions: 1. Long AID 2. Right-truncated short AID
Reference(s)	1. SP800-73, Section 7.1.1, Section 5.2 2. AS01.04, AS01.05, AS01.06, AS01.07, AS03.03, AS05.01, AS05.02, AS05.03, AS05.05, AS05.06, AS05.07, AS05.08, AS05.09
Precondition(s)	1. A valid PIV Card is inserted into the contact reader 2. There exists a valid PC/SC connection between the test system and an instance of the contact reader 3. No application is currently connected to the PIV Card Application

Test Scenario	`1. Send SELECT card command with,`
	`• AID == A0 00 00 03 08 00 00 10 00 01 00`
	
	`2. Send SELECT card command without the version number,`
	`• AID == A0 00 00 03 08 00 00 10 00`

Expected Result(s)	1. The command returns the Application Property Template (APT) with the status words “90 00” at the end. The application property template conforms to Table 8 of SP800-73 2. The command returns the Application Property Template (APT) with the status words “90 00” at the end. The application property template conforms to Table 8 of SP800-73

Post Condition(s)	PIV Card Application is now the Currently Selected Application.

C.1.1.2 Error Condition

Purpose	Validates that the PIV Card Application is not deselected, while the currently selected application is the PIV Card Application, if the SELECT command is sent with an AID that is neither the AID of the PIV Card Application nor its right-truncated version
Reference(s)	1. SP800-73, Section 7.1.1 2. AS05.10
Precondition(s)	1. A valid PIV Card is inserted into the contact reader 2. There exists a valid PC/SC connection between the test system and an instance of the contact reader 3. No application is currently connected to the PIV Card Application
Test Scenario	1. Send SELECT card command with, • AID == A0 00 00 03 08 00 00 10 00 01 00 2. Repeat step 1 with, • AID == A0 00 00 03 08 00 00 00 00 (invalid AID) 3. Send GET DATA card command with, • Data field of the command containing the tag of the Card Capability Container data object (ALWAYS READ)
Expected Result(s)	1. The command returns the Application Property Template (APT) with the status words "90 00" at the end 2. The command returns '6A 82', application not found 3. The command returns the Card Capability Container (CCC) object with the status words "90 00" at the end
Post Condition(s)	The PIV Card Application continues to be the Currently Selected Application

C.1.1.3 Contactless Interface

Purpose	Validates conformance of the SELECT card command through the contactless interface
Reference(s)	1. SP800-73, Section 7.1.1 2. AS05.01
Precondition(s)	1. A valid PIV Card is placed within the reading range of the contactless reader 2. There exists a valid PC/SC connection between the test system and an instance of the contactless reader 3. No other contactless card is within the proximity of the reader
Test Scenario	1. Send SELECT card command with, • AID == A0 00 00 03 08 00 00 10 00 01 00

	2. Send SELECT card command without the version number, • AID == A0 00 00 03 08 00 00 10 00
Expected Result(s)	1. The command returns the Application Property Template (APT) with the status words "90 00" at the end. The application property template conforms to Table 8 of SP800-73 2. The command returns the Application Property Template (APT) with the status words "90 00" at the end. The application property template conforms to Table 8 of SP800-73
Post Condition(s)	PIV Card Application is the Currently Selected Application

C.1.2 GET DATA card command

C.1.2.1 Contact Interface

Purpose	Validates that the PIV Card accepts the GET DATA command through the contact interface and with the access rule of each container as specified in Table 1 of SP800-73. This test is applicable to the mandatory data objects required by SP800-73, and the optional data objects when supported by the card.
Reference(s)	1. SP800-73, Section 7.1.2 2. AS05.01, AS05.11
Precondition(s)	1. A valid PIV Card is inserted into the contact reader 2. There exists a valid PC/SC connection between the test system and an instance of the contact reader 3. No application is currently connected to the PIV Card Application 4. The optional containers supported by the card recorded
Test Scenario	1. Send SELECT card command with, • AID == A0 00 00 03 08 00 00 10 00 01 00 2. Send GET DATA command with, • Data field of the command containing the tag of the Card Capability Container data object • Le is set to a value less than the maximum length of the CCC 3. Send GET DATA command with, • Data field of the command containing the tag of the CHUID data object 4. Send GET DATA command with, • Data field of the command containing the tag of the X.509 Certificate for PIV Authentication object 5. Send GET DATA command with, • Data field of the command containing the tag of the

	Card Holder Fingerprint I data object 6. Send GET DATA command with, • Data field of the command containing the tag of the Card Holder Fingerprint II data object 7. Send GET DATA command with, • Data field of the command containing the tag of the Printed Information data object if supported by the card 8. Send GET DATA command with, • Data field of the command containing the tag of the Card Holder Facial Image data object if supported by the card 9. Send GET DATA command with, • Data field of the command containing the tag of the X.509 Certificate for Digital Signature Key data object if supported by the card 10. Send GET DATA command with, • Data field of the command containing the tag of the X.509 Certificate for Key Management Key data object if supported by the card 11. Send GET DATA command with, • Data field of the command containing the tag of the X.509 Certificate for Card Authentication Key data object if supported by the card 12. Send GET DATA command with, • Data field of the command containing the tag of the Security Object 13. Send VERIFY card command with, • P2, key reference value is set to ‘80’ • Data field of the command will contain the PIN value obtained from the vendor, and padded with ‘FF’ to complete the total length of the field to 8 bytes 14. Send GET DATA command with, • Data field of the command containing the tag of the X.509 Certificate for PIV Authentication object 15. Send GET DATA command with, • Data field of the command containing the tag of the Card Holder Fingerprint I data object 16. Send GET DATA command with, • Data field of the command containing the tag of the Card Holder Fingerprint II data object 17. Send GET DATA command with,

	• Data field of the command containing the tag of the Printed Information data object if supported by the card 18. Send GET DATA command with, • Data field of the command containing the tag of the Card Holder Facial Image data object if supported by the card 19. Send GET DATA command with, • Data field of the command containing the tag of the X.509 Certificate for Digital Signature data object if supported by the card 20. Send GET DATA command with, • Data field of the command containing the tag of the X.509 Certificate for Key Management data object if supported by the card 21. Send GET DATA command with, • Data filed of the command containing a tag that does not identify any of the data objects resident on the card
Expected Result(s)	1. The command returns the Application Property Template with the status words "90 00" at the end 2. The command returns only a part of the Card Capability Container data object whose length is specified in Le in the command along with the status words "61 xx" at the end. (xx indicates the number of response data bytes still available) 3. The command returns the CHUID data object along with the status words "90 00" at the end 4. For Steps 4 through 10, the command returns "69 82", security status not satisfied due to lack of PIN entry. 5. In step 11, the command returns the X.509 Certificate for the Card Authentication Key with the status words "90 00" at the end 6. In step 12, the command returns the Security Object with the status words "90 00" at the end. 7. In step 13, the command returns the status words "90 00" 8. For steps 14 through 20, the command returns the requested data object along with the status words "90 00" at the end 9. In step 21, the command returns '6A 82', data object not found
Post Condition(s)	NA

C.1.2.2 Contactless Interface

Purpose	Validates the conformance of the GET DATA command through the contactless interface. This test is applicable to the mandatory data objects required by SP800-73, and the optional data objects when

	supported by the card.
Reference(s)	1. SP800-73, Section 6.1.1 2. AS05.01
Precondition(s)	1. A valid PIV Card is placed within the reading range of the contactless reader 2. There exists a valid PC/SC connection between the test system and an instance of the contactless reader 3. No other contactless card is within the proximity of the reader
Test Scenario	`1. Repeat the steps 1-13 from the Test C.1.2.1`
Expected Result(s)	1. The command returns the Application Property Template with the status words "90 00" at the end 2. The command returns the status words "69 82", security status is not satisfied due to the contactless interface 3. The command returns the CHUID data object along with the status words "90 00" at the end 4. For Steps 4 through 10, the command returns "69 82", security status is not satisfied due to the contactless interface. 5. In step 11, the command returns the Card Authentication Key Object with the status words "90 00" at the end. 6. In step 12, the command returns the status words "69 82", security status is not satisfied due to the contactless interface 7. In steps 13, the command returns "6A 81", Function not supported
Post Condition(s)	NA

