ActivID® ActivClient® SDK
PIV API Reference Guide

DOCUMENT REFERENCE: AC_SDK_PIV_7.1_RG_07.2016

PRODUCT VERSION: 7.1

JULY 2016



hidglobal.com

Copyright

© 2008-2016 HID Global Corporation/ASSA ABLOY AB. All rights reserved. This document may not be reproduced, disseminated or republished in any form without the prior written permission of HID Global Corporation.

Trademarks

HID, HID Global, the HID Blue Brick logo, the Chain Design and 4TRESS, ActivIdentity and ActivID are trademarks or registered trademarks of HID Global or its licensor(s)/supplier(s) in the US and other countries and may not be used without permission. The absence of a mark, product, service name or logo from this list does not constitute a waiver of the HID Global trademark or other intellectual property rights concerning that name or logo. All other trademarks, service marks, and product or service names are trademarks or registered trademarks of their respective owners.

Revision History

Date	Description	Document Version
July 2016	Initial release of rebranded document and major technical updates.	1.0

Contacts

Americas		Asia Pacific
United States: +1 510.574.0100	U.S. Federal: +1 571.522.1000	+61 (0) 3.9809.2892
Europe		**Corporate**
+33 (0) 1.42.04.84.00		+1 800.237.7769

Technical Support

If you purchased your product from a third party, then please contact that third party for Technical Support. For products purchased directly from HID Global, please go to http://www.hidglobal.com/support

Typographic and Document Conventions

Typography	Description
blue	Cross-references within the document.
blue, underline	References to external web addresses.
bold	Action steps (paths, buttons, options); field and drop-down list labels; emphasis.
italic	File names, document titles, and file extensions.
`Code snippets`	Highlights `code snippets` within regular content.
`Code samples`	`Highlights code samples`
	**WARNING**: This symbol indicates a critical warning. It applies to actions that if taken or not taken will break the system. Read the warning carefully and follow it.
	**Important**: This symbol indicates something very important to the reader. Ignore this symbol at your own risk.
	**Note**: This symbol indicates a note that should be of interest to the reader. It is not critical. Nevertheless, the reader should pay attention.

Table of Contents

1.0 Introduction 6

1.1 Product Overview 6
1.2 Document Scope and Audience 6
1.3 About the PIV API 7
1.3.1 Prerequisites 8
1.3.2 Standards 8

2.0 Overview 9

2.1 Include Files Required 9
2.2 How to Deploy 9
2.3 Return Codes 9
2.4 Discovery Mode 9

3.0 Packing List 10

3.1 Header Files 10
3.2 Static Libraries 10
3.3 Samples 10
3.3.1 Features Demonstrated in the Samples 10
3.3.2 PIV C Samples 10
3.3.3 PIV Java Sample 10

4.0 Constants 11

4.1 ASN.1 OID Identifiers 11
4.2 NIST [SP 800-73] Cryptographic Mechanism Identifiers 12

5.0 Data Types and Structures 13

5.1 Basic Types 13
5.2 Connection Description Buffer 13
5.2.1 Tags 14
5.2.2 Obtaining the List of Smart Card Readers 14

6.0 Entry Points 15

6.1 Summary 15
6.1.1 Communication 15
6.1.2 Data Access 15
6.1.3 Cryptographic Operations 15
6.1.4 Credential Initialization and Administration 16
6.2 Communication 16
6.2.1 pivMiddlewareVersion() 16
6.2.2 pivConnect() 16
6.2.3 pivDisconnect() 19
6.3 Data Access 20
6.3.1 pivGetData() 20
6.3.2 pivLogIntoCardApplication() 22
6.3.3 pivLogoutOfCardApplication() 23

6.3.4 pivSelectCardApplication() 24
6.4 Cryptographic Operations 26
6.4.1 pivCrypt() 26
6.5 Credential Initialization and Administration 28
6.5.1 pivGenerateKeyPair() 28
6.5.2 pivPutData() 31

7.0 Return Codes 33

Appendix A: Terms and Acronyms 35

A.1. Terms 35
A.2. Acronyms 37

1.0 Introduction



ActivID® ActivClient® guards against an ever-changing threat landscape by providing organizations with risk-appropriate and secure access to corporate IT assets.

1.1 Product Overview

HID Global's ActivID ActivClient is the latest smart card and USB token middleware from that allows enterprise and government customers to easily use smart cards and USB tokens to protect workstations and networks.

ActivClient enables the use of PKI certificates and keys and one-time passwords and static password credentials on a smart card or USB token to secure:

- Desktop applications
- Network logon
- Remote access
- Web logon
- E-mail
- Electronic transactions

ActivClient provides the following range of services:

- PKI services
- Remote access and One-Time Password (OTP) services
- Remote session services
- Management services (for end users and administrators)
- Development services with a Software Development Kit (SDK)

1.2 Document Scope and Audience

This document describes the ActivClient PIV End-Point API. It provides an overview of the implementation of the Personal Identification Verification (PIV) for Federal Employees and Contractors' End-Point Client Application Programming Interface.

Readers of this document are assumed to be experienced C programmers who are:

- Implementing applications using the HID Global implementation of the PIV End-Point Client Application Programming Interface.
- Familiar with the NIST [SP800-73-3] and NIST [SP800-85A] specifications and the PIV data model.

Java programmers who will be using the PIV API should also consult the javadoc.

1.3 About the PIV API

The ActivID Personal Identification Verification (PIV) End-Point Client API (PIV API) fully implements the [800-73] NIST Special Publication specification. The implementation is independent of any PIV API card application implementation and supports any PIV-certified smart card.

The ActivClient PIV API is NIST-certified. The certificate is available at http://csrc.nist.gov/groups/SNS/piv/npivp/documents/PIVMiddlewareCertificate20.pdf.

The PIV standard defines two interfaces for communicating with PIV cards:

- The PIV transitional interface.
- The PIV end-point interface.

A PIV end-point card is a card that implements the second of these interfaces.

The PIV API provides the following services on NIST-certified PIV end-point cards:

- Data access
- Cryptographic operations
- Credentials initialization and management

The PIV API is provided as a C API. However, Java applications can use the PIV API as well. HID Global provides a Java wrapper for the PIV API and the ActivClient SDK provides samples in C and Java to demonstrate how to call the PIV API from various languages.

Purpose: Use this API for PKI based application using PIV End Point card or PIV data retrieval.

Supports: PKI and PIV data access, as well as mutual authentication and external authentication.

Languages: C, Java; HID Global provides samples in C and Java.

