Generic Identity Device Specification

Version 2.0

October 19, 2012

Abstract

This document specifies the profile (card edge and the data model) that can be supported by a physical identity device such as a smart card that fully complies with ISO/IEC 7816.

References and resources discussed here are listed at the end of this paper.

The current version of this paper is maintained on the Web at:   
 [Generic Identity Device Specification](http://msdn.microsoft.com/en-us/library/windows/hardware/gg487496.aspx)

For feedback or questions, send an e-mail to:  
 [gids@microsoft.com](mailto:gids@microsoft.com)

**Disclaimer**: This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet website references, may change without notice. Some information relates to pre-released product which may be substantially modified before it’s commercially released. Microsoft makes no warranties, express or implied, with respect to the information provided here. You bear the risk of using it.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes. You may modify this document for your internal, reference purposes.

© 2012 Microsoft. All rights reserved.

C:\Users\jenlin\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Outlook\KN5ONHWU\dep_MicrosoftLogotype.png

****Acknowledgment****

Microsoft would like to thank Christophe Goyet from Oberthur Technologies for the initial work on GICS that he led within INCITS B10.12, from which this GIDS specification was derived.

Document History

|  |  |
| --- | --- |
| **Date** | **Change** |
| October 19, 2012 | Second publication – Version 2 |
| September 7, 2010 | Fix for bugs |
| December 4, 2009 | First publication – Version 1 |

Contents

[1 Introduction 10](#_Toc338334125)

[1.1 Scope 10](#_Toc338334126)

[1.2 Expected Benefits 10](#_Toc338334127)

[1.3 Design Approaches 10](#_Toc338334128)

[1.4 What’s New in Version 2.0 11](#_Toc338334129)

[1.4.1 Support for import of unencrypted key data 11](#_Toc338334130)

[1.4.2 Support for challenge/response retry counter reset for Application PIN. 11](#_Toc338334131)

[1.4.3 Support for external authentication using symmetric algorithm 11](#_Toc338334132)

[2 Identity Device Structures for Applications and Data 12](#_Toc338334133)

[2.1 Identity Device Structure 12](#_Toc338334134)

[2.2 MF Content 12](#_Toc338334135)

[2.2.1 EF.ATR 12](#_Toc338334136)

[2.2.2 EF.DIR 13](#_Toc338334137)

[2.2.3 CCD 13](#_Toc338334138)

[2.2.4 ACD 14](#_Toc338334139)

[2.3 Data Objects Organization 14](#_Toc338334140)

[2.4 Identity Device Elementary Files 14](#_Toc338334141)

[2.4.1 Data Object Elementary Files 14](#_Toc338334142)

[2.4.2 Key Elementary Files 15](#_Toc338334143)

[2.5 Security Architecture 15](#_Toc338334144)

[2.5.1 Security Attributes 15](#_Toc338334145)

[2.5.2 Access Mode Bytes 16](#_Toc338334146)

[2.5.3 Security Condition Byte 17](#_Toc338334147)

[2.6 GIDS Metadata 19](#_Toc338334148)

[2.6.1 Application Template Data Object (FCI) 19](#_Toc338334149)

[2.6.2 FCP Templates 20](#_Toc338334150)

[2.6.3 FMD Template 23](#_Toc338334151)

[2.6.4 Memory Resource Assignment Data Objects 24](#_Toc338334152)

[2.7 GIDS System Data Objects 24](#_Toc338334153)

[2.7.1 System Data Object Retrieved with GET DATA 24](#_Toc338334154)

[3 Addressing Data Structures 25](#_Toc338334155)

[3.1 Introduction 25](#_Toc338334156)

[3.2 P1-P2 Parameters in the GET DATA and PUT DATA Commands 25](#_Toc338334157)

[3.3 Data Handling Tags 25](#_Toc338334158)

[3.3.1 Tag List 5C 25](#_Toc338334159)

[4 Data Object Management 27](#_Toc338334160)

[4.1 GIDS Data Objects 27](#_Toc338334161)

[4.1.1 Self-Contained BER Data Object 27](#_Toc338334162)

[4.2 Data Object Management 27](#_Toc338334163)

[4.2.1 SC DO 27](#_Toc338334164)

[4.3 Atomicity of DO Management Operations 28](#_Toc338334165)

[4.4 Coordinated DO Management Operations 28](#_Toc338334166)

[5 Discovery of Applications 29](#_Toc338334167)

[5.1 Applications Discovery 29](#_Toc338334168)

[5.2 PIN Usage Policy Discovery 29](#_Toc338334169)

[5.3 Cryptographic Capabilities Discovery 29](#_Toc338334170)

[5.4 File Structure Discovery 29](#_Toc338334171)

[5.5 Current Elementary File Identification 30](#_Toc338334172)

[5.6 DO List at DF Level 30](#_Toc338334173)

[5.7 DO List at EF Level 30](#_Toc338334174)

[5.8 DO Values at DF Level 30](#_Toc338334175)

[5.9 DO Values at EF Level 30](#_Toc338334176)

[6 GIDS Life Cycle Management 31](#_Toc338334177)

[6.1 Creation State 31](#_Toc338334178)

[6.1.1 Application Creation State 31](#_Toc338334179)

[6.2 Initialization State 32](#_Toc338334180)

[6.2.1 Application Initialization State 32](#_Toc338334181)

[6.2.2 EF Initialization State 32](#_Toc338334182)

[6.3 Operational State 32](#_Toc338334183)

[6.3.1 Operational State—Activated 32](#_Toc338334184)

[6.3.2 Operational State—Deactivated 32](#_Toc338334185)

[6.4 Termination State 33](#_Toc338334186)

[6.4.1 Application Termination State 33](#_Toc338334187)

[6.4.2 EF Termination State 33](#_Toc338334188)

[7 CHV Management 34](#_Toc338334189)

[7.1 PIN Usage Policy 34](#_Toc338334190)

[7.2 Application PIN Management 34](#_Toc338334191)

[7.3 Global PIN Management 35](#_Toc338334192)

[7.4 User Authentication ALWAYS and Key Usage Counter 35](#_Toc338334193)

[7.5 CHV Status 36](#_Toc338334194)

[7.5.1 Global PIN Status 36](#_Toc338334195)

[7.5.2 Local PIN Status 36](#_Toc338334196)

[7.5.3 Local PUK Status 36](#_Toc338334197)

[8 Cryptopgraphic Algorithms 37](#_Toc338334198)

[8.1 Control Reference Template (CRT) 37](#_Toc338334199)

[8.2 Cryptographic Mechanism References 37](#_Toc338334200)

[8.2.1 Authentication 38](#_Toc338334201)

[8.2.2 Confidentiality 38](#_Toc338334202)

[8.2.3 Digital Signature 39](#_Toc338334203)

[9 Authentication and Session Key Agreement Protocols 41](#_Toc338334204)

[9.1 Mutual Authentication with Symmetric Algorithm 41](#_Toc338334205)

[9.2 External Authentication with Symmetric Algorithm 42](#_Toc338334206)

[9.3 Key Establishment with Internal Authentication Using ECC 43](#_Toc338334207)

[10 Key Management 46](#_Toc338334208)

[10.1 Key Selection 46](#_Toc338334209)

[10.2 Reserved Key References 46](#_Toc338334210)

[10.3 Administrative Key 46](#_Toc338334211)

[11 APDU References 47](#_Toc338334212)

[11.1 Command Response Pairs 47](#_Toc338334213)

[11.2 CLASS Byte Coding 47](#_Toc338334214)

[11.3 Data Fields 47](#_Toc338334215)

[11.4 Status Bytes SW1 and SW2 47](#_Toc338334216)

[11.4.1 General Meaning 47](#_Toc338334217)

[11.4.2 Specific Interindustry Warning and Error Conditions 48](#_Toc338334218)

[11.4.3 Status Word Treatment for Interoperability 49](#_Toc338334219)

[11.5 Command Chaining 49](#_Toc338334220)

[11.5.1 GIDS Commands Supporting Command Chaining 49](#_Toc338334221)

[11.5.2 Description of the Command Chaining 49](#_Toc338334222)

[11.5.3 Use of Command Chaining 50](#_Toc338334223)

[12 GIDS Command Set APDU 52](#_Toc338334224)

[12.1 ACTIVATE FILE 52](#_Toc338334225)

[12.1.1 Description 52](#_Toc338334226)

[12.1.2 Command APDU 53](#_Toc338334227)

[12.1.3 Status Word 53](#_Toc338334228)

[12.1.4 Conditional Usage 53](#_Toc338334229)

[12.2 CREATE FILE 54](#_Toc338334230)

[12.2.1 Description 54](#_Toc338334231)

[12.2.2 Command APDU 54](#_Toc338334232)

[12.2.3 Command Data Field 54](#_Toc338334233)

[12.2.4 Response Data Field 54](#_Toc338334234)

[12.2.5 Status Word 55](#_Toc338334235)

[12.2.6 Conditional Usage 55](#_Toc338334236)

[12.3 CHANGE REFERENCE DATA 55](#_Toc338334237)

[12.3.1 Description 55](#_Toc338334238)

[12.3.2 Command APDU 56](#_Toc338334239)

[12.3.3 P2 Parameter 56](#_Toc338334240)

[12.3.4 Command Data Field 56](#_Toc338334241)

[12.3.5 Response Data Field 56](#_Toc338334242)

[12.3.6 Status Word 56](#_Toc338334243)

[12.3.7 Conditional Usage 56](#_Toc338334244)

[12.4 DELETE FILE 57](#_Toc338334245)

[12.4.1 Description 57](#_Toc338334246)