C.2 Commands For Authentication

C.2.1 VERIFY Card Command

C.2.1.1 Contact Interface

Purpose	Validates the following conditions associated with the VERIFY command: 1. Successful execution of the command 2. Execution of the command with a PIN not formatted per SP800-73 3. Multiple execution of the command with an incorrect PIN (formatted correctly) until the retry counter reaches zero
Reference(s)	1. SP800-73, Section 6.1.2 2. AS01.17, AS01.18, AS05.01, AS05.12, AS05.13, AS05.14, AS05.15
Precondition(s)	1. There exists a valid physical connection between an instance of the PIV Card and the host of the calling application 2. PIV Application PIN reset retry counter value (maximum number of PIN tries allowed) is recorded

Test Scenario	1. Send SELECT card command with, • AID == A0 00 00 03 08 00 00 10 00 01 00 2. Send VERIFY card command with, • P2, key reference value is set to a value other than what is supported by the card • Data field of the command will contain the correct cardholder PIN value, and padded with 'FF' to complete the total length of the field to 8 bytes 3. Send VERIFY card command with, • P2, key reference value is set to '80' • Data field of the command will contain the correct cardholder PIN value, and padded with 'FF' to complete the total length of the field to 8 bytes 4. Send VERIFY card command with, • P2, key reference value is set to '80' • Data field of the command will contain an arbitrary PIN value other than what is obtained from the vendor, NOT padded with 'FF' to complete the total length of the field to 8 bytes 5. Send VERIFY card command repeatedly, until after the issuer specified maximum number of PIN tries are exceeded with, • P2, key reference value is set to '80' • Data field of the command will contain an arbitrary PIN value other than what is obtained from the vendor, and padded with 'FF' to complete the total length of the field to 8 bytes
Expected Result(s)	1. The command returns the Application Property Template with the status words "90 00" at the end 2. The command returns '6A 88' 3. The command returns '90 00' 4. The command returns '6A 80' 5. The command returns • '63 CX' until the maximum number of PIN tries are reached (X indicates the number of further allowed retries) • '69 83' when the maximum number of PIN tries are exceeded
Post Condition(s)	The card is blocked

C.2.1.2 Contactless Interface

Purpose	Validates the conformance of the VERIFY command with SP800-73
Reference(s)	1. SP800-73, Section 6.1.2 2. AS05.03
Precondition(s)	1. A valid PIV Card is placed within the reading range of the contactless reader 2. There exists a valid PC/SC connection between the test system

	and an instance of the contactless reader 3. No other contactless card is within the proximity of the reader 4. The PIV Application is the currently selected application
Test Scenario	1. Send VERIFY card command with, • P2, key reference value is set to either '80' • Data field of the command will contain the correct cardholder PIN value, and padded with 'FF' to complete the total length of the field to 8 bytes
Expected Result(s)	1. The command returns '6A 81' (Function not supported)
Post Condition(s)	NA

C.2.2 CHANGE REFERENCE DATA card command

C.2.2.1 Contact Interface

Purpose	Validates that the PIV Card executes the CHANGE REFERENCE DATA command for the following conditions: 1. Without the proper security condition (PIN) 2. After the security condition is satisfied 3. With an incorrect PIN until the retry counter reaches zero
Reference(s)	1. SP800-73, Section 6.1.2, 2. AS05.01, AS05.16, AS05.17, AS05.18, AS05.19
Precondition(s)	1. PIV Application PIN reset retry counter value (maximum number of PIN tries allowed) is recorded 2. Card holder PIV Card Application PIN is recorded 3. A valid PIV Card is inserted into the contact reader 4. There exists a valid PC/SC connection between the test system and an instance of the contact reader 5. No application is currently connected to the PIV Card Application
Test Scenario	1. Send SELECT card command with, • AID == A0 00 00 03 08 00 00 10 00 01 00 2. Send CHANGE REFERENCE DATA card command with, • P2, key reference value is set to '80' • Data field of the command will contain the correct PIN value(PIN 1)obtained from the vendor, concatenated without delimitation with an arbitrary new PIN value(PIN 2). Both PINs should be padded with 'FF' to complete the total length of the field to 8 bytes 3. Send CHANGE REFERENCE DATA card command with, • P2, key reference value, is set to a value other than what is supported by the card • Data field of the command will contain the correct PIN value(PIN 2)concatenated without delimitation with an arbitrary new PIN value(PIN 3). Both PINs should be padded with 'FF' to complete the total length of the

	field to 8 bytes 4. Send CHANGE REFERENCE DATA card command with, • P2, key reference value is set to '80' • Data field of the command will contain the correct PIN value (PIN 2)concatenated without delimitation with an arbitrary new PIN value(PIN 3). At least one of the PINs should be less than 8 bytes 5. Send CHANGE REFERENCE DATA card command with, • P2, key reference value is set to '80' • Data field of the command will contain an incorrect PIN value (anything other than PIN 2), concatenated without delimitation with an arbitrary new PIN value(PIN 3)until after the issuer specified maximum number of tries are exceeded. Both PINs should be padded with 'FF' to complete the total length of the field to 8 bytes.
Expected Result(s)	1. Command returns the APT with the status words '90 00' at the end 2. Command returns the status words '90 00'. Validate that the existing PIN is changed to PIN 2 3. Command returns '6A 88' 4. Command returns '6A 80'. Validate that the existing PIN is still PIN 2 5. The command returns • '63 CX' until the maximum number of tries are reached. (X indicates the number of further allowed retries) • '69 83' when the maximum number of tries are exceeded
Post Condition(s)	The card is blocked.

C.2.2.2 Contactless Interface

Purpose	Validates that the PIV Card does not accept the CHANGE REFERENCE DATA command through the contactless interface
Reference(s)	1. SP800-73, Section 6.1.2 2. AS05.03
Precondition(s)	1. A valid PIV Card is placed within the reading range of the contactless reader 2. There exists a valid PC/SC connection between the test system and an instance of the contactless reader 3. No other contactless card is within the proximity of the reader
Test Scenario	1. Send SELECT card command with, • AID == A0 00 00 03 08 00 00 10 00 01 00 2. Send CHANGE REFERENCE DATA card command with, • P2, key reference value is set to '80' • Data field of the command will contain the correct PIN

	value(PIN 1)obtained from the vendor, concatenated without delimitation with an arbitrary new PIN value(PIN 2). Both PINs should be padded with 'FF' to complete the total length of the field to 8 bytes
Expected Result(s)	1. Command returns the APT with the status words '90 00' at the end 2. Command returns the status words '6A 81'
Post Condition(s)	PIN remains unchanged

C.2.3 RESET RETRY COUNTER command

C.2.3.1 Contact Interface

Purpose	Validates that the PIV Card executes the RESET RETRY COUNTER command for the following conditions: 1. Without the security condition satisfied 2. After the security condition (authenticating with the PIV Card Application Administrator) is satisfied 3. With the correct PIN not formatted per SP800-73 4. With an incorrect PIN (formatted correctly) until the retry counter reaches zero
Reference(s)	1. SP800-73, Section 7.2.3 2. AS05.01, AS05.20, AS05.21, AS05.22, AS05.23, AS05.24
Precondition(s)	1. PIV Application PIN reset retry counter value (maximum number of PIN tries allowed) is recorded 2. There exists a valid physical connection between an instance of the PIV Card and the host of the calling application and PIV card application is the currently selected application. 3. The initial value of the retry counter associated with the PIV Card application PIN is as stated by the vendor/issuer 4. The value of the counter reference data (PUK) is as stated by the vendor/issuer
Test Scenario	1. Send SELECT card command with, • AID == A0 00 00 03 08 00 00 10 00 01 00 2. Send RESET RETRY COUNTER with, • P2, key reference value, is set to a value other than what is supported by the card • Data field of the command contains the correct PUK concatenated without delimitation with the new PIN and each padded with 'FF' to complete the total length of the field to 8 bytes 3. Repeat Step 2 with, • P2, key reference value is set to '80' • Data field of the command contains the PUK value concatenated with the new PIN value , and at least one of the two values is not padded to complete 8 bytes