Description: The ActivClient PIV API is an implementation of Personal Identity Verification (PIV) Middleware API as per National Institute of Standard and Technology (NIST) SP800-73-3 specifications. This API provides cryptographic, data storage, and utility services for FIPS 201 PIV-compliant cards.

The ActivClient PIV API:

- Provides support for:
  - Smart card cryptographic and data storage operation
  - Client-based management on smart cards and smart card readers
- Is recommended for developers who want to perform smart card cryptographic and data retrieval operations specifically on FIPS 201 PIV-compliant cards.

1.3.1 Prerequisites

The PIV API is supported on platforms where ActivClient has been installed. Refer to the *ActivID ActivClient for Windows Installation Guide* for pre-requisites for ActivClient installation.

1.3.2 Standards

The PIV API conforms to the NIST PIV standard. For additional information, see:

- Reference information on the PIV standard (http://csrc.nist.gov/groups/SNS/piv/npivp/)
- [SP800-73-3] NIST Special Publication SP800-73 Part 1, Part 2 and Part 3

2.0 Overview

2.1 Include Files Required

The necessary include files are located in the **PIV/C/Headers** folder of the distribution.

2.2 How to Deploy

PIV API applications must be linked with the appropriate PIV import library. The import library for x86 applications is located in the **PIV/C/Libraries/x86** directory. The import library for x64 applications is in the **PIV/C/Libraries/x64** directory.

2.3 Return Codes

The HID Global implementation translates the Microsoft PC/SC error codes or PIV card application error code into PIV API error codes. When applicable, the description of each function in Entry Points on page 15 indicates the mapping from the PC/SC and/or PIV card application error code to the corresponding PIV error code. For more information about return codes, see Return Codes on page 33.

2.4 Discovery Mode

[SP800-85A] Section F.3, Footnote 2 specifies that discovery mode, which enables 'calling applications that do not have an a priori knowledge of the size of the data returned to obtain the size by making the discovery call and then make a second call with the right buffer size allocated for the output parameter to retrieve the data', is optional in PIV implementations.

The use of discovery mode is supported for any PIV API call that returns output data. These calls are:

- pivSelectCardApplication()
- pivGetData()
- pivGenerateKeyPair()
- pivCrypt() when called for signatures.

The discovery mode is not applicable for `pivCrypt()` in case of external or mutual authentication because output data is not expected in those cases.

3.0 Packing List

3.1 Header Files

- piv.h
- pivd.h
- pivf.h
- pivt.h

3.2 Static Libraries

- acpivapi.lib, x86 version
- acpivapi.lib, x64 version

3.3 Samples

3.3.1 Features Demonstrated in the Samples

The samples implement the following scenarios:

- How to use a PIV signature key to sign data (sample only supports certificates with 2048-bit RSA keys).
- How to read PIV objects from the card.

3.3.2 PIV C Samples

Samples for both the 32 and 64-bit platforms are provided.

The PIV C x86 sample works with both ActivClient 32 and 64-bit editions.

3.3.3 PIV Java Sample

The sample uses the PIV Java wrapper to demonstrate the same features as the C sample. This sample runs on both the 32 and 64-bit platforms.

Javadoc is provided as well.

4.0 Constants

4.1 ASN.1 OID Identifiers

These identifiers are used to identify the PIV objects in the `pivPutData()` and `pivGetData()` methods.

ASN.1 OID	Description
2.16.840.1.101.3.7.1.219.0	Smart card Capability Container
2.16.840.1.101.3.7.2.48.0	Smart card Holder Unique Identifier
2.16.840.1.101.3.7.2.1.1	X.509 Certificate for PIV Authentication
2.16.840.1.101.3.7.2.96.16	Smart card Holder Fingerprints
2.16.840.1.101.3.7.2.48.1	Printed Information
2.16.840.1.101.3.7.2.96.48	Smart card Holder Facial Image
2.16.840.1.101.3.7.2.1.0	X.509 Certificate for Digital Signature
2.16.840.1.101.3.7.2.1.2	Certificate for Key Management
2.16.840.1.101.3.7.2.5.0	X.509 Certificate for Smart card Authentication
2.16.840.1.101.3.7.2.144.0	Security Object
2.16.840.1.101.3.7.2.96.80	Discovery Object
2.16.840.1.101.3.7.2.96.96	Key History Object
2.16.840.1.101.3.7.2.16.1	Retired X.509 Certificate for Key Management 1
2.16.840.1.101.3.7.2.16.2	Retired X.509 Certificate for Key Management 2
2.16.840.1.101.3.7.2.16.3	Retired X.509 Certificate for Key Management 3
2.16.840.1.101.3.7.2.16.4	Retired X.509 Certificate for Key Management 4
2.16.840.1.101.3.7.2.16.5	Retired X.509 Certificate for Key Management 5
2.16.840.1.101.3.7.2.16.6	Retired X.509 Certificate for Key Management 6
2.16.840.1.101.3.7.2.16.7	Retired X.509 Certificate for Key Management 7
2.16.840.1.101.3.7.2.16.8	Retired X.509 Certificate for Key Management 8
2.16.840.1.101.3.7.2.16.9	Retired X.509 Certificate for Key Management 9
2.16.840.1.101.3.7.2.16.10	Retired X.509 Certificate for Key Management 10
2.16.840.1.101.3.7.2.16.11	Retired X.509 Certificate for Key Management 11
2.16.840.1.101.3.7.2.16.12	Retired X.509 Certificate for Key Management 12

ASN.1 OID	Description
2.16.840.1.101.3.7.2.16.13	Retired X.509 Certificate for Key Management 13
2.16.840.1.101.3.7.2.16.14	Retired X.509 Certificate for Key Management 14
2.16.840.1.101.3.7.2.16.15	Retired X.509 Certificate for Key Management 15
2.16.840.1.101.3.7.2.16.16	Retired X.509 Certificate for Key Management 16
2.16.840.1.101.3.7.2.16.17	Retired X.509 Certificate for Key Management 17
2.16.840.1.101.3.7.2.16.18	Retired X.509 Certificate for Key Management 18
2.16.840.1.101.3.7.2.16.19	Retired X.509 Certificate for Key Management 19
2.16.840.1.101.3.7.2.16.20	Retired X.509 Certificate for Key Management 20
2.16.840.1.101.3.7.2.16.21	Cardholder Iris Images

4.2 NIST [SP 800-73] Cryptographic Mechanism Identifiers

The following identifiers are used to identify the type of key pair to be generated by the `pivGenerateKeyPair()` method. The table below corresponds to Table 20 of NIST [SP 800-73].