[12.4.2 Command APDU 57](#_Toc338334247)

[12.4.3 Status Word 57](#_Toc338334248)

[12.4.4 Conditional Usage 57](#_Toc338334249)

[12.5 GENERATE ASYMMETRIC KEY PAIR 57](#_Toc338334250)

[12.5.1 Description 57](#_Toc338334251)

[12.5.2 Command APDU 58](#_Toc338334252)

[12.5.3 Command Data field 58](#_Toc338334253)

[12.5.4 Response Data Field 59](#_Toc338334254)

[12.5.5 Status Word 59](#_Toc338334255)

[12.5.6 Conditional Usage 59](#_Toc338334256)

[12.6 GENERAL AUTHENTICATE 59](#_Toc338334257)

[12.6.1 Description 59](#_Toc338334258)

[12.6.2 Command APDU 60](#_Toc338334259)

[12.6.3 P1-P2 Parameters 60](#_Toc338334260)

[12.6.4 Command Data Field 60](#_Toc338334261)

[12.6.5 Response Data Field 61](#_Toc338334262)

[12.6.6 Status Word 61](#_Toc338334263)

[12.6.7 Conditional Usage 62](#_Toc338334264)

[12.7 GET DATA 62](#_Toc338334265)

[12.7.1 Description 62](#_Toc338334266)

[12.7.2 Command APDU 62](#_Toc338334267)

[12.7.3 P1-P2 Parameters 63](#_Toc338334268)

[12.7.4 Command Data Field 63](#_Toc338334269)

[12.7.5 Response Data Field for BER-TLV Data Object 63](#_Toc338334270)

[12.7.6 Status Word 63](#_Toc338334271)

[12.7.7 Conditional Usage 63](#_Toc338334272)

[12.8 GET PUBLIC KEY 64](#_Toc338334273)

[12.8.1 Description 64](#_Toc338334274)

[12.8.2 Command APDU 64](#_Toc338334275)

[12.8.3 Command Data Field 64](#_Toc338334276)

[12.8.4 Response Data Field 65](#_Toc338334277)

[12.8.5 Status Word 65](#_Toc338334278)

[12.8.6 Conditional Usage 65](#_Toc338334279)

[12.9 INTERNAL AUTHENTICATE 65](#_Toc338334280)

[12.9.1 Description 65](#_Toc338334281)

[12.9.2 Command APDU 65](#_Toc338334282)

[12.9.3 Command Data Field 65](#_Toc338334283)

[12.9.4 Response Data Field 66](#_Toc338334284)

[12.9.5 Status Word 66](#_Toc338334285)

[12.9.6 Conditional Usage 67](#_Toc338334286)

[12.10 MANAGE SECURITY ENVIRONMENT 67](#_Toc338334287)

[12.10.1 Description 67](#_Toc338334288)

[12.10.2 Command APDU 67](#_Toc338334289)

[12.10.3 P1Parameter 67](#_Toc338334290)

[12.10.4 P2 Parameter 67](#_Toc338334291)

[12.10.5 Command Data Field 67](#_Toc338334292)

[12.10.6 Status Word 68](#_Toc338334293)

[12.10.7 Conditional Usage 68](#_Toc338334294)

[12.11 PERFORM SECURITY OPERATION 68](#_Toc338334295)

[12.11.1 Description 68](#_Toc338334296)

[12.11.2 Command APDU 68](#_Toc338334297)

[12.11.3 P1-P2 Parameters 69](#_Toc338334298)

[12.11.4 Command Data Field 69](#_Toc338334299)

[12.11.5 Status Word 69](#_Toc338334300)

[12.11.6 Conditional Usage 69](#_Toc338334301)

[12.12 PUT DATA 70](#_Toc338334302)

[12.12.1 Description 70](#_Toc338334303)

[12.12.2 Command APDU 70](#_Toc338334304)

[12.12.3 P1-P2 Parameters 70](#_Toc338334305)

[12.12.4 Command Data Field 70](#_Toc338334306)

[12.12.5 Response Data Field 70](#_Toc338334307)

[12.12.6 Status Word 70](#_Toc338334308)

[12.12.7 Conditional Usage 71](#_Toc338334309)

[12.13 PUT KEY 72](#_Toc338334310)

[12.13.1 Key Usage Template 72](#_Toc338334311)

[12.13.2 Key Value Templates 72](#_Toc338334312)

[12.14 RESET RETRY COUNTER 74](#_Toc338334313)

[12.14.1 Description 74](#_Toc338334314)

[12.14.2 Command APDU 74](#_Toc338334315)

[12.14.3 Command APDU when External or Mutual Authentication with an Administrative Key is used as authentication method 75](#_Toc338334316)

[12.14.4 Command Data Field 75](#_Toc338334317)

[12.14.5 Status Word 75](#_Toc338334318)

[12.14.6 Conditional Usage 75](#_Toc338334319)

[12.15 SELECT 76](#_Toc338334320)

[12.15.1 Description 76](#_Toc338334321)

[12.15.2 Command APDU 76](#_Toc338334322)

[12.15.3 P1 Parameter 76](#_Toc338334323)

[12.15.4 P2 Parameter 76](#_Toc338334324)

[12.15.5 Command Data Field 77](#_Toc338334325)

[12.15.6 Response Data Field 77](#_Toc338334326)

[12.15.7 Status Word 77](#_Toc338334327)

[12.15.8 Conditional Usage 77](#_Toc338334328)

[12.15.9 Channel Selection 78](#_Toc338334329)

[12.16 TERMINATE DF 78](#_Toc338334330)

[12.16.1 Description 78](#_Toc338334331)

[12.16.2 Command APDU 79](#_Toc338334332)

[12.16.3 Status Word 79](#_Toc338334333)

[12.16.4 Conditional Usage 79](#_Toc338334334)

[12.17 VERIFY 79](#_Toc338334335)

[12.17.1 Description 79](#_Toc338334336)

[12.17.2 Command APDU 80](#_Toc338334337)

[12.17.3 P2 Parameter 80](#_Toc338334338)

[12.17.4 Application Security Status Resetting Code 80](#_Toc338334339)

[12.17.5 Command Data Field 80](#_Toc338334340)

[12.17.6 Response Data Field 80](#_Toc338334341)

[12.17.7 Status Word 80](#_Toc338334342)

[12.17.8 Conditional Usage 81](#_Toc338334343)

[13 APDU Mapping 82](#_Toc338334344)

[13.1 Authentication Mechanisms 82](#_Toc338334345)

[13.1.1 Mutual Authentication with Symmetric Algorithm 82](#_Toc338334346)

[13.1.2 External Authentication with Symmetric Algorithm 82](#_Toc338334347)

[13.1.3 Key Establishment with Internal Authentication Using ECC 83](#_Toc338334348)

[14 Transport Protocol Management 84](#_Toc338334349)

[14.1 Communication Interface and Supported Protocols 84](#_Toc338334350)

[14.2 Extended Length 84](#_Toc338334351)

[14.3 Sending More than 255 bytes to the ICC 84](#_Toc338334352)

[14.3.1 Use of Extended Length for Incoming Data 84](#_Toc338334353)

[14.3.2 Use of Command Chaining for Incoming Data 84](#_Toc338334354)

[14.4 Command Returning More Than 256 Bytes 84](#_Toc338334355)

[14.4.1 Case 1 85](#_Toc338334356)

[14.4.2 Case 2 85](#_Toc338334357)

[14.4.3 Case 3 86](#_Toc338334358)

[14.4.4 GET RESPONSE 86](#_Toc338334359)

[15 Appendix A: Technical Limitations 88](#_Toc338334360)

[15.1 Technical Minima 88](#_Toc338334361)

[15.2 Compliance with ISO 7816 88](#_Toc338334362)

[15.2.1 Discrepancies in APDU Behavior 88](#_Toc338334363)

[15.2.2 Tag Created by GIDS 88](#_Toc338334364)

[16 Appendix B: Definitions and Acronyms 89](#_Toc338334365)

[16.1 Definitions 89](#_Toc338334366)

[16.2 Acronyms 89](#_Toc338334367)

[16.3 Notation 89](#_Toc338334368)

[17 Appendix C: References 90](#_Toc338334369)

Tables

[Document History 3](#_Toc329857738)

[Table 1: EF ATR Data Object 11](#_Toc329857739)

[Table 2: Security Attribute in Compact Form 14](#_Toc329857740)

[Table 3: Access Mode Bytes for Application Data Objects 14](#_Toc329857741)

[Table 4: Access Mode Bytes for Keys 15](#_Toc329857742)

[Table 5: Access Mode Byte for ADF 15](#_Toc329857743)

[Table 6: Security Condition Byte 16](#_Toc329857744)

[Table 7: Application Template Assignment Data Objects 17](#_Toc329857745)

[Table 8: Discretionary Data Objects 17](#_Toc329857746)

[Table 9: Supported authentication and key establishment protocols 18](#_Toc329857747)

[Table 10: FCP Template Assignment Data Objects for DF 18](#_Toc329857748)

[Table 11: Cryptographic Mechanism Identifier Template 19](#_Toc329857749)

[Table 12: FCP Template Assignment Data Objects for Data Object and Binary EF 19](#_Toc329857750)

[Table 13: FCP Template Assignment Data Objects for Key EF with CRT support 20](#_Toc329857751)

[Table 14: CRT from EF FCP 20](#_Toc329857752)

[Table 15: FMD Template Assignment Data Objects for the ADF 21](#_Toc329857753)

[Table 16: FMD Template Assignment Data Objects for the ADF 21](#_Toc329857754)