	4. Repeat step 2 with: • P2, key reference value is set to '80' • Data field of the command contains the correct PUK concatenated without delimitation with the new PIN and each padded with 'FF' to complete the total length of the field to 8 bytes 5. Repeat step 2 with • P2, key reference value is set to '80' • Data field of the command containing an incorrect PUK value concatenated with a new PIN value. This operation is repeated until the number of resets allowed is exceeded.
Expected Result(s)	1. Command returns the APT with the status words '90 00' at the end 2. The command returns '6A 88' (key reference not found) 3. The command returns ' '6A 80' (incorrect parameter in command data field) 4. The command returns '90 00'. The PIN value is changed to the new PIN (say PIN 2) and its retry counter is set to reset retry counter value 5. The command returns • '63 CX' , (X==number of resets left) • '69 83' – when the command is invoked after the value of X becomes zero. NOTE: Testing this condition may leave the card unusable in some implementations for all operations related to the key reference associated with this reset counter.
Post Condition(s)	1. No further resets of reference data associated with key reference possible.

C.2.3.2 Contactless Interface

Purpose	Validates that the RESET RETRY COUNTER command cannot be issued through the contactless interface
Reference(s)	1. SP800-73, Table 15 2. AS05.03
Precondition(s)	1. A valid PIV Card is placed within the reading range of the contactless reader 2. There exists a valid PC/SC connection between the test system and an instance of the contactless reader 3. No other contactless card is within the proximity of the reader
Test Scenario	1. Repeat steps 1 and 4 of test C.2.3.1
Expected Result(s)	1. Command returns the APT with the status words '90 00' at the end 2. The command returns '6A 81' (Function not supported)

Post Condition(s)	1. Reference data associated with key reference is not changed. Retry counter value associated with the key reference is not reset. The Reset counter value is unchanged.

C.2.4 GENERAL AUTHENTICATE command

C.2.4.1 Contact Interface

Purpose	Validates the authentication of the following entities using the GENERAL AUTHENTICATE command: 1. PIV Card Application to Test Toolkit Application (INTERNAL AUTHENTICATE) 2. Two way authentication of PIV Card Application and Test Toolkit Application (MUTUAL AUTHENTICATE)
Reference(s)	1. SP800-73, Section 7.2.4 2. AS05.01, AS03.02, AS03.07, AS05.25, AS0526, AS05.27
Precondition(s)	1. A valid PIV Card is inserted into the contact reader 2. There exists a valid PC/SC connection between the test system and an instance of the contact reader 3. The PIV Card Application is the currently selected application on the card 4. The length of the challenge supported by the card is obtained 5. The PIN security condition is satisfied
Test Scenario	1. Send GENERAL AUTHENTICATE card command including a random challenge with, • CLA is set to 00 • P1, algorithm reference, is set to ‘06’ identifying the 1024 bits RSA algorithm identifier • P2, key reference, is set to ‘9A’ indicating the PIV Authentication Key • Data field in the command is to include ‘81’ specifying a challenge, followed by a randomly generated challenge NOTE: The following two test invocations (2 & 3) are to be performed only if the PIV Card Application supports the use of the key ‘9B’. 2. Send GENERAL AUTHENTICATE card command to request a witness from the PIV Card Application using the reference data associated with a specified key reference (parameter P2) and a specified algorithm (parameter P1) • CLA is set to 00 • P1, algorithm reference, is set to ‘00’ • P2, key reference, is set to ‘9B’ • Data field in the command is to include ‘80’ requesting a witness from the PIV Card application. 3. When the PIV Card application responds with the encryption of the nonce using the specified data and algorithm (in the previous command), send the GENERAL AUTHENTICATE card

	command with, • P1, algorithm reference is set to '00' • P2, key reference is set to '9B' • Data field in the command is to include '80' followed by decryption of the encrypted nonce sent by the card application and '81' followed by another nonce. 4. Repeat Step 1 with the Le field is set to a value smaller than what the card is prepared to return in length 5. Repeat Step 1 with P2, key reference is set to a value for a key that is not supported by the card 6. Repeat Step 1 with data field in the command including a data object tag that is not in Table 17 of SP800-73
Expected Result(s)	1. The command returns the encrypted challenge with '90 00' at the end. Decrypt the encrypted challenge and compare it to the one sent to the card 2. The PIV card application returns with the encryption of a nonce followed by '90 00' 3. The PIV card application verifies the witness and then responds with encryption of the nonce sent by test toolkit application followed by '90 00' 4. The command returns '61 xx' indicating xx number of response data bytes still available 5. The command returns '6A 86' 6. The command returns '6A 80'
Post Condition(s)	N/A

C.2.4.2 Contactless Interface

Purpose	Validates the mutual authentication between a PIV Card and terminal
Reference(s)	1. SP800-73, Section B.3 2. AS05.03
Precondition(s)	1. A valid PIV Card is placed within the reading range of the contactless reader 2. There exists a valid PC/SC connection between the test system and an instance of the contactless reader 3. No other contactless card is within the proximity of the reader
Test Scenario	NOTE: The following tests are to be performed only if the PIV Card Application supports the use of the key '9B'. 1. Repeat steps 2 and 3 of C.2.4.1
Expected Result(s)	1. The two test invocations referred to in Step 1 above should return the same responses as 2 and 3 of Expected Results under test C.2.4.1

Post Condition(s)	N/A

C.3 Card Commands for Credential Initialization and Administration

C.3.1 PUT DATA Command

C.3.1.1 Contact Interface

Purpose	Validates that the PUT DATA command exhibits the appropriate behavior under the following conditions: 1. Without the security condition is satisfied 2. After the security condition is satisfied
Reference(s)	1. SP800-73, Section 7.3.1 2. AS05.01, AS05.28
Precondition(s)	1. A valid PIV Card is inserted into the contact reader 2. There exists a valid PC/SC connection between the test system and an instance of the contact reader 3. The PIV Card Application is the currently selected application on the card 4. The length of the challenge supported by the card is obtained
Test Scenario	NOTE: The following tests (1 through 11) are run based on the assumption that the PIV Card Application either does not support the key referenced by '9B' or no mutual authentication of the PIV Card Application and the test toolkit application has been performed using steps 2 and 3 of C.2.4.1 (GENERAL AUTHENTICATE) 1. Send PUT DATA card command with, • CLA is set to 00 • Data field in the command is to include the tag of the Card Capability Container object • Data field in the command is to include the data that will replace the CCC 2. Repeat step 1 with • CLA is set to 00 • Data field in the command is to include the tag of the CHUID object • Data field in the command is to include the data content that will replace the CHUID 3. Repeat step 1 with • CLA is set to 00 • Data field in the command is to include the tag of the PIV Authentication Certificate object • Data field in the command is to include data content that will replace the PIV Authentication Certificate 4. Repeat step 1 with • CLA is set to 00

	• Data field in the command is to include the tag of the Card Holder Fingerprint I object • Data field in the command is to include data content that will replace the Card Holder Fingerprint I 5. Repeat step 1 with • CLA is set to 00 • Data field in the command is to include the tag of the Card Holder Fingerprint II object • Data field in the command is to include the data content that will replace the Card Holder Fingerprint II 6. Repeat step 1 with • CLA is set to 00 • Data field in the command is to include the tag of the Security Object • Data field in the command is to include the data content that will replace the Security Object 7. If the card supports the Printed Information, repeat step 1 with • Data field in the command is to include the tag of the Printed Information object • Data field in the command is to include the data content that will replace the Printed Information 8. If the card supports the Card Holder Facial Image, repeat step 1 with • Data field in the command is to include the tag of the Card Holder Facial Image object • Data field in the command is to include the data content that will replace the Card Holder Facial Image 9. If the card supports the certificate for Digital Signature, repeat step 1 with • Data field in the command is to include the tag of the certificate for Digital Signature object • Data field in the command is to include the data content that will replace the certificate for Digital Signature 10. If the card supports the certificate for Key Management, repeat step 1 with • Data field in the command is to include the tag of the certificate for Key Management object • Data field in the command is to include the data content that will replace the certificate for Key Management 11. If the card supports the certificate for Card Authentication, repeat step 1 with • Data field in the command is to include the tag of the certificate for Card Authentication object