Cryptographic Mechanism Identifier	Description
05	RSA 3072
06	RSA 1024
07	RSA 2048

5.0 Data Types and Structures

Data types and other structures used by the PIV API are described in this chapter.

5.1 Basic Types

The HID Global implementation follows the C language binding of basic types defined in 800-85A.

PIV_Bool

Boolean.

```
typedef unsigned char PIV_Bool
```

PIV_Byte

Byte.

```
typedef unsigned char PIV_Byte
```

PIV_CARDHANDLE

Specify a type for the card connection handle managed by PIV.

```
typedef unsigned long PIV_CARDHANDLE
```

PIV_RV

Type used for all PIV API functions' return value.

```
typedef unsigned long PIV_RV
```

PIV_ULong32

Unsigned Long.

```
typedef unsigned long PIV_ULong32
```

5.2 Connection Description Buffer

This section provides a detailed description of the `ConnectionDescription` `pivConnect()` function parameter. This parameter, defined in [SP800-73-1] section 5.4.Table 11, represents a BER-TLV buffer containing the connection description data to be used to connect to the PIV API card application.

The connection description V-Buffer consists of an ordered sequence of tags and values. Each tag labels a specific type of information about the connection. Each tag is followed by a length indicating the length of the value buffer corresponding to that tag, and then a value buffer that indicates the actual value of the information for the corresponding tag.

5.2.1 Tags

The HID Global implementation supports the following tags:

- 0x81: PC/SC tag
- 0x90: Network node tag

In the case of the 0x90 tag, the HID Global implementation ignores the length and value buffers that follow the 0x90 tag.

Other tags are rejected by the `pivConnect()` function. An error is returned depending of the tag value:

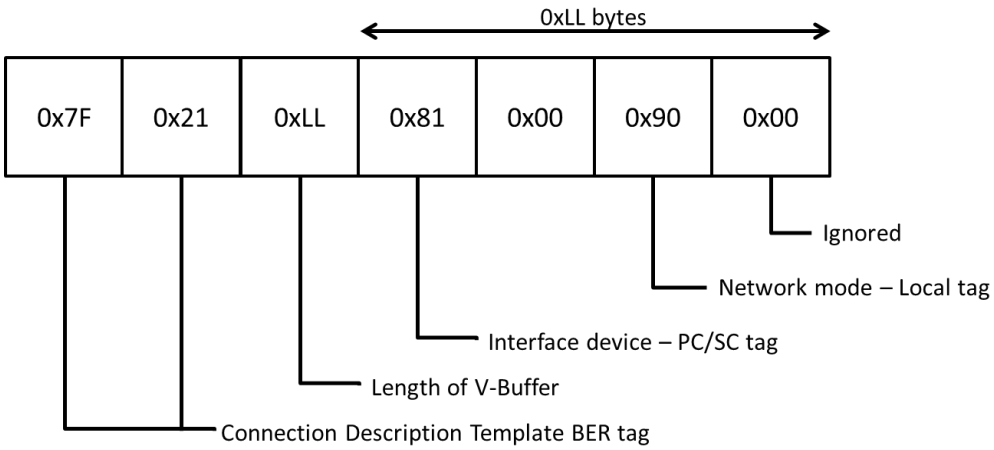
- If the value of the unsupported tag is a value described in NIST [SP 800-73] publication (0x82, 0x83, 0x84, 0x85, 0x86, 0x91, 0x92, 0x93), then the function returns PIV_CONNECTION_FAILURE.
- In all other cases, the function returns PIV_CONNECTION_DESCRIPTION_MALFORMED.

The presence of both tags (0x81 and 0x90) in the connection description is mandatory. This implies a minimum size for the input buffer of 7 bytes.

5.2.2 Obtaining the List of Smart Card Readers

The ConnectionDescription buffer should be formatted as shown in the following illustration when used to retrieve the list of smart card readers that are connected to the PC:

Figure 1: ConnectionDescription buffer used to retrieve a list of readers attached to a PC



Note: 0xLL is the length of the V-buffer of the BER-TLV connection description buffer.

6.0 Entry Points

6.1 Summary

6.1.1 Communication

Method	Description
pivMiddlewareVersion()	Returns the PIV Middleware version string.
pivConnect()	Connects the client application to the PIV smart card application.
pivDisconnect()	Disconnects the client application from the PIV smart card application.

6.1.2 Data Access

Method	Description
pivGetData()	Returns the entire data content of the named data object.
pivLogIntoCardApplication()	Authenticates the client application to the smart card application (for example, present a PIN to the smart card) and hence establishes the initial security status in the smart card application context.
pivLogoutOfCardApplication()	Resets the applications security status of the PIV smart card application.
pivSelectCardApplication()	Sets the currently selected smart card application.

6.1.3 Cryptographic Operations

Method	Description
pivCrypt()	Performs a cryptographic operation such as encryption or signing.

6.1.4 Credential Initialization and Administration

Method	Description
pivGenerateKeyPair()	Generates an asymmetric key pair from the PIV smart card application.
pivPutData()	Replaces the entire data content of the named data object with the provided data.

6.2 Communication

6.2.1 pivMiddlewareVersion()

Returns the PIV Middleware version string

Syntax

```
PIV_RV pivMiddlewareVersion (
char * versionString)
```

Parameters

`versionstring`

[inout] Pointer to a buffer containing the version of PIV Middleware. For SP 800-73-3 Part 3 conformant PIV Middleware, the parameter returns “800-73-3 Client API”.

Returns

`PIV_OK`

versionString has been has been retrieved successfully.

`PIV_INPUT_BYTES_MALFORMED`

versionString could not be retrieved

6.2.2 pivConnect()

Connects the client application to the PIV smart card application on a specific integrated circuit chip (ICC). If the smart card that is inserted in the reader and to which the method is attempting to connect does not contain a PIV smart card application, then the function fails with an error code. Otherwise, the function succeeds and the PIV smart card application (AID = [A0 00 00 03 08 00 00 10 00]) is selected.

Syntax

```
PIV_RV pivConnect (
```

PIV_Bool sharedConnection, PIV_Byte *connectionDescription, PIV_ULong32 *pCDLength, PIV_CARDHANDLE *pCardHandle)

Parameters

`sharedConnection`

[in] Boolean value that indicates whether other applications can establish concurrent connections with the PIV smart card application. If this parameter is true then the connection is shared, otherwise it is exclusive.