[Table 17: System Data Object Retrieved with GET DATA 22](#_Toc329857755)

[Table 18: Life Cycle Status Byte 29](#_Toc329857756)

[Table 19: First Byte of PIN Usage Policy 31](#_Toc329857757)

[Table 20: Global PIN Status 33](#_Toc329857758)

[Table 21: Local PIN Status 33](#_Toc329857759)

[Table 22: Local PUK Status 33](#_Toc329857760)

[Table 23: CRT Tags 34](#_Toc329857761)

[Table 24: Usage Qualifier Byte 34](#_Toc329857762)

[Table 25: Cryptographic Mechanism Reference for AT CRT 35](#_Toc329857763)

[Table 26: Cryptographic Mechanism Reference for CT CRT with Symmetric Algorithm 35](#_Toc329857764)

[Table 27: Cryptographic Mechanism Reference for CT CRT with Asymmetric Algorithm 35](#_Toc329857765)

[Table 28: Cryptographic Mechanism Reference for DST CRT 36](#_Toc329857766)

[Table 29: DigestInfo Values for Hash Functions (from RFC 3447) 37](#_Toc329857767)

[Table 30: Length of Pre-master Secret Z as a Function of Authentication Key Used 39](#_Toc329857768)

[Table 31: C1,1 ECC CDH Protocol for Key Establishment with Internal Authentication 41](#_Toc329857769)

[Table 32: Key Reference 43](#_Toc329857770)

[Table 33: CLASS Byte Interindustry Values 44](#_Toc329857771)

[Table 34: General Meaning of the Interindustry Values of SW1-SW2 44](#_Toc329857772)

[Table 35: Specific Interindustry Warning and Error Conditions 45](#_Toc329857773)

[Table 36: GIDS APDUs 48](#_Toc329857774)

[Table 37: ACTIVATE FILE APDU 49](#_Toc329857775)

[Table 38: CREATE FILE APDU 50](#_Toc329857776)

[Table 39: CHANGE REFERENCE DATA APDU 52](#_Toc329857777)

[Table 40: Reference Data ID for CHANGE REFERENCE DATA P2 Parameter 52](#_Toc329857778)

[Table 41: DELETE FILE APDU 53](#_Toc329857779)

[Table 42: GENERATE ASYMMETRIC KEY PAIR Data Field 54](#_Toc329857780)

[Table 43: Cryptographic Mechanism Identifiers for Generation of an Asymmetric Key with CRT 54](#_Toc329857781)

[Table 44: GENERATE ASYMMETRIC KEY PAIR Response Field for RSA Keys 55](#_Toc329857782)

[Table 45: GENERATE ASYMMETRIC KEY PAIR Response Field for ECC Keys 55](#_Toc329857783)

[Table 46: GENERAL AUTHENTICATE APDU 56](#_Toc329857784)

[Table 47: Data Field of the GENERAL AUTHENTICATE APDU 56](#_Toc329857785)

[Table 48: GET DATA APDU 58](#_Toc329857786)

[Table 49: GET DATA Command Data Field 59](#_Toc329857787)

[Table 50: Response Data Field for BER-TLV Data Object 59](#_Toc329857788)

[Table 51: Command Data field to Retrieve a Public Key Value 60](#_Toc329857789)

[Table 52: Response Data Field with Public Key Value 60](#_Toc329857790)

[Table 53: INTERNAL AUTHENTICATE APDU 61](#_Toc329857791)

[Table 54: Command Data Field of the INTERNAL AUTHENTICATE APDU 61](#_Toc329857792)

[Table 55: Response Data Field of the INTERNAL AUTHENTICATE APDU 62](#_Toc329857793)

[Table 56: INTERNAL AUTHENTICATE APDU 63](#_Toc329857794)

[Table 57: MSE P1 Parameter 63](#_Toc329857795)

[Table 58: Control Reference Data Objects in Control Reference Template for MSE 63](#_Toc329857796)

[Table 59: PSO APDU 64](#_Toc329857797)

[Table 60: PSO P1-P2 Parameters 65](#_Toc329857798)

[Table 61: PUT DATA APDU 66](#_Toc329857799)

[Table 62: Data field of PUT DATA Command for a BER-TLV data object 66](#_Toc329857800)

[Table 63: Data Field of PUT DATA Command to Load Keys 68](#_Toc329857801)

[Table 64: Key Usage Template 68](#_Toc329857802)

[Table 65: Key Value Template for PKCS#11 Key Import 68](#_Toc329857803)

[Table 66: RESET RETRY COUNTER APDU 70](#_Toc329857804)

[Table 67: RESET RETRY COUNTER APDU FOR EXTERNAL OR MUTUAL AUTHENTICATION WITH AN ADMINISTRATIVE KEY 71](#_Toc329857805)

[Table 68: SELECT APDU 72](#_Toc329857806)

[Table 69: SELECT P1 Parameter 72](#_Toc329857807)

[Table 70: SELECT P2 Parameter 72](#_Toc329857808)

[Table 71: TERMINATE DF APDU 74](#_Toc329857809)

[Table 72: VERIFY APDU 75](#_Toc329857810)

[Table 73: Reference Data Qualifier 75](#_Toc329857811)

[Table 74: Mutual Authentication with Session Key Agreement Using Symmetric Algorithm 78](#_Toc329857812)

[Table 75: External Authentication Using Symmetric Algorithm 79](#_Toc329857813)

[Table 76: Internal Authentication with Session Key Agreement Using ECC Algorithm 79](#_Toc329857814)

Figures

[Figure 1: Mutual authenticate mechanism with key establishment using symmetric algorithm 38](#_Toc329857815)

[Figure 2: External authenticate mechanism using symmetric algorithm 39](#_Toc329857816)

[Figure 3: Establishment with internal authentication using elliptic curve cryptography 41](#_Toc329857817)

# Introduction

This document specifies the profile (card edge and the data model) that can be supported by a physical identity device such as a smart card that fully complies with ISO/IEC 7816. Throughout the rest of this document, this specification is referred to as GIDS (Generic Identity Device Specification). The intent of this part is to provide a minimum generic identity command set that enables implementers to use a subset of ISO/IEC 7816 application protocol data unit (APDU) commands in a consistent manner for interaction with smart cards that are used for identity applications, from card personalization to the use phase. Other parts of this specification that cover additional commands relevant to the physical card device are not defined in this first part (such as in ISO/IEC 7816-9 to ISO/IEC 7816-4).

## Scope

The proposed smart card command set shall:

* Employ a common data object (DO) access methodology.
* Include a common set of administrative commands.
* Provide a predictable means for expanding the application data model.
* Enable a means to extend functionality by using ISO/IEC 7816 commands.
* Incorporate discovery mechanisms.

## Expected Benefits

A well-defined application profile is likely to boost the development of the identity market for smart cards for the following reasons:

* Card manufacturers and third-party application developers will be more inclined to develop and use a command set that is commonly available to multiple applications.
* Cards that implement this command set would provide better interoperability in some situations without requiring a translation mechanism that could be costly in terms of overhead, security, and performance.
* A stable command set could be hard coded (ROM) to further improve performance and reduce the end product cost.
* Personalization/card management systems suppliers will be able to offer their customers a system that is compatible with a broader range of card suppliers and minimizes customization.
* Customers will have an easier-to-manage multisource supply of cards, which allows them to mix suppliers based on quality of products and services.

## Design Approaches

This specification does not mandate the use of a particular technology (such as native, Java Card, Multos, and .NET).

## What’s New in Version 2.0

### Support for import of unencrypted key data

The PUT KEY command now supports importing of unencrypted key data. To perform this operation, specify 0x00 as the transport key (tag 84) in the Key Value Template.

### Support for challenge/response retry counter reset for Application PIN.

The RESET RETRY COUNTER now supports resetting the Application PIN when External or Mutual authentication using an Administrator key is used. When this method is being used, the PUK does not exist and any GET DATA requests for the Local PUK Status object must return 6A 88 (Reference Data Not Found).

### Support for external authentication using symmetric algorithm

The GENERAL AUTHENTICATE command now supports external authentication using a symmetric algorithm. This is different than the “Mutual Authentication with Symmetric Algorithm” protocol since it does not require a random value to be generated off-card, uses a shorter challenge length, and does not generate a pre-master secret that may be subsequently used to open secure messaging.

This is performed by sending an empty challenge (tag 81) in the body of the data field. See “External Authentication with Symmetric Algorithm” for more detailed information on this new authentication protocol. If the card application supports this protocol, it must specify this in its Application Template Data Object (FCI) discretionary data objects. An example can be found in “Authentication Mechanisms.”

# Identity Device Structures for Applications and Data

## Identity Device Structure

The command set that is specified in this document is DO-oriented and uses the structures that are defined by ISO/IEC 7816 to select, access, protect, and store data that is related to an application. The goal of the specification is to provide a carefully selected set of APDUs that identity applications can use to manage their own data models in full compliance with BER-TLV and ISO/IEC 7816 parts 4, 8, 9, 11, and 13.

Each card application is identified by its application identifier (AID) that maps into an application dedicated file (ADF) that is based on an identity device command set. Application DOs are stored in elementary files (EFs) under the ADF. Files are readable only by using an odd INS byte form of the GET DATA command. A card that supports the identity device command set has at least one and possibly several ADFs. Only one level of file is allowed in any application, that is, an ADF can contain only EFs and no sub-DFs. ADFs do not have file IDs and can be selected only by name (which can be discovered by using EF.DIR).