	• Data field in the command is to include the data content that will replace the certificate for Card Authentication NOTE: The following tests are to be performed only if the PIV Card Application supports the use of the key '9B'. 12. Perform mutual authentication of PIV Card Application and the test toolkit application using steps 2 and 3 of C.2.4.1 (GENERAL AUTHENTICATE) 13. Repeat steps 1-11
Expected Result(s)	1. In Steps 1 through 11, commands return '69 82', (security status not satisfied) and the contents of the data objects remained unchanged 2. The two test invocations referred to in Step 12 should return the same responses as 2 and 3 of Expected Results under test C.2.4.1 3. In Step 13, commands return '90 00', and the contents of the data objects have been replaced
Post Condition(s)	1. Validate that the contents of each object has been overwritten with the new values provided in step 13.

C.3.1.2 Contactless Interface

Purpose	Validates that the PUT DATA command cannot be issued through the contactless interface
Reference(s)	1. SP800-73, Table 15 2. AS05.03
Precondition(s)	1. Record the existing values of all data objects 2. A valid PIV Card is placed within the reading range of the contactless reader 3. There exists a valid PC/SC connection between the test system and an instance of the contactless reader 4. No other contactless card is within the proximity of the reader
Test Scenario	1. Repeat steps 1 through 11 of 3.1.1
Expected Result(s)	1. All commands return '6A 81'
Post Condition(s)	1. The data container values remain unchanged

C.3.2 GENERATE ASYMMETRIC KEY PAIR command

C.3.2.1 Contact Interface

Purpose	Validates that the card executes the GENERATE ASYMMETRIC KEY PAIR command for the following conditions: 1. Without the security condition satisfied

	2. After the security condition (authenticating with the PIV Card Application Administrator) is satisfied
Reference(s)	1. SP800-73, Section 7.3.1 2. AS05.01, AS05.29, AS05.30, AS05.31
Precondition(s)	1. A valid PIV Card is inserted into the contact reader 2. There exists a valid PC/SC connection between the test system and an instance of the contact reader 3. The PIV Card Application is the currently selected application on the card 4. The length of the challenge supported by the card is obtained
Test Scenario	NOTE: The following test (# 1) is run based on the assumption that the PIV Card Application either does not support the key ‘9B’ or no mutual authentication of the PIV Card Application and the test toolkit application has been performed using steps 2 and 3 of C.2.4.1 (GENERAL AUTHENTICATE) 1. Send GENERATE ASYMMETRIC KEY PAIR card command with, • P2 is set to the Key Reference to be assigned to the generated key pair • Data field in the command is to include the ‘06’ as the Cryptographic Mechanism Identifier NOTE: The following tests are to be performed only if the PIV Card Application supports the use of the key ‘9B’. 2. Perform mutual authentication of PIV Card Application and the test toolkit application using steps 2 and 3 of C.2.4.1 (GENERAL AUTHENTICATE) 3. Repeat Step 1 4. Repeat Step 1 with Le, the length of public key of data object template, is set to a value smaller than the value the card supports 5. Repeat Step 1 with the “Cryptographic Mechanism Identifier” value in the data field is set to a value that is not supported by the card. 6. Repeat Step 1 with P2 is set to a key reference value whose associated cryptographic mechanism is different than the cryptographic mechanism in the data field of the command
Expected Result(s)	1. Command returns ’69 82’ 2. The two test invocations referred to in Step 2 should return the same responses as 2 and 3 of Expected Results under test C.2.4.1 3. Command returns the data objects of the public key of the generated key pair followed by ’90 00’. Validate that the public key data object is replaced in full by the generated new data 4. Command returns ’61 xx’ indicating xx number of response data

	bytes are still available 5. Command returns '6A 80' 6. Command returns '6A 86'
Post Condition(s)	1. The contents of the public key object are updated with the new values

C.3.2.2 Contactless Interface

Purpose	Validates that the GENERATE ASYMMETRIC KEY PAIR command cannot be issued through the contactless interface with or without authenticating with the PIV Card Application Administrator
Reference(s)	1. SP800-73, Table 15 2. AS05.03
Precondition(s)	1. Record the existing contents of the public key data object 2. A valid PIV Card is placed within the reading range of the contactless reader 3. There exists a valid PC/SC connection between the test system and an instance of the contactless reader 4. No other contactless card is within the proximity of the reader
Test Scenario	`1. Perform steps 1 of test C.3.2.1`
Expected Result(s)	1. Command returns '6A 81'
Post Condition(s)	1. Validate the contents of the public key object by GET DATA command to make sure the value remained unchanged and no new asymmetric key pair is generated

Appendix D—PIV Data Objects Representation Test Assertions

Test Plan Assumptions:

1.0	When the length of the value field is between 0 and 127 bytes, the length field consists of a single byte where bit 8 is set to 0 and bits 7 to 1 encode the number of bytes in the value field. When the length of the value field is greater than 127 bytes, the length field consists of two or more bytes. The first byte is '81', '82', '83' or '84' where the low order nibble of each of these possible first-byte values (1, 2, 3, or 4 respectively) encodes the number of subsequent and remaining bytes in the length field. These subsequent and remaining bytes are taken together in order to be a big-endian integer encoding the number of bytes in the value field. Table D-1 shows the encoding of the length field.
1.1	Each BER-TLV tag is encoded as three bytes.
1.2	Each data object returned is appended with a 2 byte status word.
1.3	All variable length value fields can have zero lengths, which will result in a tag length field being immediately followed by the next tag, if applicable.
1.4	The final byte of the command string can be set to 0x00 to retrieve an entire data object regardless of the size of that object.

Number of Bytes in the Length Field	First Byte	Subsequent Bytes	Length of the Value Field
1 byte	‘00’ to ‘7F’	None	0 to 127
2 byte	‘81’	'00' to 'FF'	0 to 255
3 byte	‘82’	'0000' to 'FFFF'	0 to 65,535
4 byte	‘83’	'000000' to 'FFFFFF'	0 to 16,777,215
5 byte	‘84’	'00000000' to 'FFFFFFFF'	0 to 4,294,967,295

Table D-1. Encoding of Length Field

D.1 “Card Capabilities Container” Data Object

Purpose	Confirms that the CCC of the PIV Card Application conforms to the PIV Data Model requirements as per Appendix A of SP800-73
Target	`Card Capabilities Container`
Reference(s)	1. SP800-73, Section 7.1.2
Precondition(s)	1. There exists valid physical connection between an instance of the PIV Card and the host of the calling application. 2. The PIV Application is the currently selected application.