Note: To be compatible with ActivClient, the installed connection must be shared.

`connectionDescription`

[inout]

Input: Pointer to a buffer containing a connection description BER-TLV data object. This buffer should conform to the format described in Connection Description Buffer on page 13. This parameter must be large enough to hold the returned value.
This parameter should never be NULL.

Output: When the connection description is used to retrieve the list of readers (tag 0x81 and NULL length), the function updates the `ConnectionDescription` buffer with the list of readers. The format of the returned list is:

reader 1	0x0	reader 2	0x0	.....	reader n	0x0	0x0

The `connectionDescription` buffer length is indicated by the `pCDLength` parameter.

`pCDLength`

[inout]

Input: Pointer to an unsigned long containing the `connectionDescription` buffer size. This parameter should never be NULL.

Output: When tag 0x81 is used with a null length to get the reader list, the function returns the reader list length into this parameter. This value is used by the calling application to allocate the correct input buffer size if necessary.

`pCardHandle`

[out]

Pointer to a PIV_CARD_HANDLE. If this function call succeeds, the returned `CardHandle` can be used in all subsequent calls to other PIV API functions. Otherwise the output value should not be used.

Returns

`PIV_OK`

The smart card is positioned in the reader and the reader returns a handle. Return Code: SCARD_S_SUCCESS from PC/SC.

PIV_CONNECTION_DESCRIPTION_MALFORMED

Either:

- The content of the connectionDescription parameter does not conform to Table 11 of [SP 800-73].
- The description buffer length as indicated by the pCDLength parameter is too small to hold the returned reader list.
- When attempting to connect to the reader, PC/SC returned one of the following error codes:

SCARD_E_UNKNOWN_READER
SCARD_E_DUPLICATE_READER

PIV_CONNECTION_LOCKED

Either:

- A shared connection exists and an exclusive connection was requested.
- An exclusive connection exists and a shared/an exclusive connection was requested. Return Code: SCARD_E_SHARING_VIOLATION from PC/SC

PIV_CONNECTION_FAILURE

This may have occurred because:

- There is no smart card in the reader.
- A wrong connection mode was specified (say, ISDN connection parameters for a PC/SC reader).
- The PC/SC service was not started (perhaps because the reader is not connected to local host).

PC/SC return codes handled by the PIV API in its pivConnect() implementation, and mapped into the PIV_CONNECTION_FAILURE error code:
SCARD_E_NO_SMARTCARD
SCARD_E_UNKNOWN_CARD
SCARD_W_UNRESPONSIVE_CARD
SCARD_E_INVALID_ATR
SCARD_E_NO_SERVICE
SCARD_E_NO_READERS_AVAILABLE
SCARD_F_COM_ERROR

6.2.3 pivDisconnect()

Disconnects the PIV application programming interface from the PIV smart card application and the ICC containing the PIV smart card application.

Syntax

```
PIV_RV pivDisconnect ( PIV_CARDHANDLE cardHandle)
```

Parameters

`cardHandle`

[in] Communication handle returned by a successful call to the `pivConnect()` function. After the call to pivDisconnect, this handle is undefined and should not be used in subsequent calls.

Returns

`PIV_OK`

The request to disconnect was successful.

`PIV_INVALID_CARD_HANDLE`

Either:

- Function was called with an invalid smart card handle, or
- Successful disconnect was already called with this smart card handle

```
Return code from PC/SC: SCARD_E_INVALID_HANDLE
```

`PIV_CARD_READER_ERROR`

- Reader is not connected to the local host.
- There is no smart card in the reader.
- An event has occurred that prevents communication with the smart card.

The following return codes are the PC/SC return codes handled by the PIV API in its `pivDisconnect()` implementation and mapped into the PIV_CARD_READER_ERROR error code:

```
SCARD_F_COMM_ERROR
SCARD_W_UNPOWERED_CARD
SCARD_W_RESET_CARD
SCARD_W_REMOVED_CARD
SCARD_E_READER_UNAVAILABLE
```

6.3 Data Access

6.3.1 pivGetData()

Returns the entire data content of the named data object.

Note: The HID Global implementation of `pivGetData()` does not require the calling application to select the PIV smart card application (`pivSelectCardApplication()`) before it calls this function.

Syntax

```
PIV_RV pivGetData (
PIV_CARDHANDLE cardHandle,
PIV_Byte *OID,
PIV_ULong32 oidLength,
PIV_Byte *data, PIV_ULong32 *pDataLength)
```

Parameters

`cardHandle`

[in] Communication handle returned by a successful call to the `pivConnect()` function.

`OID`

[in] Pointer to a byte string containing the ASN.1 identifier of the object whose data content is to be retrieved. See ASN.1 OID Identifiers on page 11 for a list of supported identifiers.

This parameter should never be NULL. The function fails if any other ASN.1 OID is used. This function fails if the smart card does not contain the object with the corresponding OID.

`oidLength`

[in] Unsigned long containing the OID buffer size. This parameter should never be NULL.

`data`

[out] Pointer to a buffer to be set by the function with the required data content. This pointer may be NULL if the calling application requires only the length of the data content. If this pointer is NULL, then the pDataLength pointer should not be NULL.

Note: For performance purposes, HID Global middleware implements a secure data cache mechanism to store data read from the smart card into memory. This mechanism ensures that data is read only once from the smart card, even if the `pivGetData()` function is called twice in a row.

`pDataLength`

[inout] This parameter should never be NULL.

Input: Pointer to an unsigned long containing the size of the data buffer. If data parameter is NULL, then the value pointed to by pDataLength may be NULL as well.

Output: The pointed value is set by the function with the data content length. This parameter should be used by the calling application to allocate the correct input buffer size if necessary, before calling this function once again.

Note: To retrieve the number of bytes required for the OID data (discovery mode), set the data parameter to NULL; this function sets the variable pointed to by pDataLength to the required length and returns PIV_INSUFFICIENT_BUFFER.If the data parameter is not NULL, the pDataLength parameter must not be NULL either.

Returns

`PIV_OK`

The OID is valid; an existing smart card handle is returned. Return code from smart card: 0x90 0x00

`PIV_INVALID_CARD_HANDLE`

Function was called with an invalid smart card handle. Return code from PC/SC: SCARD_E_INVALID_HANDLE

`PIV_CARD_READER_ERROR`

- The reader is not connected to the local host, or
- No smart card is in the reader, or
- An event occurred that prevented communication with the smart card.