## MF Content

This specification does not require an MF. However, the following files and data structures (usually stored in an MF) are always accessible from all card applications regardless of whether the command set implements an MF:

* EF.ATR
* EF.DIR
* Card capability description (CCD)
* Application capability description (ACD)

The foregoing data are always freely retrievable from any ADF by using GET DATA. Creation of the MF and its direct content (EF.ATR, EF.DIR, CCD, and ACD, as well as one ADF for each application) is out of scope of this part of this specification.

Personalization capabilities that are offered by GIDS part 1 starts from the application creation state and are defined in “GIDS Life Cycle Management.”

### EF.ATR

This specification supports the ISO EF.ATR file that indicates operating characteristics of the card. Its content is listed in “Table 1: EF ATR Data Object.”

The contents of the EF.ATR can be freely retrieved by using a GET DATA command with P1 P2 = 2F01 and a command data field of 5C00. The EF.ATR is a global data, and the response data field of the foregoing command contains the BER-TLV content of the EF.ATR regardless of the currently selected EF/DF.

Retrieving the EF.ATR content with this method does not deselect the currently selected EF.

The contents of the EF.ATR shall be BER-TLV encoded as per ISO 7816-4:2005 section 8.2.1.1.

The following table defines the minimum content for the EF.ATR file.

Table : EF ATR Data Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **Len** | **What** | **Value** | **Meaning** |
| **43** | **01** | **Card service data tag** |  | **Tag for next byte** |
|  | | Card service data byte F4 | F4 | b8=1: Application selection by full DF name  b7=1: Application selection by partial name  b6=1 BER-TLV DO present in EF.DIR  b5=1 BER-TLV DO present in EF.ATR  b4=0 READ BINARY not available to access EF.ATR and EF.DIR  b3=1: EF.DIR and EF.ATR access service by GET DATA command (TLV structure)  b2=0 READ RECORD not available to access EF.ATR and EF.DIR  b1 = 0: Card with MF |
| **47** | **03** | **Card capabilities tag** |  | **Tag for next 3 bytes** |
|  | | Card capabilities data byte 1 🡪 selection method | 08 | b4=1: Implicit DF selection |
| Card capabilities data byte 2 🡪 data coding byte method | 01 | b4...b1 = 0001: data unit size is 1 byte |
| Card capabilities data byte 3 🡪Miscellaneous | CC | b8=1: Command chaining is supported  b7=1: Extended Lc and Le fields are supported  Chaining is supported  b4 =1 : Logical channel number assignment by the interface device  b3 b2 b1: 100: Maximum number of channels supported: 4 |
| **46** | **va** | **Pre-issuing DO** |  | **DO Information in ASCII to identify the card manufacturer and the product** |

### EF.DIR

GIDS supports the ISO EF.DIR file that lists the AID of all identity device ADFs in the card. Its contents are constructed automatically by the identity device and updated whenever a new ADF is created. The termination of an ADF does not change the content of EF.DIR (see “GIDS Life Cycle Management”).

The content of EF.DIR can be freely retrieved by using a GET DATA command that has P1-P2 = 2F00 and a command data field of 5C00. EF.DIR is global data, and the response data field of this command contains the concatenation of all application templates, regardless of the currently selected EF/DF. See “Application Template Data Object (FCI).”

Retrieving EF.DIR content by using the foregoing method does not deselect the currently selected EF.

### CCD

To facilitate the integration of a card that supports the identity device command set with ISO/IEC 24727 middleware, a CCD DO (tag 7F62) is always retrievable from all card applications.

The CCD can be retrieved at any time and from any identity device ADF by using a GET DATA command with P1- P2 = 3F FF and a command data field that contains 5C027F62. The content of the CCD is defined by ISO/IEC 24727 and is outside the scope of this specification.

### ACD

To facilitate the integration of a card that supports the identity device command set with ISO/IEC 24727 middleware, an ACD DO (tag 7F63) is always retrievable from all card applications.

The ACD can be retrieved at any time with a GET DATA command with P1-P2 = 3F FF and a command data field that contains 5C027F63. The content of the ACD is defined by ISO/IEC 24727 and is outside the scope of this specification.

## Data Objects Organization

Application data is identified at the interface by using DO identifiers (BER-TLV encoded tags), which can retrieve and modify DOs by using only their reference. Primitive as well as constructed DOs are allowed.

Application DOs, with tags of up to 3 bytes, can be created by using the odd INS byte form of the PUT\_DATA command.

Application DOs are grouped by identical access rules into EFs. Files are the structures to which access control rules are attached. This specification allows more than one file that has identical access control rules.

The EF into which a DO is stored provides access control rules and context for context-dependent DOs.

The information that is related to a file is always available at the interface by using the response to the SELECT command (FCI, FCP, or FMD depending on command parameters). SELECT by parameters P1-P2 = 00 00 returns information on the currently selected EF, and SELECT with data field = 3F FF returns information on the current application (ADF).

A SELECT command with parameters P1-P2 = 00-00 (current EF) or P1-P2 = 3F-FF (current DF) does not modify any security status of the current DF/ EF or the card.

## Identity Device Elementary Files

DOs are organized by identity device into the following two types of EFs:

* Data object
* Key

### Data Object Elementary Files

DO EFs are used to store DOs that fully comply with BER-TLV encoding rules. Because such DOs strictly follow BER-TLV encoding rules, their content can be parsed and all data access methods that are described in an identity device can be exercised.

A DO EF can contain a mix of primitive and constructed DOs.

The content of a BER DO EF is a concatenation of DOs that share the same access control rules.

DO EFs support access only with GET DATA and PUT DATA. Access with any command other than GET DATA and PUT DATA shall fail with “Command incompatible with file structure” (6981).

DO EF behavior is entirely consistent with ISO/IEC 7816-4, and its ISO file descriptor in the FCP template is 39 (TLV structure for BER-TLV DOs).

### Key Elementary Files

The role of the key EF is to define management rules, including access conditions and key management properties according to a subset of ISO/IEC 7816-4 as described in “Cryptopgraphic Algorithms"

There is one key EF per symmetric key or asymmetric key pair. The FCP of the key file provides public information on the key algorithm and ID, as well as the key role (such as administration, authentication, or signature).

There are two types of key EFs, depending on whether the associated key supports the use of a cryptographic reference template (CRT). See “Control Reference Template (CRT).”

A key for which one or more CRTs shall be enforced by the card is associated with a key EF that is identified by a file descriptor byte set to 18 in the FCP template. This FCP template shall list the CRT that is supported by the key. See “Table 13: FCP Template Assignment Data Objects for Key EF with CRT support.”

From a data management perspective, as exposed in the APDUs, the key EF does *not* contain key values.

## Security Architecture

Access control to an object is managed by binding access rules to the related object.

In the identity device command set, the binding is achieved by organizing DOs (application data or keys) into EFs where all DOs that are stored in a given EF share the same access control rules.

### Security Attributes

The file control parameter (FCP) of a DF/EF provides the security attribute in compact format (tag 8C) that lists the various access rules that apply to the DOs in the file.

In compact format, an access rule consists of an access mode byte (AMB) followed by one or more security condition bytes (SCBs).

The AMB lists the commands that may be executed, and the SCB provides information on the conditions under which the command can be executed (access conditions). One SCB per command is listed as executable in the AMB.

If several access rules are present in the value field of a DO with tag 8C, they represent an OR condition. This allows defining different security conditions for contact and for contactless modes of communication.

The following table illustrates a typical security attribute.

Table : Security Attribute in Compact Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | **Description** | |
| 8C | Var. |  | Security attribute in compact form | |
|  |  | AMB | Access mode byte | Access rule #1 |
|  |  | *SCB* | Concatenation of security condition bytes  (one per bit [b7 to b1] set in the AMB) |
|  |  | *SCB* |
|  |  | *SCB* |
|  |  | AMB | Optional additional access mode byte | Access rule #2 |
|  |  | *SCB* | Concatenation of security condition bytes  (one per bit [b7 to b1] set in the AMB) |
|  |  | *SCB* |
|  |  | *SCB* |

**Note:** Access conditions are enforced only when the application (DF) life cycle state is OPERATIONAL. For access controls information outside the OPERATIONAL life cycle, see “GIDS Life Cycle Management.”

### Access Mode Bytes

The AMB lists the commands that may be executed. Because this list depends on the type of file, the format of the AMB is different depending on whether the EF contains application DOs or a key. There is also a specific AMB for DF.

Each bit 7 to 1 indicates either the absence of an SCB when set to 0 or the presence of an SCB in the same order (bit 7 to 1) when set to 1. See ISO/IEC 7816-4:2005 section 5.4.3.1.

If a listed command does not have its bit set in the AMB, it has no associated SCB and access conditions for that command are not defined in this access rule.

By default, none of the commands that apply to file content are executable. For a command to be executable, there must be at least one access rules that provides access condition to that command, except ACTIVATE DF, which is always free. See “GIDS Life Cycle Management.”

#### Access Mode Byte for Application Data Objects

Application DOs are stored exclusively in EFs. The AMD of the EF applies to all data that is stored in this EF. Supported access modes for application data are described in the following table, which is a subset from ISO 7816-4:2005 table 18.

Table : Access Mode Bytes for Application Data Objects

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **Meaning** |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | - | PUT DATA |
| 0 | 0 | 0 | 0 | 0 | 0 | - | 1 | GET DATA |

#### Access Mode Byte for Keys

Keys properties are stored in special files that are called key EFs. The AMB for a key uses the same structure as the AMB for application data except that bit 8 is set to 1 (bits 7 to 4 are proprietary) to allow support for key-specific APDUs with the use of bits b4 to b7.

The AMD for keys is shown in the following table.