	3. The required security condition is satisfied. 4. The actual content of the value fields in the CCC is obtained from the vendor
Test Scenario	`Call GET DATA w/ the following hexadecimal command` `00 CB 3F FF 05 5C 03 5F C1 07`

Expected Result(s)	1. Call returns
	2. Call returns a byte array with a maximum size of 297 bytes
	3. Call returns BER-TLV formatted byte array with the following byte sequence:
	`DataObject[0] == 0xF0`
	`DataObject[1] == 0x15`
	`DataObject[23] == 0xF1`
	`DataObject[24] == 0x01`
	`DataObject[26] == 0xF2`
	`DataObject[27] == 0x01`
	`DataObject[29] == 0xF3`
	
	`IF tag 0xF3 Length < 128`
	`DataObject[30] <= 0x7F`
	`Set byte i = DataObject[30]`
	`ELSE`
	`DataObject[30..31] == 0x10 00`
	`Set byte i = DataObject[30..31] + 1`
	
	`DataObject[30+i] == 0xF4`
	`DataObject[31+i] == 0x01`
	`DataObject[33+i] == 0xF5`
	`DataObject[34+i] == 0x01`
	`DataObject[36+i] == 0xF6`
	`DataObject[37+i] == 0x11`
	`DataObject[55+i] == 0xF7`
	`DataObject[56+i] == 0x00`
	`DataObject[57+i] == 0xFA`
	`DataObject[58+i] == 0x00`
	`DataObject[59+i] == 0xFB`
	`DataObject[60+i] == 0x00`
	`DataObject[61+i] == 0xFC`
	`DataObject[62+i] == 0x00`
	`DataObject[63+i] == 0xFD`
	`DataObject[64+i] == 0x00`
	`DataObject[65+i] == 0xE3`
	`DataObject[66+i] == 0x30`
	`DataObject[115+i] == 0xB4`
	`DataObject[116+i] == 0x30`
	`DataObject[165+i] == 0xFE`
	`DataObject[166+i] == 0x00`

	`DataObject[167+i..168+i] == 0x90 00`
Post Condition(s)	• Client Application has received the Card Capabilities Container data object with "Status Words" appended.

D.2 "Card Holder Unique Identifier" Data Object

Purpose	Confirms that the CHUID of the PIV Card Application conforms to the PIV Data Model requirements as per Appendix A of SP800-73
Target	`Card Holder Unique Identifier`
Reference(s)	1. SP800-73, Section 7.1.2
Precondition(s)	1. There exists valid physical connection between an instance of the PIV Card and the host of the calling application. 2. The PIV Application is the currently selected application. 3. The required security condition is satisfied. 4. The actual content of the value fields in the CHUID is obtained from the vendor.
Test Scenario	`1. Call GET DATA w/ the following command (in hexadecimal)` • `00 CB 3F FF 05 5C 5F C1 02 00`

Expected Result(s)	1. Call returns
	2. Call returns a byte array with a maximum size of 2394 bytes.
	3. Call returns BER-TLV formatted byte array with the following byte sequence:
	`DataObject[0] == 0x30`
	`DataObject[1] == 0x 19`
	
	`DataObject[26] == 0x34`
	`DataObject[27] == 0x 10`
	
	`DataObject[44] == 0x35`
	`DataObject[45] == 0x08`
	
	`DataObject[54] == 0x36 (if found in vendor documentation)`
	`DataObject[55] == 0x01`
	
	`DataObject[57] == 0x3D (if found in vendor documentation)`
	
	`IF tag 0x3D Length < 128`
	`DataObject[58] <= 0x7F`
	`Set byte i = DataObject[55]`

	`ELSE`
	`  DataObject[58..59] <= 0x02 00`
	`  Set byte i = DataObject[30..31] + 1`
	
	`DataObject[59+i] == 0x3E`
	
	`IF tag 0x3E Length < 128`
	`  DataObject[60+i] <= 0x7F`
	`  Set byte i = i + DataObject[57+i]`
	`ELSE`
	`  DataObject[60+i..61+i] <= 0x0B 00`
	`  Set byte i = i + DataObject[60+i..61+i] + 1`
	
	`DataObject[60+i] == 0xFE`
	`DataObject[61+i] == 0x00`
	`DataObject[62+i..63+i] == 0x90 00`

Post Condition(s)	• Client Application has received the requested data object with "Status Words" appended.

D.3 "X.509 Certificate for PIV Authentication" Data Object

Purpose	Confirms that the X.509 Certificate for PIV Authentication of the PIV Card Application conforms to the PIV Data Model requirements as per Appendix A of SP800-73
Target	`X.509 Certificate for PIV Authentication`
Reference(s)	1. SP800-73, Section 7.1.2
Precondition(s)	1. There exists valid physical connection between an instance of the PIV Card and the host of the calling application. 2. The PIV Application is the currently selected application. 3. The required security condition is satisfied. 4. The actual content of the value fields in the X.509 Certificate for PIV Authentication data object is obtained from the vendor
Test Scenario	`1. Call GET DATA w/ the following command (in hexadecimal)` • `00 CB 3F FF 05 5C 5F C1 07 00`

Expected Result(s)	1. Call returns
	2. Call returns a byte array with a maximum size of 1660 bytes
	3. Call returns BER-TLV formatted byte array with the following byte sequence:
	`DataObject[0]  == 0x70`
	
	`IF tag 0x70 Length < 128`
	`  DataObject[1] <= 0x7F`

	`Set byte i = DataObject[1]`
	`ELSE`
	`DataObject[1..2] <= 0x06 4C`
	`Set byte i = DataObject[1..2] + 1`
	
	`DataObject[1+i] == 0x71`
	`DataObject[2+i] == 0x01`
	
	`DataObject[4+i] == 0x72`
	`DataObject[5+i] <= 0x26`
	
	`Set byte i = i + DataObject[5+i]`
	
	`DataObject[6+i] == 0xFE`
	`DataObject[7+i] == 0x00`
	`DataObject[8+i..9+i] == 0x90 00`

Post Condition(s)	• Client Application has received the requested data object with "Status Words" appended.

D.4 "Card Holder Fingerprint I" Data Object

Purpose	Confirms that the "Card Holder Fingerprint I" data object of the PIV Card Application conforms to the PIV Data Model requirements as per Appendix A of SP800-73
Target	`Card Holder Fingerprint I`
Reference(s)	1. SP800-73, Section 7.1.2
Precondition(s)	1. There exists valid physical connection between an instance of the PIV Card and the host of the calling application. 2. The PIV Application is the currently selected application. 3. The required security condition is satisfied. 4. The actual content of the value fields in the "Card Holder Fingerprint I" data object is obtained from the vendor
Test Scenario	`1. Call GET DATA w/ the following command (in hexadecimal)` • `00 CB 3F FF 05 5C 5F C1 03 00`

Expected Result(s)	1. Call returns
	2. Call returns a byte array with a maximum size of 7773 bytes
	3. Call returns BER-TLV formatted byte array with the following byte sequence:
	`DataObject[0] == 0xBC`
	
	`IF tag 0xBC Length < 128`

	`DataObject[1] <= 0x7F`
	`Set byte i = DataObject[1]`
	`ELSE`
	`DataObject[1..2] <= 0x1E 58`
	`Set byte i = DataObject[1..2] + 1`
	
	`DataObject[1+i] == 0xFE`
	`DataObject[2+i] == 0x00`
	`DataObject[3+i..4+i] == 0x90 00`

Post Condition(s)	• Client Application has received the requested data object with "Status Words" appended.

D.5 "Card Holder Fingerprint II" Data Object

Purpose	Confirms that the "Card Holder Fingerprint II" data object of the PIV Card Application conforms to the PIV Data Model requirements as per Appendix A of SP800-73
Target	`Card Holder Fingerprint II`
Reference(s)	1. SP800-73, Section 7.1.2
Precondition(s)	1. There exists valid physical connection between an instance of the PIV Card and the host of the calling application. 2. The PIV Application is the currently selected application. 3. The required security condition is satisfied. 4. The actual content of the value fields in the "Card Holder Fingerprint II" data object is obtained from the vendor
Test Scenario	`1. Call GET DATA w/ the following command (in hexadecimal)` • `00 CB 3F FF 05 5C 5F C1 04 00`

Expected Result(s)	1. Call returns
	2. Call returns a byte array with a maximum size of 7773 bytes
	3. Call returns BER-TLV formatted byte array with the following byte sequence:
	`DataObject[0] == 0xBC`
	
	`IF tag 0xBC Length < 128`
	`DataObject[1] <= 0x7F`
	`Set byte i = DataObject[1]`
	`ELSE`
	`DataObject[1..2] <= 0x1E 58`
	`Set byte i = DataObject[1..2] + 1`
	
	`DataObject[1+i] == 0xFE`
	`DataObject[2+i] == 0x00`

	`DataObject[3+i..4+i] == 0x90 00`
Post Condition(s)	• Client Application has received the requested data object with "Status Words" appended.