The PC/SC error codes that are mapped into PIV_CARD_READER_ERROR in the `pivGetData()` implementation:

```
SCARD_F_COMM_ERROR
SCARD_W_UNPOWERED_CARD
SCARD_W_RESET_CARD
SCARD_W_REMOVED_CARD
SCARD_E_READER_UNAVAILABLE
```

`PIV_INVALID_OID`

The OID is not found in Table 6 of [SP 800-73].

`PIV_DATA_OBJECT_NOT_FOUND`

The OID refers to a data object not found on the smart card. Return code from smart card: 0x6A 0x82

`PIV_SECURITY_CONDITIONS_NOT_SATISFIED`

Access to the smart card object denoted by OID is denied since security condition for its access not satisfied. Return code from smart card: 0x69 0x82

PIV_INSUFFICIENT_BUFFER

The buffer supplied in the data parameter is too small. pDataLength contains the required size.

6.3.2 pivLogIntoCardApplication()

Establishes the application security status within the PIV smart card application, either by verifying the smart card holder PIV smart card application PIN or by performing an external authentication using the PIV smart card application administration key.

The HID Global implementation of pivLogIntoCardApplication() does not require the calling application to select the PIV smart card application (pivSelectCardApplication()) before it calls this function.

Syntax

```
PIV_RV pivLogIntoCardApplication (
PIV_CARDHANDLE cardHandle,
PIV_Byte *authenticators,
PIV_ULong32 pAuthLength)
```

Parameters

cardHandle

[in] Communication handle returned by a successful call to the pivConnect() function.

authenticators

[in] Pointer to a buffer containing a sequence of zero or more BER-TLV authenticator data objects. This buffer should conform to the format described in the Authenticator description section in [800-73-1] Table 10.

If the authenticators parameter is NULL, the AuthLength parameter must also be 0 and no authentication is performed. In this case, the function may return PIV_OK if no error condition occurs.

pAuthLength

[in] Unsigned long containing the authenticators buffer size. This parameter may be 0 if no authentication is to be performed; if this is the case, the function returns PIV_OK.

Returns

PIV_OK

Authenticators are well-formed; an existing smart card handle is returned. Return code from smart card: 0x90 0x00

PIV_INVALID_CARD_HANDLE

Function was called with an invalid smart card handle. Return code from PC/SC: SCARD_E_INVALID_HANDLE

PIV_AUTHENTICATOR_MALFORMED

The authenticator does not conform to the format specified in Table 10 of [SP 800-73-1]. Return code from smart card: 0x6A 0x80 or 0x6A 0x88

PIV_AUTHENTICATION_FAILURE

The reference data in the authenticator is not valid. Return code from smart card: 0x63Cx or 0x69 0x83

PIV_CARD_READER_ERROR

- The reader is not connected to the local host, or
- No smart card in the reader, or
- An event occurred that prevented communication with the smart card.

The PC/SC error codes that are mapped into PIV_CARD_READER_ERROR in the pivLogIntoCardApplication() implementation:

```
SCARD_F_COMM_ERROR
SCARD_W_UNPOWERED_CARD
SCARD_W_RESET_CARD
SCARD_W_REMOVED_CARD
SCARD_E_READER_UNAVAILABLE
```

6.3.3 pivLogoutOfCardApplication()

Resets the application security status of the PIV smart card application. Access rights acquired by a previous call to the pivLogIntoCardApplication() are lost.

The HID Global implementation does not require the calling application to select the PIV smart card application (pivSelectCardApplication()) before calling this function.

Syntax

```
PIV_RV pivLogoutOfCardApplication(
PIV_CARDHANDLE cardHandle)
```

Parameters

cardHandle

[in] Communication handle returned by a successful call to the pivConnect() function.

PIV_OK

An existing smart card handle is returned. Return code from smart card: 0x90 0x00

PIV_INVALID_CARD_HANDLE

Function was called with an invalid smart card handle. Return code from PC/SC: SCARD_E_INVALID_HANDLE

PIV_CARD_READER_ERROR

- The reader is not connected to the local host, or
- No smart card is in the reader, or
- An event occurred that prevented communication with the smart card.

The PC/SC error codes that are mapped into PIV_CARD_READER_ERROR in the pivLogoutOfCardApplication() implementation:

```
SCARD_F_COMM_ERROR
SCARD_W_UNPOWERED_CARD
SCARD_W_RESET_CARD
SCARD_W_REMOVED_CARD
SCARD_E_READER_UNAVAILABLE
```

6.3.4 pivSelectCardApplication()

Sets the currently selected smart card application. The call is successful if the given PIV smart card application is found; otherwise, an error code is returned.

Note: HID Global optimizes the implementation of this method so that if it is called multiple times once the application has been selected, it does not go to the card to select the application again. This means that invoking this method multiple times does not impact performance.Discovery mode is supported by this method; if the applicationProperties parameter is NULL, then this function sets the value pointed to by pAPLength to the required length of the applicationsProperties buffer and returns PIV_OK.

Syntax

```
PIV_RV pivSelectCardApplication (
PIV_CARDHANDLE cardHandle,
PIV_Byte *applicationAID,
PIV_ULong32 aidLength,
PIV_Byte *applicationProperties,
PIV_ULong32 *pAPLength)
```

Parameters

cardHandle

[in] Communication handle returned by a successful call to the pivConnect() function.

applicationAID

[in] Pointer to a buffer containing the AID of the PIV smart card application to select. It may be either a long AID including the version number or a short AID without version number. This parameter should never be NULL.

aidLength

[in] Unsigned long containing the applicationAID buffer size. This parameter should never be NULL.

applicationProperties

[out] Pointer to a buffer to be set by the function with the application properties of the specified smart card application. This pointer may be NULL if application properties are not required by the calling application. If this parameter is not NULL, then the pAPLength parameter should not be NULL either. If this pointer is NULL, discovery mode is triggered and the function returns PIV_INSUFFICIENT_BUFFER.

pAPLength

[inout]

Input: Pointer to an unsigned long containing the size in bytes of the applicationProperties buffer. This parameter must not be NULL.

Output: In discovery mode, the buffer to which this parameter points is set by the function with the length of the application properties. This information is used by the calling application to allocate the correct input buffer size, if necessary, before calling this function once again.