Table : Access Mode Bytes for Keys

| **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **Meaning** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | - | - | - | - | - | - | - | Bits 3 to 1 according to ISO 7816-4:2005 table 18, and bits 7 to 4 proprietary |
| 1 | 0 | - | - | - | - | - | - | RFU |
| 1 | - | 0 | - | - | - | - | - | RFU |
| 1 | - | - | 0 | - | - | - | - | RFU |
| 1 | - | - | - | 1 | - | - | - | GENERATE ASYMMETRIC KEY PAIR |
| 1 | - | - | - | - | 1 | - | - | MANAGE SECURITY ENVIRONMENT |
| 1 | - | - | - | - | - | 1 | - | PUT DATA (PUT KEY see PUT KEY) |
| 1 | - | - | - | - | - | - | 1 | GET DATA (GET PUBLIC KEY see GET PUBLIC KEY) |

**Note:** The access conditions to use the key in a cryptographic process are the ones that are associated with the MANAGE SECURITY ENVIRONMENT command, regardless of whether an MSE SET APDU is sent or implicitly performed by the P1P2 parameters of the GENERAL AUTHENTICATE command. The FCP of the key EF where the key is stored provides further information on card holder verification (CHV) requirements (such as PIN ALWAYS). For details, see “User Authentication ALWAYS and Key Usage Counter.”

#### Access Mode Byte for DF

The following table provides the AMB for the ADF and is a subset of ISO.IEC 7816- 4:2005 table 16.

Table : Access Mode Byte for ADF

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **Meaning** |
| 0 | - | - | - | - | - | - | - | Bits 7 to 1 according to this table |
| 0 | 0 | - | - | - | - | - | - | DELETE FILE (self) not supported |
| 0 | - | 1 | - | - | - | - | - | Terminate DF |
| 0 | - | - | 0 | - | - | - | - | ACTIVATE FILE (always free) |
| 0 | - | - | - | 0 | - | - | - | DEACTIVATE FILE not supported |
| 0 | - | - | - | - | 0 | - | - | CREATE FILE (DF creation) not supported |
| 0 | - | - | - | - | - | 1 | - | CREATE FILE (EF creation) |
| 0 | - | - | - | - | - | - | 1 | DELETE FILE (Child) |

Terminate DF is used to set the application from life cycle “Operational” to “Terminated”. See “TERMINATE DF.” To create an EF file of type Key that will host the parameters for an ADMIN key, a mandatory external authenticate is added to the conditions that were set by the associated SCB. For details, see “Administrative Key.”

### Security Condition Byte

In an access rule, one SCB per command is listed as executable in the AMB. SCBs are sorted by commands in the AMB for which the corresponding bit is set, starting from the most significant bit. See ISO/IEC 7816-4:2005 section 5.4.3.1.

The SCB format is common to all files and is described in the following table from ISO 7816‑4:2005 table 20.

Table : Security Condition Byte

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | | **b2** | **b1** | **Meaning** |
| 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | No condition |
| 1 | 1 | 1 | 1 | 1 | | 1 | 1 | 1 | Never |
| - | - | - | - | 0 | | 0 | 0 | 0 | No reference to security environment |
| - | - | - | - | Not all equal | | | | | Security enviornment identifier (SEID) from 1 to 14 |
| - | - | - | - | 1 | | 1 | 1 | 1 | RFU |
| 0 | - | - | - | - | | - | - | - | At least one condition |
| 1 | - | - | - | - | | - | - | - | All conditions |
| ~~-~~ | 1 | - | - | - | | - | - | - | RFU |
| - | - | 1 | - | - | | - | - | - | External or Mutual Authentication |
| - | - | - | 1 | - | | - | - | - | User authentication |

External and Mutual authentication can be performed only by using an administrative key. For details, see “Administrative Key.”

GIDS supports one security environment for each mode of operation (contact versus contactless). They are static and cannot be managed by the application. They have the following SEIDs:

* SEID = 1 for contact operations
* SEID = 2 for contactless operations

When no SEID is referenced in the SCB, the SCB applies to all security environments (SEID1 and SEID2).

When SCB bits b7, b6, and b5 are all set to zero, the associated command is always available within the defined security environment referenced by b4 to b1.

Bit b8 of the SCB allows developers to specify that all listed conditions must be satisfied (such as SM and PIN). However, a single SCB does not allow a combination of AND and OR. If a combination of AND and OR is required, such as ([SM and PIN] or [SM and EXT AUTH]), multiple access rules would be required. As described in “Security Attributes” earlier in this specification, if several access rules are present they represent an OR condition. For interoperability purposes, GIDS supports a maximum of four access rules (AMBs) in the value field of the security attribute DO (tag 8C).

**Notes:**

* DO tags are stored together with the associated DO value in the EF and therefore are subject to the same read access condition as the DO value.
* Access conditions are enforced when the EF in which data is stored is in an operational state (active) unless the parent DF is still in a creation or initialization state. For details, see “GIDS Life Cycle Management.”
* To optimize performances, we recommend that the number of AMBs be minimized in the compact security attribute (tag 8C) of ADF and EF FCP. The recommended structure is to list AMB in the following order:

1. AMB/SCBs with SEID 0 (access rules apply to all interfaces)  
2. AMB/SCBs with SEID 1 (rules specific to contact)  
3. AMB/SCBs with SEID 2 (rules specific to contactless)

## GIDS Metadata

The DOs described in this section are meta-DOs that are managed by GIDS and can be retrieved only in the response data field of the SELECT command. Unless otherwise specified, they are always present.

Because their value is used to define card behavior, any value that is proposed at the interface shall be checked for validity and rejected if incorrect. See the error messages in “GET PUBLIC KEY” and “CREATE FILE.”

A card application can personalize the card with an application DO of the same tag to allow retrieval by using a GET DATA command. However, we do not recommend this because GIDS does not check the value of application DOs that are being loaded and the values that are transmitted with a PUT DATA may not match the actual behavior of the card or its life cycle.

### Application Template Data Object (FCI)

The application template allows retrieving the characteristics of the currently selected application (DF). It can be accessed as the FCI response to the SELECT DF APDU.

The application template is a strict subset of interindustry DOs in ISO/IEC 7816-4:2005 table 91 and is defined in the following table.

Table : Application Template Assignment Data Objects

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | | |
| 61 | Var. | **Application Template Data Object** | | |
|  | | **Tag** | **Len** | **Value** |
| 4F | Var. | Application AID |
| 50 | Var. | Application label (optional) |
| 5F50 | Var. | Uniform resource locator (optional) |
| 73 | Var. | Discretionary data objects (optional) |

Additional DOs may be present in this template to answer the specific need of a given application, but they are not mandatory for GIDS compliance.

#### Discretionary data objects

The following discretionary data objects are used to describe additional characteristics that are specific to GIDS application:

Table : Discretionary Data Objects

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | | |
| 73 | Var. | **Discretionary data objects (optional)** | | |
|  | | **Tag** | **Len** | **Value** |
| 40 | 01 | Supported authentication protocols |

Table : Supported authentication and key establishment protocols

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **Meaning** |
| 1 | - | - | - | - | - | - | - | Mutual Authentication with Symmetric Algorithm |
| - | 1 | - | - | - | - | - | - | External Authentication with Symmetric Algorithm |
| - | - | 1 | - | - | - | - | - | Key Establishment with Internal Authentication using ECC |
| - | - | - | 0 | 0 | 0 | 0 | 0 | Reserved for future use |

If the supported authentication protocols byte is not present, it should be assumed to have a value of 0xA0. For a detailed explanation of these protocols, see “.”

### FCP Templates

The FCP template is a set of file control parameters—that is, logical, structural, and security attributes that can be retrieved in the response data field of the SELECT command. The template is available for both DFs and EFs.

#### DF FCP

For ADF (currently selected application), the FCP template is constructed as shown in the following table.

Table : FCP Template Assignment Data Objects for DF

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | | |
| 62 | Var. | **FCP Template** | | |
|  | | **Tag** | **Len** | **Value** |
| 82 | 01 | File descriptor byte: 38 (“not shareable-DF”) |
| 8A | 01 | Life cycle status byte (see “Table 18: Life Cycle Status Byte”) |
| 84 | Up to 16 | DF name |
| 8C | Var. | Security attribute in compact format (see “Table 2: Security Attribute in Compact Form”) |
| AC | Var. | Cryptographic mechanism identifier template |
| … | … | … |
| AC | Var. | Cryptographic mechanism identifier template |

Additional DOs may be present in this template to answer the specific need of a given application, but they are not mandatory for GIDS compliance.

##### Cryptographic Mechanism Identifier Template

Referenced by tag AC, one or more cryptographic mechanism identifier templates may be present in the control parameters of the DF. See “Table 11: Cryptographic Mechanism Identifier Template.” Each one explicitly indicates the meaning of a cryptographic mechanism reference in the DF and its hierarchy. Such a template shall consist of two or more DOs:

* The first DO shall be a cryptographic mechanism reference (tag 80).
* The second DO shall be an object identifier (tag 06) as defined in ISO/IEC 8825-1. The identified object shall be a cryptographic mechanism that is specified or registered within a standard, such as an ISO standard. Examples of cryptographic mechanisms are encryption algorithms (such as ISO/IEC 18033), message authentication codes (such as ISO/IEC 9797), authentication protocols (such as ISO/IEC 9798), digital signatures (such as ISO/IEC 9796 or 14888), registered cryptographic algorithms (such as ISO/IEC 9979), and so on.
* If present, one or more subsequent DOs shall either identify a mechanism (tag 06) that was used by the previous mechanism (that is, a mode of operation such as ISO/IEC 10116 or a hash-function such as ISO/IEC 10118) or shall indicate parameters (tag-dependent on the previous mechanism).