D.6 "Printed Information" Data Object

Purpose	Confirms that the "Printed Information" Data Object of the PIV Card Application conforms to the PIV Data Model requirements as per Appendix A of SP800-73
Target	`Printed Information`
Reference(s)	1. SP800-73, Section 7.1.2
Precondition(s)	1. There exists valid physical connection between an instance of the PIV Card and the host of the calling application. 2. The PIV Application is the currently selected application. 3. The required security condition is satisfied. 4. The actual content of the value fields in the "Printed Information" Data Object is obtained from the vendor
Test Scenario	1. Call GET DATA w/ the following command (in hexadecimal) • `00 CB 3F FF 05 5C 5F C1 09 00`

Expected Result(s)	1. Call returns
	2. Call returns a byte array with a maximum size of 120 bytes
	3. Call returns BER-TLV formatted byte array with the following byte sequence:
	`DataObject[0] == 0x01`
	`DataObject[1] == 0x 20`
	
	`DataObject[34] == 0x02`
	`DataObject[35] == 0x 14`
	
	`DataObject[56] == 0x03`
	`DataObject[57] == 0x14`
	
	`DataObject[78] == 0x04`
	`DataObject[79] == 0x 09`
	
	`DataObject[89] == 0x05`
	`DataObject[90] == 0x 0A`
	
	`DataObject[101] == 0x06`
	`DataObject[102] == 0x 0F`
	
	`DataObject[118+i] == 0xFE`
	`DataObject[119+i] == 0x00`

	`DataObject[120+i..121+i] == 0x90 00`
Post Condition(s)	• Client Application has received the Card Capabilities Container data object with "Status Words" appended.

D.7 "Card Holder Facial Image" Data Object

Purpose	Confirms that the "Card Holder Facial Image" data object of the PIV Card Application conforms to the PIV Data Model requirements as per Appendix A of SP800-73
Target	`Card Holder Facial Image`
Reference(s)	1. SP800-73, Section 7.1.2
Precondition(s)	1. There exists valid physical connection between an instance of the PIV Card and the host of the calling application. 2. The PIV Application is the currently selected application. 3. The required security condition is satisfied. 4. The actual content of the value fields in the "Card Holder Facial Image" data object is obtained from the vendor
Test Scenario	`1. Call GET DATA w/ the following command (in hexadecimal)` • `00 CB 3F FF 05 5C 5F C1 08 00`

Expected Result(s)	1. Call returns
	2. Call returns a byte array with a maximum size of 12,709 bytes
	3. Call returns BER-TLV formatted byte array with the following byte sequence:
	`DataObject[0] == 0xBC`
	
	`IF tag 0xBC Length < 128`
	`DataObject[1] <= 0x7F`
	`Set byte i = DataObject[1]`
	`ELSE`
	`DataObject[1..2] <= 0x31 A0`
	`Set byte i = DataObject[1..2] + 1`
	
	`DataObject[1+i] == 0xFE`
	`DataObject[2+i] == 0x00`
	`DataObject[3+i..4+i] == 0x90 00`

Post Condition(s)	• Client Application has received the requested data object with "Status Words" appended.

D.8 "X.509 Certificate for Digital Signature" Data Object

Purpose	Confirms that the "X.509 Certificate for Digital Signature" data object of the PIV Card Application conforms to the PIV Data Model requirements as per Appendix A of SP800-73
Target	`X.509 Certificate for Digital Signature`
Reference(s)	1. SP800-73, Section 7.1.2
Precondition(s)	1. There exists valid physical connection between an instance of the PIV Card and the host of the calling application. 2. The PIV Application is the currently selected application. 3. The required security condition is satisfied. 4. The actual content of the value fields in the X.509 Certificate for Digital Signature data object is obtained from the vendor
Test Scenario	1. Call GET DATA w/ the following command (in hexadecimal) • `00 CB 3F FF 05 5C 5F C1 0A 00`

Expected Result(s)	1. Call returns
	2. Call returns a byte array with a maximum size of 1660 bytes
	3. Call returns BER-TLV formatted byte array with the following byte sequence:
	`DataObject[0] == 0x70`
	
	`IF tag 0x70 Length < 128`
	`DataObject[1] <= 0x7F`
	`Set byte i = DataObject[1]`
	`ELSE`
	`DataObject[1..2] <= 0x06 4C`
	`Set byte i = DataObject[1..2] + 1`
	
	`DataObject[1+i] == 0x71`
	`DataObject[2+i] == 0x01`
	
	`DataObject[4+i] == 0x72`
	`DataObject[5+i] <= 0x26`
	
	`Set byte i = i + DataObject[5+i]`
	
	`DataObject[6+i] == 0xFE`
	`DataObject[7+i] == 0x00`
	`DataObject[8+i..9+i] == 0x90 00`
	4. Attempt to export the Digital Signature Key is declined
	5. Attempt to perform cryptographic operations using the digital signature key through the contactless interface is declined

Post Condition(s)	• Client Application has received the requested data object with

	"Status Words" appended.

D.9 "X.509 Certificate for Key Management" Data Object

Purpose	Confirms that the "X.509 Certificate for Key Management" data object of the PIV Card Application conforms to the PIV Data Model requirements as per Appendix A of SP800-73
Target	`X.509 Certificate for Key Management`
Reference(s)	1. SP800-73, Section 7.1.2
Precondition(s)	1. There exists valid physical connection between an instance of the PIV Card and the host of the calling application. 2. The PIV Application is the currently selected application. 3. The required security condition is satisfied. 4. The actual content of the value fields in the X.509 Certificate for Key Management data object is obtained from the vendor
Test Scenario	1. Call GET DATA w/ the following command (in hexadecimal) • 00 CB 3F FF 05 5C 5F C1 0B 00

Expected Result(s)	1. Call returns
	2. Call returns a byte array with a maximum size of 1660 bytes
	3. Call returns BER-TLV formatted byte array with the following byte sequence:
	`DataObject[0] == 0x70`
	
	`IF tag 0x70 Length < 128`
	`DataObject[1] <= 0x7F`
	`Set byte i = DataObject[1]`
	`ELSE`
	`DataObject[1..2] <= 0x06 4C`
	`Set byte i = DataObject[1..2] + 1`
	
	`DataObject[1+i] == 0x71`
	`DataObject[2+i] == 0x01`
	
	`DataObject[4+i] == 0x72`
	`DataObject[5+i] <= 0x26`
	
	`Set byte i = i + DataObject[5+i]`
	
	`DataObject[6+i] == 0xFE`
	`DataObject[7+i] == 0x00`
	`DataObject[8+i..9+i] == 0x90 00`
	4. Attempt to access the key management key through the contactless interface is declined.

Post Condition(s)	• Client Application has received the requested data object with "Status Words" appended.

D.10 "X.509 Certificate for Card Authentication" Data Object

Purpose	Confirms that the "X.509 Certificate for Card Authentication" data object of the PIV Card Application conforms to the PIV Data Model requirements as per Appendix A of SP800-73
Target	`X.509 Certificate for Card Authentication`
Reference(s)	1. SP800-73, Section 7.1.2
Precondition(s)	1. There exists valid physical connection between an instance of the PIV Card and the host of the calling application. 2. The PIV Application is the currently selected application. 3. The required security condition is satisfied. 4. The actual content of the value fields in the X.509 Certificate for Card Authentication data object is obtained from the vendor
Test Scenario	1. Call GET DATA w/ the following command (in hexadecimal) • 00 CB 3F FF 05 5C 5F C1 01 00

Expected Result(s)	1. Call returns
	2. Call returns a byte array with a maximum size of 1660 bytes
	3. Call returns BER-TLV formatted byte array with the following byte sequence:
	`DataObject[0] == 0x70`
	
	`IF tag 0x70 Length < 128`
	`DataObject[1] <= 0x7F`
	`Set byte i = DataObject[1]`
	`ELSE`
	`DataObject[1..2] <= 0x06 4C`
	`Set byte i = DataObject[1..2] + 1`
	
	`DataObject[1+i] == 0x71`
	`DataObject[2+i] == 0x01`
	
	`DataObject[4+i] == 0x72`
	`DataObject[5+i] <= 0x26`
	
	`Set byte i = i + DataObject[5+i]`
	
	`DataObject[6+i] == 0xFE`
	`DataObject[7+i] == 0x00`
	`DataObject[8+i..9+i] == 0x90 00`

Post Condition(s)	• Client Application has received the requested data object with "Status Words" appended.