Returns

PIV_OK

An existing smart card handle and a valid application AID are returned. Return code from smart card: 0x90 0x00

PIV_INVALID_CARD_HANDLE

Function was called with an invalid smart card handle. Return code from PC/SC: SCARD_E_INVALID_HANDLE

PIV_CARD_APPLICATION_NOT_FOUND

Application AID does not refer to an AID that the smart card supports. Return code from smart card: 0x6A 0x82

PIV_CARD_READER_ERROR

- Reader is not connected to the local host,
- No smart card in the reader, or
- An event occurred that prevented communication with the smart card.

The PC/SC error codes that are mapped into PIV_CARD_READER_ERROR in the pivSelectCardApplication() implementation:

SCARD_F_COMM_ERROR

```
SCARD_W_UNPOWERED_CARD
SCARD_W_RESET_CARD
SCARD_W_REMOVED_CARD
SCARD_E_READER_UNAVAILABLE
```

`PIV_INSUFFICIENT_BUFFER`

pAPLength is NULL or applicationProperties is NULL or buffer is too small.

6.4 Cryptographic Operations

6.4.1 pivCrypt()

Performs a cryptographic operation such as encryption or signing on a sequence of bytes.

The HID Global implementation does not require the calling application to select the PIV smart card application (`pivSelectCardApplication()`) before calling this function.

Syntax

```
PIV_RV pivCrypt (
PIV_CARDHANDLE cardHandle,
PIV_Byte algID,
PIV_Byte keyReference,
PIV_Byte *algInput,
PIV_ULong32 inputLength,
PIV_Byte *algOutput,
PIV_ULong32 *pOutputLength)
```

Parameters

`cardHandle`

[in] Communication handle returned by a successful call to the `pivConnect()` function.

`algID`

[in] Unsigned char containing the identifier of the cryptographic algorithm to be used for the cryptographic operation. The HID Global implementation supports all the algorithms listed in Table 7 of NIST [SP800-73-1] publication:

- 05, 06, 07 for asymmetric algorithms.
- 01, 02, 03, 04, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F, 10, 11, 12, 13 for symmetric algorithms.
- 00 for symmetric or asymmetric algorithm depending on the keyReference.

Other values are rejected and the function returns an error.

keyReference

[in] Key reference value of the key to be used for the cryptographic operation. The HID Global implementation supports all the key reference values listed in Table 12 of the NIST [SP 800-73] publication other than 9A, 9B, 9C, and 9D. The HID Global implementation does support the value 9E, even though it is not listed in Table 12 of NIST [SP 800-73] (see PIV Project Questions & Answers Web site: http://piv.nist.gov/pivqa/faq.php?qid=122)

Other key reference values are rejected and the function returns an error.

algInput

[in] Pointer to a buffer containing the Dynamic Authentication Template to be sent to the smart card, including complete formatting and padding when necessary. For more information, see the Dynamic Authentication Template Format definition in Table 17 of [SP-800-73-1].

This parameter should never be NULL.

inputLength

[in] Unsigned long containing the algInput buffer size. This parameter should never be NULL.

algOutput

[out] Pointer to a buffer to be set by the function with the Dynamic Authentication Template returned by the smart card, including complete formatting.

This parameter can be NULL, in which case the function returns PIV_INSUFFICIENT_BUFFER.

pOutputLength

[out] Pointer to an unsigned long containing the algOutput buffer size.

This parameter can be NULL, in which case the function returns PIV_INSUFFICIENT_BUFFER.

Returns

Sequence of bytes output by the cryptographic operation.

PIV_OK

An existing smart card handle is returned. Return code from smart card: 0x90 0x00

PIV_INVALID_CARD_HANDLE

Function was called with an invalid smart card handle. Return code from PC/SC: SCARD_E_INVALID_HANDLE

PIV_CARD_READER_ERROR

- The reader is not connected to the local host, or
- No smart card is in the reader, or
- An event occurred that prevented communication with the smart card.

The PC/SC error codes that are mapped into PIV_CARD_READER_ERROR in the pivCrypt() implementation:

```
SCARD_F_COMM_ERROR
SCARD_W_UNPOWERED_CARD
SCARD_W_RESET_CARD
SCARD_W_REMOVED_CARD
SCARD_E_READER_UNAVAILABLE
```

PIV_INPUT_BYTES_MALFORMED

The algInput and/or inputLength parameter was NULL. Return code from smart card: 0x6A 0x80

PIV_SECURITY_CONDITIONS_NOT_SATISFIED

The sign operation is protected and the application security status was not established accordingly by a successful call to pivLogIntoCardApplication(). This error code is also returned if the PIN is locked.

Note: In case of mutual authentication or external authentication, when called to get the cryptogram, the pivCrypt function should not be called twice in a row. The second call will result in a PIV_SECURITY_CONDITIONS_NOT_SATISFIED error. Return code from smart card: 0x69 0x82 or 0x69 0x83

6.5 Credential Initialization and Administration

6.5.1 pivGenerateKeyPair()

Generates an asymmetric key pair in the currently selected application.

The HID Global implementation does not require the calling application to select the PIV smart card application (pivSelectCardApplication()) before it calls this function.

Syntax

```
PIV_RV pivGenerateKeyPair(
PIV_CARDHANDLE cardHandle,
PIV_Byte keyReference,
PIV_Byte cryptographicMechanism,
PIV_Byte *publicKey,
PIV_ULong32 *pKeyLength)
```

Parameters

cardHandle

[in] Communication handle returned by a successful call to the pivConnect() function.

KeyReference

[in] Key reference value of the key to be generated. The HID Global implementation supports all the key reference values listed in Table 12 of NIST [SP 800-73] publication, corresponding to asymmetric key pairs: 9A, 9C, 9D. The HID Global implementation supports value 9E as well, although it is not listed in Table 12 of the NIST [SP 800-73] (see PIV Project Questions & Answers Web site: http://piv.nist.gov/pivqa/faq.php?qid=122.)

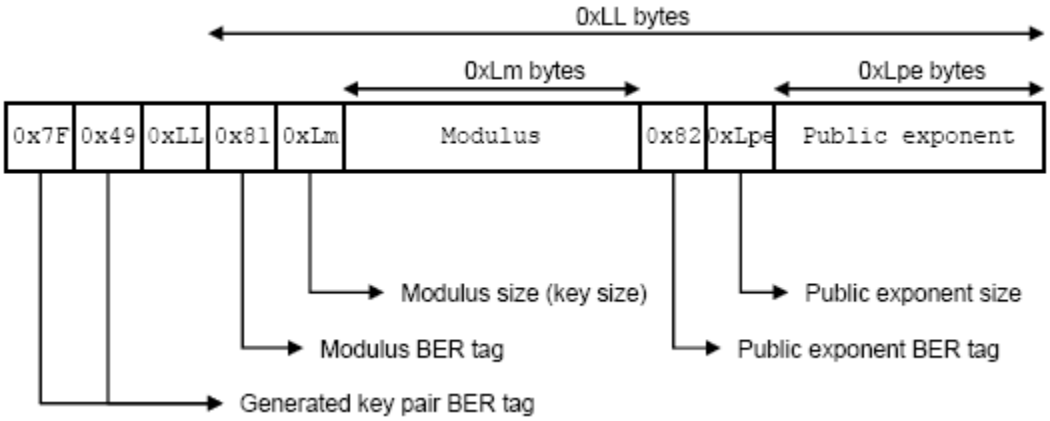
Other key reference values are rejected and the function returns an error.