Table : Cryptographic Mechanism Identifier Template

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | | |
| AC | Var. | **Cryptographic Mechanism Identifier Template** | | |
|  | | **Tag** | **Len** | **Value** |
| 80 | Var. | Cryptographic mechanism reference (see “Cryptographic Mechanism References”) |
| 06 | Var. | Cryptographic mechanism identifier (See ISO/IEC 7816-4:2005 section 5.4.2) |
| 06 | Var. | Optional additional identifiers (mode of operation, hash function, or parameter) |

For details on how the cryptographic mechanism identifier is built up, see ISO/IEC 7816‑4:2005 section 5.4.2, “Cryptographic Mechanism Identifier Template.”

**Notes:**

* The presence of cryptographic mechanism identifier templates (tag AC) is optional. These templates could be used to list the cryptographic primitives that the on-card application may want to make discoverable to other applications. There could be as many cryptographic mechanism identifier templates as there are cryptographic mechanisms that are supported by the application. The cryptographic mechanism identifier templates provide information on some of the algorithms that are available to the application, regardless of whether a key has been initialized. (An algorithm might be available without any key that uses that algorithm being initialized yet.) It is not a key discovery mechanism, but is more a capability descriptor. Because no algorithm is mandatory, tag AC could help retrieve some of the algorithms that are actually available in the selected on-card application.
* To minimize the size of the DF FCP and optimize transmission speed, an application may decide to list only a subset of the cryptographic mechanism identifiers that it wants to make discoverable to other off-card applications. Likewise, an application may decide to list only the basic cryptographic algorithms when all modes of operation (such as encrypt, decrypt, ECB, and CBC) and padding that is listed in “Cryptographic Mechanism References” are supported.

#### EF FCP

The following sections describes the minimum structure of EF FCP. Additional DOs may be present in these FCP templates, but are not part of this specification.

##### FCP for Data Object EF and Binary EF

For DO and binary EFs, the FCP template shall be constructed as shown in the following table.

Table : FCP Template Assignment Data Objects for Data Object and Binary EF

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | | |
| 62 | Var. | **FCP Template** | | |
|  | | **Tag** | **Len** | **Value** |
| 82 | 01 | File descriptor byte: 39 for DO EF and 10 for binary EF. |
| 83 | 02 | EFID |
| 8A | 01 | File life cycle (see “Table 18: Life Cycle Status Byte”) |
| 8C | Var. | Security attribute in compact format (see “Table 2: Security Attribute in Compact Form”) |

##### FCP for Key EF with CRT Support

For key EFs with CRT support, the FCP template shall be constructed as shown in the following table.

Table : FCP Template Assignment Data Objects for Key EF with CRT support

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | | | | |
| 62 | Var. | **FCP Template** | | | | |
|  | | **Tag** | **Len** | **Value** | | |
| 82 | 01 | File descriptor byte = 18 | | |
| 83 | 02 | EFID | | |
| 8A | 01 | File life cycle (see “Table 18: Life Cycle Status Byte”) | | |
| 8C | Var. | Security attribute in compact format (see “Table 2: Security Attribute in Compact Form”) | | |
| A5 | Var. | List of all CRTs supported by the key as described in “Control Reference Template (CRT)” | | |
|  | | | | **Tag** | **Len** | **Value** |
| CRT Tag as per “Table 23: CRT Tags” | Var. | CRT value as per “Table 14: CRT from EF FCP” |
| CRT Tag as per “Table 23: CRT Tags” | Var. | CRT value as per “Table 14: CRT from EF FCP” |
| … | … | … |
| CRT Tag as per “Table 23: CRT Tags” | Var. | CRT value as per “Table 14: CRT from EF FCP” |

With each CRT being coded as per “Table 23: CRT Tags.”

Table : CRT from EF FCP

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | | | | | |
| CRT tag as per “Table 23: CRT Tags” | Var. | **CRT Value**—see “Control Reference Template (CRT)” | | | | | |
| **Tag** | **Len** | **Value** | | | |
| 80 | Var. | Cryptographic mechanism reference as defined in “Cryptopgraphic Algorithms” | | | |
| 83 | Var. | Key reference: Reference of a secret key (for direct use) or reference of a public key (tag present only if applicable) | | | |
| 84 | Var. | Key reference: Reference of a private key (tag present only if applicable) | | | |
| 95 | Var. | Usage qualifier byte as per “Table 24: Usage Qualifier Byte” | | | |
| A3 | Var. | **Key Usage Template** (optional) | | | |
| **Tag** | **Len** | **Value** | **Comment** |
| 90 | 02 | xxxx | Key usage counter (see “User Authentication ALWAYS and Key Usage Counter”) |

**Notes:**

* The key reference, which uniquely identifies the key value, must be the same in all CRTs.
* Both public and private keys of a key pair must share the same key reference. The tag in the CRT (83 or 84), combined with the cryptographic mechanism reference, allows developers to define the precise use of each key within a key pair.
* The FCP can list multiple CRTs even within the same category when the CRTs reference different cryptographic mechanisms (as long as they are compatible with the key value that is stored in the key EF). However, a generally good cryptographic practice is to employ a given key in only one scheme. This avoids the risk that vulnerability in one scheme can compromise the security of other and may be essential to maintain provable security.
* The FCP does not change when a key value is updated. However, if the new key requires a different set of CRTs, the EF must first be deleted and a new key EF created with the correct FCP.

### FMD Template

The FMD template is a set of interindustry DOs for file management. It is retrieved in the data field of the SELECT command with P2 = 08 and is available only for the ADF. The FMD template is constructed as shown in the following table.

Table : FMD Template Assignment Data Objects for the ADF

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | | |
| 64 | Var. | **FMD Template** | | |
|  | | **Tag** | **Len** | **Value** |
| 5F2F | Var. | PIN usage policy (see “PIN Usage Policy”) |
| 7F65 | Var. | Memory resource assignment template (ISO/IEC 7816-13)—see “Memory Resource Assignment Data Objects” |
| 79 | Var. | First coexistent tag allocation authority. (This DO could be repeated as many times as the number of coexistent tag allocation authorities that are supported. They are sorted in decreasing priorities.) For content, see ISO/IEC 7816-4:2005 section 5.2.4.2. |
| … | … | … |
| 79 | Var. | Cryptographic mechanism identifier template |
| 7F681 | Var. | Sequence of primitive BER-TLV encoded names under which the application may be known (such as 06 for OID, 12 for printable text, and so on) |

1 Tag 7F68 is not yet defined in an official published ISO standard.

Additional DOs may be present in this template to answer the specific need of a given application, but they are not mandatory for GIDS compliance.

### Memory Resource Assignment Data Objects

A memory resource assignment template (tag 7F65) provides the amount of free persistent memory resources that is still available to the application. It is linked to an application and can be retrieved only in the FMD of the ADF.

Table : FMD Template Assignment Data Objects for the ADF

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | | |
| 7F65 | Var. | **Memory Resource Assignment Template** (ISO/IEC 7816-13) | | |
|  | | **Tag** | **Len** | **Value** |
| 80 | 00 | This DO must be present for compliance with ISO/IEC 7816-13 (amount of reserved memory in persistent storage for the application's code). However, it is not used by GIDS and its length should be 00. |

The amount of free EEPROM that is available to the application can be directly retrieved with the GET DATA by using the Free EEPROM GIDS system DO.

## GIDS System Data Objects

GIDS includes support for several DOs that are managed and used by the operating system itself. They are used to set system parameters or to retrieve data that is dynamically updated by the card. These DOs are listed in the following sections.

### System Data Object Retrieved with GET DATA

The following system DOs, which are dynamically updated by the card, can be retrieved with a GET DATA command with P1-P2 = 3FFF if the security status satisfies the security attributes that are associated with the DO.

Table : System Data Object Retrieved with GET DATA

|  |  |
| --- | --- |
| **Name** | **Tag** |
| Global PIN status (see “Global PIN Status”) | 7F71 |
| Local PIN status (see “Local PIN Status”) | 7F72 |
| Local PUK status (see “Local PUK Status”) | 7F73 |

# Addressing Data Structures

## Introduction

Addressing application data and keys is achieved by using the GET DATA and PUT DATA commands with an odd INS byte. The use of the odd INS byte requires that both data and response fields of the command be encoded as BER-TLV DOs.

## P1-P2 Parameters in the GET DATA and PUT DATA Commands

In the context of the odd INS byte form of the GET DATA and PUT DATA commands, the GET DATA and PUT DATA command P1-P2 parameters provide the file identifier (FID) on 2 bytes where the DOs are to be located. This FID provides access control rules and context for context-dependent DOs.

GIDS supports only the following FID values:

* FID = 00-00 Current EF
* FID = 2F-00 EF.DIR
* FID = 2F-01 EF.ATR
* FID = 3F-FF Current Application (ADF)
* FID = EFID EFID of an EF that is present in the current application

A GET DATA or PUT DATA command performs an implicit selection of the EF that is indicated by P1-P2 except when P1-P2 equals 2F-00 or 2F-01. This allows a GIDS application to retrieve the contents of EF.ATR and EF.DIR that are in the MF without losing the current selected application or its security status.

Selection of the current application does not change the security status of the selected application. GIDS does not support an implicit current EF when an application (ADF) is selected. When an ADF is selected (new or current), the position of the current EF is reset to “No currently selected EF”.

## Data Handling Tags

To facilitate handling DOs, GIDS supports thetTag 5C tag list subset of ISO/IEC 7816-4 DO handling tags.