D.11 "Security Object" Data Object

Purpose	Confirms that the "Security Object" data object of the PIV Card Application conforms to the PIV Data Model requirements as per Appendix A of SP800-73
Target	`Security Object`
Reference(s)	1. SP800-73, Section 7.1.2
Precondition(s)	1. There exists valid physical connection between an instance of the PIV Card and the host of the calling application. 2. The PIV Application is the currently selected application. 3. The required security condition is satisfied. 4. The actual content of the value fields in the "Security Object" data object is obtained from the vendor
Test Scenario	1. `Call GET DATA w/ the following command (in hexadecimal)` • `00 CB 3F FF 05 5C 5F C1 06 00`

Expected Result(s)	1. Call returns
	2. Call returns a byte array with a maximum size of 1007 bytes
	3. Call returns BER-TLV formatted byte array with the following byte sequence:
	`DataObject[0] == 0xBA`
	`DataObject[1] <= 0x64`
	
	`Set byte i = DataObject[1]`
	
	`DataObject[2 + i] <= 0xBB`
	
	`IF tag 0xBB Length < 128`
	`DataObject[3 + i] <= 0x7F`
	`Set byte i = i + DataObject[3 + i]`
	`ELSE`
	`DataObject[3 + i..4 + i] <= 0x03 84`
	`Set byte i = i + DataObject[1..2] + 1`
	
	`DataObject[3+i] == 0xFE`
	`DataObject[4+i] == 0x00`
	`DataObject[5+i..6+i] == 0x90 00`

Post Condition(s)	• Client Application has received the requested data object with "Status Words" appended.

Appendix E—PIV Authentication Use Case Test Assertions

Test Assertion Template

Purpose	A quick description of the test and why it is being run
Reference(s)	References to the SP800-73 or other relevant publications References to DTRs
Precondition(s)	Anything that must be done or known prior to executing the scenario
Test Scenario	`Sequence of events and card responses` `Card Commands`
Expected Result(s)	What the expected execution path yields in terms of progress and values

E.1 Authentication Using PIV CHUID

Purpose	Validates that the PIV Card can be authenticated using the PIV CHUID
Reference(s)	1. SP800-73, Section C.1.2
Precondition(s)	1. A valid PIV Card is inserted into the contact reader 2. There exists a valid PC/SC connection between the test system and an instance of the contact reader 3. No application is currently connected to the PIV Card Application

Test Scenario	`1. Send SELECT card command with,`
	`• AID == A0 00 00 03 08 00 00 10 00 01 00`
	
	`2. Send GET DATA command with,`
	`• Data field of the command containing the tag of the CHUID data object`

Expected Result(s)	1. The command returns the CHUID data object along with the signature of the data object with the response status code ’90 00’. 2. Validate the digital signature of the CHUID 3. Validate that the PIV Card is valid by checking the Expiration Date field in the CHUID.

E.2 Authentication Using PIV Authentication Key

Purpose	Validates that the PIV Card can be authenticated using the PIV Authentication Key
Reference(s)	1. SP800-73, Section C.1.4
Precondition(s)	1. The length of the challenge supported by the card is obtained

Test Scenario	`1. Send SELECT card command with,`
	`• AID == A0 00 00 03 08 00 00 10 00 01 00`
	
	`2. Send VERIFY card command with,`
	`• P2, key reference value is set to ‘80’`
	`• Data field of the command will contain the PIV card application PIN value obtained from the vendor, padded with ‘FF’ to complete the total length of the field to`

	8 bytes 3. Send GET DATA command with, • Data field of the command containing the tag of the X.509 Certificate for PIV Authentication data object 4. Send GENERAL AUTHENTICATE card command with a random challenge with, • CLA is set to 00 • P1, algorithm reference, is set to ‘06’ identifying the 1024 bits RSA algorithm identifier • P2, key reference, is set to ‘9A’ indicating the PIV Authentication Key • Data field in the command is to include ‘81’ specifying a challenge, followed by a randomly generated challenge
Expected Result(s)	1. The command returns ’90 00’ 2. The command returns ’90 00’ 3. The command returns ’90 00’, followed by the X.509 certificate of the PIV authentication KEY. Validate the Certificate by validating the signature, the expiration date and the revocation status. 4. The command returns the encrypted challenge with ’90 00’ at the end. Decrypt the encrypted challenge and compare it to the one sent to the card.

Appendix F—Test Reports

Following execution of each test class, test labs will prepare a summary report to document and communicate the result of all test cases belonging to that class. A new report shall be kept for each separate run against the same unit under test. Each test report shall be signed and dated by the tester, the lab representative and the authorized NIST personnel when needed, and accompany a cover letter to be sent to the vendor to inform the outcome of the test. Templates are provided below as samples and demonstrate the type of information to be included in the reports. Actual format of the report and the quantity of information contained may vary among labs.

F.1 PIV Client API Test Results Summary

Test Run No		**Test Time**		**Test Date**	
	Name/Description		**Version**		**Date**
Implementation Under Test (IUT)					
Test Assertion					

Test Case	Test Description	Result[6]
B.1	**Connection Test Assertions**	
B.1.1	Valid Path Test Assertions	
B.1.1.1	Initiate Standalone Connection	
B.1.1.2	Initiate Shared Connection	
B.1.2	Test Assertions for Error Conditions	
B.1.2.1	Malformed Connection Description	
B.1.2.2	Sharing a Locked Connection	
B.1.2.3	Attempting to Lock a Shared Connection	
B.1.2.4	Attempting to Open an Unsupported Connection	
B.2	**Disconnection Test Assertions**	
B.2.1	Valid Test Assertions	
B.2.1.1	Disconnect a Standalone Connection	
B.2.1.2	Disconnect a Shared Connection	
B.2.2	Test Assertions for Error Cases	
B.2.2.1	Logically disconnecting a Client Application from a physically disconnected Card Application	
B.2.2.2	Attempt Disconnect with Invalid Card Handle	
B.2.2.3	Disconnecting a previously disconnected Client Application	
B.3	**pivSelectCardApplication**	

[6] The result should be either PASS or FAIL.

B.3.1	Valid Test Assertions	
B.3.1.1	Select a Card Application with a full AID	
B.3.1.2	Use a right truncated AID to Select a Card Application	
B.3.2	Test Assertions for Error Conditions	
B.3.2.1	Detect and handle an invalid cardHandle reference	
B.3.2.2	Detect and handle an invalid applicationAID	
B.4	**pivLogIntoCardApplication**	
B.4.1	Valid Test Assertions	
B.4.1.1	Log on to the Card Application	
B.4.2	Test Assertions for Error Conditions	
B.4.2.1	Attempt Logon without an invalid cardHandle	
B.4.2.2	Attempt Logon with a malformed authenticator	
B.4.2.3	Attempt Logon with invalid authenticator	
B.4.2.4	Attempt Logon with failed pivSelectCardApplication	
B.5	**pivLogoutOfCardApplication**	
B.5.1	Valid Test Assertions	
B.5.1.1	Log out of the Card Application	
B.5.2	Test Assertions for Error Conditions	
B.5.2.1	Attempt Log out with Invalid Cardhandle	
B.5.2.2	Attempt Log out with no security context	
B.6	**pivGetData**	
B.6.1	Valid Test Assertions	
B.6.1.1	Get a reference to data on a Client Application data object	
B.6.2	Test Assertions for Error Conditions	
B.6.2.1	Handle an invalid cardHandle	
B.6.2.2	Handle an invalid Object Identifier	
B.6.2.3	The Client Application can handle missing data object	
B.6.2.4	Security Conditions are enforced for Secured Objects	
B.7	**pivPutData**	
B.7.1	Valid Test Assertions	
B.7.1.1	Write data to an object on the Client Application	
B.7.2	Test Assertions for Error Conditions	
B.7.2.1	Identify and handle an invalid card handle	
B.7.2.2	Identify and handle an invalid Object Identifier (OID)	
B.7.2.3	Client Application can handle missing data object	