CryptographicMechanism

[in] The type of the key pair to be generated. See NIST [SP 800-73] Cryptographic Mechanism Identifiers on page 12 for the supported cryptographic mechanism identifiers.

publicKey

[out] Pointer to the buffer to be set by the function with the public key of the generated key pair. Format of the returned buffer is:



This parameter can be NULL, in which case the function returns PIV_INSUFFICIENT_BUFFER and sets pKeyLength with the required buffer size.

pKeyLength

[out] Pointer to an unsigned long containing the size of the publicKey buffer.

This parameter can be NULL, in which case the function returns PIV_INSUFFICIENT_BUFFER.

Returns

PIV_OK

An existing smart card handle is returned.

PIV_INVALID_CARD_HANDLE

Function was called with an invalid smart card handle. Return code from PC/SC: SCARD_E_INVALID_HANDLE

PIV_CARD_READER_ERROR

- The reader is not connected to the local host, or
- No smart card is in the reader, or
- An event occurred that prevented communication with the smart card.

The PC/SC error codes that are mapped into PIV_CARD_READER_ERROR in the pivGenerateKeyPair() implementation:

SCARD_F_COMM_ERROR
SCARD_W_UNPOWERED_CARD
SCARD_W_RESET_CARD
SCARD_W_REMOVED_CARD
SCARD_E_READER_UNAVAILABLE

PIV_INVALID_KEY_OR_KEYALG_COMBINATION

The value of the keyReference parameter is unsupported as an RSA key reference value (see Table 12 of NIST [SP 800-73] publication and http://piv.nist.gov/pivqa/faq.php?qid=122). Return code from smart card: 0x6A 0x86

PIV_UNSUPPORTED_CRYPTOGRAPHIC_MECHANISM

The value of the cryptographicMechanism is not a supported value. Return code from smart card: 0x6A 0x80

PIV_SECURITY_CONDITIONS_NOT_SATISFIED

The generated key pair operation is protected and the application security status was not established by a successful call to the pivLogIntoCardApplication(). This error code is also returned if the PIN is locked. Return code from smart card: 0x69 0x82 or 0x69 0x83

PIV_INSUFFICIENT_BUFFER

The supplied output buffer is too small. pKeyLength contains the required buffer size.

6.5.2 pivPutData()

Replaces the entire data content of the named data object with the provided data. The HID Global implementation does not require the calling application to select the PIV smart card application (pivSelectCardApplication()) before it calls this function.

Syntax

```
PIV_RETURN pivPutData(
PIV_CARDHANDLE cardHandle,
PIV_Byte *OID,
PIV_ULong32 oidLength,
PIV_Byte *data,
PIV_ULong32 dataLength)
```

Parameters

cardHandle

[in] The communication handle returned by a successful call to the pivConnect() function.

OID

[in] Pointer to a null-terminated string containing the ASN.1 identifier of the object whose data content is to be replaced. Table 6 of NIST [SP 800-73] publication lists supported identifiers. See ASN.1 OID Identifiers on page 11 for a list of supported identifiers. Any other ASN.1 OID is rejected and the function fails.

If the object identified by the OID is valid according to the standard (defined in table 6), but is not present in the smart card, this function fails. This parameter should never be NULL.

oidLength

[in] Unsigned long containing the OID buffer size. This parameter should never be NULL.

data

[in] Pointer to a buffer containing the data to be used to replace the data content of the requested object. This parameter should never be NULL.

dataLength

[in] Unsigned long containing the data buffer size. This parameter should never be NULL.

Returns

PIV_OK

An existing smart card handle is returned. Return code from smart card: 0x90 0x00

PIV_INVALID_CARD_HANDLE

Function was called with an invalid card handle. Return code from PC/SC: SCARD_E_INVALID_HANDLE

PIV_CARD_READER_ERROR

- The reader is not connected to the local host, or
- No card is in the reader, or
- An event occurred that prevented communication with the card.

The PC/SC error codes that are mapped into PIV_CARD_READER_ERROR in the pivPutData() implementation:

SCARD_F_COMM_ERROR
SCARD_W_UNPOWERED_CARD
SCARD_W_RESET_CARD
SCARD_W_REMOVED_CARD
SCARD_E_READER_UNAVAILABLE

PIV_INVALID_OID

The buffer pointed to by the OID parameter contains an OID unsupported by ASN.1 OID (see Table 6 of NIST [SP 800-73] publication or ASN.1 OID identifiers, above).

PIV_SECURITY_CONDITIONS_NOT_SATISFIED

The sign operation is protected but the application security status was not been established by a successful call to pivLogIntoCardApplication(). This error code is also returned if the PIN is locked. Return codes from smart card: 0x69 0x82 or 0x69 0x83

PIV_INSUFFICIENT_CARD_RESOURCE

The smart card does not have enough resources to process the operation. Return code from smart card: 0x6A 0x84

7.0 Return Codes

The PIV API translates error codes returned by the PC/SC Microsoft API into PIV API error codes.

Return Code	Description
PIV_OK	Successful return code.
PIV_AUTHENTICATION_FAILURE	Authentication failed.
PIV_AUTHENTICATOR_MALFORMED	The authenticator's data objects to be used by the `pivLogIntoCardApplication()` function do not conform to the expected data format (Table 10 of [SP 800-73]).
PIV_CARD_APPLICATION_NOT_FOUND	Application AID does not refer to an AID that the card supports.
PIV_CARD_READER_ERROR	An unexpected APDU response or an unexpected response from the PC/SC layer was received. This usually happens when, for some reason, an event occurred that prevented communication with the card.
PIV_CONNECTION_DESCRIPTION_MALFORMED	The `ConnectionDescription` parameter in the function call does not conform to the required data format or the PC/SC returned one of the following error codes: • SCARD_E_UNKNOWN_READER • SCARD_E_DUPLICATE_READER
PIV_CONNECTION_FAILURE	There is no smart card in the reader or the smart card returned an unexpected answer or no answer at all or the PC/SC service is not started.
PIV_CONNECTION_LOCKED	An SCARD_E_SHARING_VIOLATION error was returned by PC/SC. Either: • A shared connection exists but an exclusive connection was requested, or • An exclusive connection exists but a shared/exclusive connection was requested.
PIV_INPUT_BYTES_MALFORMED	Incorrect input data format (Table 17 of [SP 800-73]).