### Tag List 5C

The use of tag list 5C in the GET DATA command data field is supported in the following cases:

* Tag list with a single tag value
* Empty tag list
* Tag list within a tag list

#### Tag List with a Single Tag Value

This is the most common use of a tag list because it allows retrieving a DO by its tag.

GIDS supports only a single tag value in the tag list: 5C “Tag length” “Tag of value to retrieve”. When multiple DOs are available in the file that is indicated in P1-P2 according to the current access conditions that were established in the card application at the time of request, the returned byte string is the concatenation of the available DOs without delimiter.

#### Empty Tag List

The use of an empty tag list 5C 00, (that is, no DO tag is provided in the request) is allowed. This coding allows retrieving all DOs that are available in the file that is indicated in P1-P2 according to the current access conditions that are established in the card application at the time of the request.

The returned byte string is the concatenation of the available DOs without delimiter.

When used with P1-P2 = 3F-FF, this allows developers to get all DOs that are available from the current application (ADF) in one command. This same request executed before and after a modification of the security conditions (such as PIN validation) might provide a different response byte string because more DOs may be accessible (such as after a valid PIN is presented).

#### Tag List within a Tag List

The specific format 5C015C allows retrieving the list of all tags of DOs that are available in the addressing space that is provided by P1-P2. Only the first level of tags is returned (no inner tags in constructed DOs are returned), and the returned byte string is formatted as tag list 5C-LL, which is a list of tags without delimiter.

# Data Object Management

## GIDS Data Objects

GIDS defines only the self-contained BER DO that is managed differently.

### Self-Contained BER Data Object

A self-contained BER DO (also called SC DO) is a BER-TLV DO that is stored into the card in one block.

GIDS requires that an SC DO tag be consistent with its content—that is, if the tag is of constructed type, the DO shall contain at least two nested DOs within the first level in its inner structure.

A constructed SC DO can have duplicate tags in its inner structure, and GIDS does not check whether the BER tag that is used authorizes the presence of duplicate tags in the inner structure.

SC DOs are stored together with other BER DOs in DO EFs.

## Data Object Management

### SC DO

#### SC DO Creation

An SC DO is always created in a single PUT DATA command that may be chained between several APDU if the size of the DO requires it.

The EF in which the DO is created is specified by the P1-P2 parameters of the PUT DATA APDU. It can be explicit (that is, the EFID of the EF in which the DO is to be created) or implicit (that is, P1-P2 = 00 00 to use the currently selected EF). If no EF is currently selected, an error is returned.

#### SC DO Update

An SC DO update is triggered by a PUT DATA command with the new SC DO in the command data field.

The EF in which the DO to be updated is stored is specified by the P1-P2 parameters of the PUT DATA APDU. It can be explicit (that is, EFID of the EF) or implicit (that is, P1-P2 = 00 00 to use the currently selected EF or = 3F FF when only one DO with that tag exists in the card application).

If a DO with the same tag already exists in the addressing span that is specified by P1-P2, it is overwritten with the new value. If the length of the new value is null, the existing SC DO is deleted.

#### SC DO Deletion

A SC DO in an EF can be deleted by using a PUT DATA on the DO’s tag with an empty value.

## Atomicity of DO Management Operations

If a DO operation fails for any reason, the GIDS implementation shall ensure that older data, if any, is still available in the same way as it was before the command that failed.

When the GIDS implementation cannot guarantee atomicity of an update, the command shall fail with a status that indicates that the operation cannot be completed. It is then the responsibility of the client application to free enough resources, such as by removing older contents first.

## Coordinated DO Management Operations

GIDS allows chaining DO updating operations (such as create, update, and delete).

Multiple DO can be updated within the same chaining sequence, but a new APDU must start with every DO.

If any of the operations in the chain fails, the complete set of updates is canceled (that is, all the DOs updated within the chaining) and previous values, if any, remain unaffected.

Repeated updates to the same DO in a chain is allowed. In that case, only the last value is committed.

# Discovery of Applications

## Applications Discovery

From any GIDS-based application, it is possible to obtain the list of all GIDS-based applications (ADFs) that are present in the card by sending a GET DATA command with P1-P2 = 2F 00 and a command data field 5C00. For details, see “EF.DIR.”

Information on the current application can be retrieved by sending a SELECT command with P1 = 00 and a command data field set to 3F FF.

If P2 = 00, the command returns the application control information (FCI) from which application AID and other application-related data can be extracted.

If P2 = 04, the command returns the application control parameters (FCPs) from which logical, structural, and security attributes and cryptographic capability can be retrieved.

If P2 = 08, the command returns the application management data (FMD) from which information can be retrieved, such as PIN usage policy, tag allocation authorities, OID under which the application is known, and memory resources still available.

Selecting the current application by using a command data field of 3F-FF changes neither the currently selected application nor the currently selected EF or the current security status.

For details on the SELECT command, see “Application Template Data Object (FCI).”

## PIN Usage Policy Discovery

The PIN usage policy is the DO with tag 5F2F that is part of the FMD that is returned when the application is selected with P1 = 00 (select by EFID) and P2 = 08. For details, see “PIN Usage Policy.”

## Cryptographic Capabilities Discovery

Cryptographic capabilities of the application can be retrieved from the DO with tag AC that is part of the FCP that is returned when the application is selected with P1 = 00 (select by EFID) and P2 = 04.

Tag AC lists only the algorithms that are actually available in implementation. For details, see “DF FCP.”

## File Structure Discovery

GIDS supports file structure discovery by supporting SELECT/FIRST and SELECT/NEXT with an empty data field. SELECT/FIRST selects the first EF that was created in the ADF if it exists. Successive SELECT/NEXT commands then iterate over the list of EFs, selecting every available EF exactly once. SELECT/NEXT after the last EF fails with “File or application not found” (6A82). The iteration order is unspecified.

## Current Elementary File Identification

The client application can always access the FCI/FCP of the current EF by using a SELECT/FID with a file identifier of 00 00. Depending on P2, SELECT then returns the appropriate FCI/FCP. See ISO/IEC 7816-4 section 5.3.1.1.

## DO List at DF Level

The list of all top-level DOs from the DF that are accessible with respect to the current security status can be retrieved by using GET DATA that targets the DF and a tag list that requests a tag list (5C 01 5C).

Some tags may be duplicated when corresponding DOs are present in different EFs.

## DO List at EF Level

The list of all top-level DOs from an EF that are accessible with respect to the current security status can be retrieved by using GET DATA that targets the EF (either P1-P2 = EFID, or P1-P2 = 00 00 with EF selected) and a tag list that requests a tag list (5C 01 5C).

If the EF is a DO EF, the list contains the tags of BER-TLV DOs (SC DO, template, and reference).

If the EF is a binary EF, the returned tag is the virtual tag that is associated with the container if defined; otherwise, tag 53 is returned.

In an EF, there can be no repetition of a tag at the outermost level. Tags from global DOs that are stored at the MF level are *not* returned as part of the list at EF level.

## DO Values at DF Level

The value of all top-level DO that are accessible at DF level with respect to the current security status can be retrieved by using GET DATA, which targets the DF and an empty tag list (5C 00).

The list contains all DO with tags and values, including DOs from DO EFs and binary EFs as well as global DOs from MF. A DO from a binary EF is returned wrapped into the virtual tag that is associated with its container, if it is defined. Otherwise, it is wrapped with tag 53.

Top-level reference DOs are not returned. Returning a DO that is visible through references would be redundant because the referred DO with the same tag will be returned on its own.

## DO Values at EF Level

The value of all top-level DO that are available from an EF can be retrieved by using GET DATA, which targets the EF and an empty tag list (5C 00). The behavior is the same as when retrieving the value of all DOs at DF level except that reference DOs are also returned when the referred data is not stored in the targeted EF.

**Note:** Because an EF can contain a DO reference, the content of DOs in the list may depend on the security status.

# GIDS Life Cycle Management

Information on the application life cycle status (LCS) can be retrieved from the LCS byte, which is a file control parameter DO with tag 8A that is retrievable from the ADF FCP. Similarly, information on the LCS of an EF can be retrieved from the EF FCP.

The following LCS values apply to ADF and EF.

Table : Life Cycle Status Byte

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **Meaning** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Creation state |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | Initialization state |
| 0 | 0 | 0 | 0 | 0 | 1 | - | 1 | Operational state—activated |
| 0 | 0 | 0 | 0 | 0 | 1 | - | 1 | Operational state—deactivated (key EF only) |
| 0 | 0 | 0 | 0 | 1 | 1 | - | - | Termination state (only for ADF) |

For the EF LCS byte, the value of bit b2 is set automatically by GIDS, depending on whether the file was created while the application DF was in the creation state (b2=0) such as during initial personalization or in the operational state (b2 = 1) such as during post-issuance update.

## Creation State

### Application Creation State

When a GIDS application is created, the ADF is in the creation state, access control rules are not enforced, and the following commands are available:

* SELECT to select the application:

P1 = 04, P2 = 00 (SELECT By Name, First Occurrence, and FCI Returned)

* PUT DATA with P1-P2 = 00 00 to load GIDS metadata as described in “GIDS Metadata”:

DF FCI (Application Template DO) (Tag 61)

This command data field shall be per “Table 7: Application Template Assignment Data Objects.”

DF FMD (Tag 64)

This command data field shall be per “Table 15: FMD Template Assignment Data Objects for the ADF” except it shall not include DO 7F65 (memory resource assignment template). This is dynamic information that GIDS adds automatically.

DF FCP (Tag 62)

This command data field shall be per “Table 10: FCP Template Assignment Data Objects for DF” except it shall not include DO 8A, LCS byte. This is dynamic information that GIDS adds automatically.