B.7.2.4	Identify and handle an excessively large data parameter	
B.7.2.5	Security Conditions are enforced for Secured Objects	
B.8	**pivGenerateKeyPair**	
B.8.1	Valid Test Assertions	
B.8.1.1	Generate an asymmetric key pair	
B.8.2	Test Assertions for Error Conditions	
B.8.2.1	Identify and handle an invalid card handle	
B.8.2.2	Identify and handle an invalid keyReference	
B.8.2.3	Identify and handle an invalid cryptographicMechanism	
B.8.2.4	Identify and handle an invalid keyReference / cryptographicMechanism combinations	
B.9	**pivCrypt**	
B.9.1	Valid Test Assertions	
B.9.1.1	Encrypt a sequence of bytes	
B.9.2	Test Assertions for Error Conditions	
B.9.2.1	Identify and handle invalid card handles	
B.9.2.2	Identify and handle invalid key references	
B.9.2.3	Identify and handle invalid input data	
B.9.2.4	Identify and handle invalid keyReference/algorithmIdentifier combinations	
B.9.2.5	Identify and handle pivGenerateKeyPair method errors	

F.2 Card Command Interface Test Results Summary

Test Run No		Test Time		Test Date	
		Name/Description		Version	Date
Implementation Under Test (IUT)					
Test Assertions File					

Test Case	Test Description	Result[7]	
		Contact	Contactless
C.1	Card Commands for Data Access		
C.1.1	SELECT Card Command		
C.1.1.1 & C.1.1.3	Long AID		
	Right-truncated short AID		
C.1.1.2	Error Condition		
	Invalid AID		
C.1.2	GET DATA card command		
C.1.2.1 & C.1.2.2	NO PIN		
	1. CCC		
	2. CHUID		
	3. PIV Authentication Object		
	4. Card Holder Fingerprint I data object		
	5. Card Holder Fingerprint II data object		
	6. Printed Information data object		
	7. Card Holder Facial Image data object		
	8. X.509 Certificate - Digital Signature Key data object		
	9. X.509 Certificate - Key Management Key data object		
	10. X.509 Certificate - Card Authentication Key data object		
	11. Security Object		
	WITH PIN		
	12. PIV Authentication Object		
	13. Card Holder Fingerprint I data object		
	14. Card Holder Fingerprint II data object		

[7] The result should be either PASS or FAIL.

	15. Printed Information data object		
	16. Card Holder Facial Image data object		
	17. X.509 Certificate - Digital Signature Key data object		
	18. X.509 Certificate - Key Management Key data object		
C.2	Commands For Authentication		
C.2.1	VERIFY Card Command		
C.2.1.1 – C.2.1.2	1. Valid PIN with invalid key reference		
	2. Valid PIN		
	3. PIN with invalid format		
	4. Invalid PIN		
C.2.2	CHANGE REFERENCE DATA card command		
C.2.2.1 – C.2.2.2	1. Valid PIN		
	2. Valid PIN with invalid key reference		
	3. PIN with invalid format		
	4. Invalid PIN (repeated invocation until PIN is blocked)		
C.2.3	RESET RETRY COUNTER command		
C.2.3.1 – C.2.3.2	1. Invalid Key Reference, valid PIN and Valid PUK		
	2. Valid PIN with invalid format		
	3. Valid PIN and PUK combination		
	4. Invalid PIN until the card is blocked		
C.2.4	GENERAL AUTHENTICATE command		
C.2.4.1 – C.2.4.2	1. Successful Internal Authenticate		
	2. Successful Mutual Authenticate		
	3. Response length is set to small value		
	4. Invalid key reference		
	5. Invalid data in the data field		
C.3	**Card Commands for Credential Initialization and Administration**		
C.3.1	PUT DATA Command		
C.3.1.1 – C.3.1.2	Without authenticating the card application administrator		
	1. CCC		
	2. CHUID		

	3. PIV Authentication Certificate		
	4. Card Holder Fingerprint I		
	5. Card Holder Fingerprint II		
	6. Printed Information		
	7. Card Holder Facial Image		
	8. Certificate for Digital Signature		
	9. Certificate for Key Management		
	10. Card Authentication Key		
	11. Security Object		
	With authenticating the card application administrator		
	1. CCC		
	2. CHUID		
	3. PIV Authentication Certificate		
	4. Card Holder Fingerprint I		
	5. Card Holder Fingerprint II		
	6. Printed Information		
	7. Card Holder Facial Image		
	8. Certificate for Digital Signature		
	9. Certificate for Key Management		
	10. Card Authentication Key		
	11. Security Object		
C.3.2	GENERATE ASYMMETRIC KEY PAIR command		
C.3.2.1 - C.3.2.2	Without authenticating with the PIV Card Application Administrator		
	1. Invoke command		
	2. Mutual Authenticate using key ’9B’		
	3. Invoke command		
	4. Invalid length for response		
	5. Invalid cryptographic algorithm		
	6. Invalid key reference-cryptographic mechanism combination		

F.3 Data Objects Representation Test Results Summary

Test Run No		Test Time		Test Date	
	Name/Description		Version		Date
Implementation Under Test (IUT)					
Test Assertions File					

Test Case	Test Description	Result[8]		
		OID	Implemented BER-TLV Format	Correct Format (Y/N)
D.1	Card Capabilities Container Data Object			
D.2	Card Holder Unique Identifier Data Object			
D.3	X.509 Certificate - PIV Authentication Data Object			
D.4	Card Holder Fingerprint I Data Object			
D.5	Card Holder Fingerprint II Data Object			
D.6	Printed Information Data Object			
D.7	Card Holder Facial Image Data Object			
D.8	X.509 Certificate - Digital Signature Data Object			
D.9	X.509 Certificate - Key Management Data Object			
D.10	X.509 Certificate - Card Authentication Data Object			
D.11	Security Object Data Object			

[8] The result should be either PASS or FAIL.

F.4 PIV Authentication Use Case Test Results Summary

Test Run No		Test Time		Test Date	
	Name/Description		Version	Date	
Implementation Under Test (IUT)					
Test Assertions File					

Test Case	Test Description	Result		
		Read Object	Validate Signature	Expiry Date
E.1	Authentication Using PIV CHUID			
E.2	Authentication Using PIV Authentication Key			

F.5 Detail Test Report

Test Case Information				
Name/Description			Number	
Conductor			Test Run	
Implementation Under Test (IUT)			Version	
Test Date			Time	
Type of Test	o Client API	o Card Command	o Data Object	o Use Case
Test Result	o PASS		o FAIL	
Test Details				
Command / Function name				
Test Step No				
Expected Result				
Outcome				
Reason for Discrepancy				
Corrective steps				
Comments				

Appendix G—Acronyms

The following acronyms and abbreviations are used throughout this standard:

AID	Application Identifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
BER-TLV	Basic Encoding Rules Tag-Length-Value
BSP	Biometric Service Provider
CBEFF	
CCC	Card Capability Container
CHUID	Cardholder Unique Identifier
CSP	Cryptographic Service Provider
DES-ECB	Data Encryption Standard Electronic Code Book
DTR	Derived Test Requirement
FIPS	Federal Information Processing Standards
FISMA	Federal Information Security Management Act
HSPD	Homeland Security Presidential Directive
ICC	Integrated Circuit Chip
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
ITL	Information Technology Laboratory
NIST	National Institute of Standards and Technology
OID	Object Identifier
OMB	Office of Management and Budget
PIN	Personal Identification Number
PIV	Personal Identity Verification
PIX	Proprietary Identifier eXtension
RID	Registered application provider IDentifier
SP	Special Publication