Return Code	Description
PIV_INSUFFICIENT_BUFFER	Supplied output buffer too small.
PIV_INVALID_CARD_HANDLE	The communication handle is invalid. This generally happens when a function was called with an invalid card handle or when the smart card has been removed from the reader. The error corresponds to SCARD_E_INVALID_HANDLE from PC/SC.
PIV_INVALID_KEY_OR_KEYALG_COMBINATION	Inconsistency in `pivGenerateKeyPair()` input parameters.
PIV_INVALID_KEYREF_OR_ALGORITHM	Unsupported key reference value or unsupported cryptographic mechanism for a general authenticate operation, either by the standard (Table 12 of [SP 800-73] and http://piv.nist.gov/pivqa/faq.php?qid=122) or by the card.
PIV_INVALID_OID	Object identifier is not one defined in ASN.1 OID (Table 6 of [SP 800-73]).
PIV_SECURITY_CONDITIONS_NOT_SATISFIED	This error code occurs if you do not have the AR to perform the operation. For example, you need to perform an external authenticate to generate a key. It is not limited to the PIN AR.
PIV_UNSUPPORTED_CRYPTOGRAPHIC_MECHANISM	The cryptographic mechanism value is not one of the algorithm identifiers found in Table 7 of [SP 800-73] or supported by the card.

Appendix A: Terms and Acronyms

This appendix lists terms and acronyms used throughout the full set of the set of technical publications for this product. Not all terms and acronyms appear in all documents in the set.

A.1. Terms

Certificate Authority (CA)

The CA issues and manages security credentials and public keys for message encryption in a networked environment. As part of a Public Key Infrastructure (PKI), a CA checks with a registration authority (RA) to verify information provided by the requestor of a digital certificate. If the RA verifies the requestor's information, the CA issues a certificate.

ActivID Credential Management System (CMS)

Formally known as ActivID Card Management System, ActivID CMS is a web-based, smart card, credential and application lifecycle management system. ActivID CMS augments and works in concert with an enterprise's primary identity management infrastructure components, including popular directory, database, and PKI components.

Challenge

Random number generated by the server API for authentication of a user in the asynchronous (challenge/response) mode.

Cryptographic Service Provider (CSP)

An independent software module that performs cryptography algorithms for authentication, encoding, and encryption.

Discovery mode

Discovery mode enables a calling application to find out the size of the data that will be returned to by making a preliminary discovery call and then making a second call after it allocates a buffer large enough to accommodate the data that will be returned.

End-point card

The PIV standard defines two interfaces for communicating with PIV cards:

- The PIV transitional interface.
- The PIV end-point interface.

A PIV end-point card is a card that implements the second of these interfaces.

Note: The PIV transitional interface is not supported by the PIV API.

Federal Information Processing Standard (FIPS 140-2)

FIPS 140-2 is the standard for crypto-module security. FIPS 140-2 level 3 adds additional requirements to FIPS 140-2 level 2. These requirements concern physical security and a trusted path for entering a Cryptographic Service Provider, such as a PIN. FIPS 140-2 level 3 uses local ports and the key pad to enforce such security.

Federal Information Processing Standard 201 (FIPS 201)

FIPS 201 is the standard for Personal Identity Verification (PIV) cards defined for US Government employees and contractors.

Force change PIN flag

Flag which indicates whether the user must change the PIN on first use of the card.

Integrated circuit chip (ICC)	The chip on the smart card.
Mini Driver	Smart card middleware for the Microsoft platform that works with the Microsoft Base Smart Card CSP (Cryptographic Service Provider). The ActivClient Mini Driver replaces the ActivClient CSP available in previous versions. The Mini Driver architecture provides stronger cryptographic services.
My Digital ID Card (MDIDC)	This CMS component allows end users to access the self-service CMS functions, which includes card and credential management.
One-Time Password (OTP)	A one-time password is a password used only once to authenticate to remote applications. One-Time Passwords are only present on smart cards issued with SKI credentials.
Personal Identification Number (PIN)	The Personal Identification Number (PIN) code used to access an HID Global device's services such as Windows PKI logon, remote access and email signature. HID Global devices can only be used after a correct PIN is entered.
Public Key Infrastructure (PKI)	PKI describes the laws, policies, standards, and software that regulate or manipulate certificates and public and private keys.
Registration Authority (RA)	RA is an authority in a network that verifies user requests for a digital certificate and instructs the CA to issue it. An RA is part of a PKI, a networked system that enables companies and users to exchange information safely and securely.
Symmetric Key Infrastructure (SKI)	SKI keys are used to perform strong authentication on remote applications. SKI keys encrypt passwords in: • Synchronous mode (generates 1 password without any challenge. The server uses the same method to create a password than the smart card) • Asynchronous: encrypts a challenge
Standalone smart card	Smart card with pre-loaded applets issued by the manufacturer.
Unlock code	Value that the card holder needs to provide in order to unlock a locked smart card. Depending upon the smart card unlock mechanism, the unlock code may or may not be different from the unlock key.
Verification	Process in which a signature that was produced by the signing operation is verified.
Weak PIN	A weak PIN is a PIN in which: • The length is less than three characters or digits, or • The difference between each character or digit and the following one is a constant. For example, a PIN that is a sequence of the same number (1111) or an increasing/decreasing sequence of numbers (1234, 4321) is a weak PIN.

A.2. Acronyms

CA Certificate Authority

CAC Common Access Card (for the United States Department of Defense)

CSP Cryptographic Service Provider

CUID Card Unique Identifier

CUID is a number that uniquely identifies a card.

FIPS Federal Information Processing Standard

GAL Global Address List

GP GlobalPlatform

Replaces OpenPlatform (OP).

OTP One-Time Password

PKI Public Key Infrastructure

PIV Personal Identity Verification.

Smart card issued by the United States government to federal employees and contractors.

RA Registration Authority

SKI Symmetric Key Infrastructure