GIDS implementation is not required to check the FCI or FMD contents and the presence of the field that is mandated by 2.6.

EF cannot be created when the application is in the creation state.

GIDS does not support any other APDU while the application is in the creation state. Transition from the creation state to the initialization state is achieved automatically when the DF FCP is loaded. As a result, DF FCI and DF FMD must be loaded before DF FCP.

## Initialization State

### Application Initialization State

The application initialization state is the life cycle phase when EFs are created and populated.

In the application initialization state access, control rules are not enforced (even on an EF that has been set in the operational—activated state) and cryptographic operations are not supported (except generate asymmetric key pair and secure messaging).

### EF Initialization State

An EF that has just been created is in the initialization state. In this state, its content is not yet subject to the access control rules that were defined in the FCP and the following default applies. For EFs that contain application DOs, PUT DATA and GET DATA are always available.

For EFs that contain key properties, PUT KEY, GENERATE ASYMMETRIC KEY, and GET PUBLIC KEY commands are always available. Other APDUs are not allowed.

## Operational State

### Operational State—Activated

ADF and EF LFC can be transitioned from the initialization state to the operational state by using the ACTIVATE FILE command. See “ACTIVATE FILE.”

When the ADF is in the operational state, all functionalities are available and access rules are fully enforced on the ADF as well as on all operational—activated EFs.

When the EF is in the operational—activated state, all functionalities are available and access rules are fully enforced if the ADF is also in the operational state. Otherwise, the EF behaves as if it were still in the initialization state until the ADF is set to the operational state.

### Operational State—Deactivated

The LFC operational—deactivated state is only supported for key EFs.

For EFs of the key type, the file life cycle, once in the operational state, is automatically updated from deactivated to activated when a key is loaded and reverts automatically to deactivated when a key is zeroized until a new key is loaded.

## Termination State

### Application Termination State

The application switches to the termination state with the TERMINATE DF command. See “TERMINATE DF.”

When the DF is in the termination state, PUT DATA and cryptographic operations are no longer authorized. Only SELECT and GET DATA on freely available DOs can still be performed.

### EF Termination State

EFs do not support the termination state. EFs that are no longer used should be deleted.

# CHV Management

GIDS may support multiple ways to perform CHV. This specification describes only specific verification by using a PIN (application or global).

In this specification, the meaning of PIN has been extended beyond a numeric value to include any hexadecimal characters. As a result, the PIN (application or global) can be a passphrase of up to 127 characters. The minimum size of the PIN value that is transmitted to the card is 8 bytes.

It is the responsibility of the off-card application to format the value that the cardholder supplies and apply padding if necessary.

Unless specified otherwise, each CHV method has its own retry counter limit. When CHV is successful, only the retry counter that is associated with the CHV value is reset.

## PIN Usage Policy

The PIN usage policy in the DF FMD provides information that indicates which CHV method is supported—that is, whether a successful global PIN verification can satisfy CHV status or only the application PIN. See “Table 15: FMD Template Assignment Data Objects for the ADF.” The following table provides the construction of the PIN usage policy.

Table : First Byte of PIN Usage Policy

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **Meaning** |
| 1 | - | - | - | - | - | - | - | RFU |
| - | 1 | - | - | - | - | - | - | Application PIN is present and may be used to satisfy CHV. If administrator key challenge/response is not used to reset the application PIN retry counter, the application PUK must also be present to reset the application PIN retry counter. |
| - | - | 1 | - | - | - | - | - | Global PIN is present and may be used to satisfy CHV. |
| - | - | - | 1 | - | - | - | - | RFU |
| - | - | - | - | 1 | - | - | - | RFU |
| - | - | - | - | - | 1 | - | - | RFU |
| - | - | - | - | - | - | 1 | - | RFU |
| - | - | - | - | - | - | - | 1 | RFU |

## Application PIN Management

The presence of an application PIN is optional. The application PIN can be created only when the application is in the initialization state. Its initial value is set by using the CHANGE REFERENCE DATA command with P1 = 01. The application PIN value can be changed during the personalization or operational state with a customer-specific value by using the CHANGE REFERENCE DATA with P1 = 00. See “CHANGE REFERENCE DATA.”

The number of consecutive failed PIN verifications before the PIN becomes blocked (PIN retry counter) can be set at any time after the PIN has been created by using the PUT DATA command. See “GIDS System Data Objects.” When blocked, the application PIN may be unblocked by using RESET RETRY COUNTER in a resetting code (PUK) if one has been previously initialized.

## Global PIN Management

The presence of a global PIN is optional.

When present, the global PIN is set by using the CHANGE REFERENCE DATA command with P1 = 01 when the application is in initialization state (if it has not already been initialized by another on-card application). See “CHANGE REFERENCE DATA.”

The PIN value that is loaded during application initialization can be replaced during the personalization or operational state with a customer-specific value by using the CHANGE REFERENCE DATA with P1 = 00.

The number of consecutive failed PIN verifications before the PIN becomes blocked (PIN retry counter) can be set at any time after the PIN has been created by using the PUT DATA command. See “GIDS System Data Objects.”

## User Authentication ALWAYS and Key Usage Counter

ISO/IEC 7816-4 SCB does not support “User Authentication Always.” To offer a “Pin Always” access condition before using a key for a cryptographic computation (such as a digital signature), GIDS uses the key usage counter from the EF FCP.

The key usage counter in the key EF FCP is used to define a maximum number of cryptographic operations with the key under a single CHV. A value set to zero means an unlimited number of consecutive cryptographic operations. A value different from zero (that is, “n”) requires a CHV immediately before the first use of the key and then every “nth” use of the key until the card reset or another on-card application is selected.

For example, a value of 1 would produce a CHV ALWAYS condition that is similar to the PIN ALWAYS condition that is defined in U.S. Federal Specification SP 800-73-2. A value of 01F4 that is associated to a signature key requires a CHV every 500 signatures, which limits the delegation that a manager can give to an assistant.

The key usage counter applies only for APDUs that are protected by a CHV access condition.

## CHV Status

The following DOs can be used to retrieve the status that is associated with CHV methods:

* Global PIN status
* Local PIN status
* Local PUK status

### Global PIN Status

Table : Global PIN Status

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | | |
| 7F71 | Var. | Global PIN Status | | |
|  | | **Tag** | **Len** | **Value** |
| 97 (9F17[[1]](#footnote-1)) | 01 | Try counter (tries remaining) |
| 93 | 01 | Try limit |

### Local PIN Status

Table : Local PIN Status

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | | |
| 7F72 | Var. | Local PIN Status | | |
|  | | **Tag** | **Len** | **Value** |
| 97 (9F171) | 01 | Try counter (tries remaining) |
| 93 | 01 | Try limit |

### Local PUK Status

Table : Local PUK Status

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | | |
| 7F73 | Var. | Local PUK Status | | |
|  | | **Tag** | **Len** | **Value** |
| 97 (9F171) | 01 | Try counter (tries remaining) |
| 93 | 01 | Try limit |

If RESET RETRY COUNTER command can reset the *Application Password* after performing an External or Mutual authentication with the Administrator key using GENERAL AUTHENTICATE, the PUK is not available for resetting the reference data. In that case, GET DATA command for local PUK status must return 6A 88 (Reference Data Not Found).

# Cryptopgraphic Algorithms

## Control Reference Template (CRT)

In ISO 7816-4, the security architecture is shared by secure messaging, authentication (EXTERNAL/INTERNAL/GENERAL AUTHENTICATE), and general cryptographic services (PERFORM SECURITY OPERATION). The central concept is the CRT.

A CRT is a structure that holds an association of:

* A function (defined by the CRT tag).
* A cryptographic mechanism (reference into a list at DF level).
* A key value (secret or public reference to unspecified storage).
* Management data such as initialization vectors or counters.

As part of the ADF security environment, GIDS maintains a set of CRTs—known as current security environment (SE) components in ISO/IEC 7816-4:2005 section 6.3.3—that are used whenever an implicit cryptographic service is required (as in secure messaging, or with PERFORM SECURITY OPERATION or GENERAL AUTH with P1-P2 = 00-00).

Setting the SE is achieved by MANAGE SECURITY ENVIRONMENT.

The structure of a CRT is defined in ISO/IEC 7816-4 table 33. GIDS supports the CRTs in the following table.

Table : CRT Tags

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag** | **Name** | **Meaning** | **Used for** |
| A4 | AT | Authentication | INT AUTH and GEN AUTH |
| B6 | DST | Digital signature | PSO |
| B8 | CT | Confidentiality | SM-in, SM-out, and PSO |

The format of the CRT that GIDS supports is defined in “Table 14: CRT from EF FCP.”

Each CRT includes a usage qualifier byte (tag 95) that further defines the usage of the template. The following table extracted from ISO/IEC 7816-4:2005 table 35 defines the usage qualifier byte values that GIDS supports.

Table : Usage Qualifier Byte

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **Meaning** |
| 1 | - | - | - | - | - | - | - | Verification (DST), encipherment (CT), and external authentication (AT) |
| - | 1 | - | - | - | - | - | - | Computation (DST), decipherment (CT), and internal authentication (AT) |
| - | - | - | - | 0 | 0 | 0 | 0 | Reserved for future use |

## Cryptographic Mechanism References

The following cryptographic mechanism identifiers have been defined by GIDS for greater interoperability.

A card that complies with GIDS may not support all the listed cryptographic mechanism identifiers. The list of supported cryptographic mechanisms may be provided in the cryptographic mechanism identifier template (tag AC) from the DF FCP. See “DF FCP.”

### Authentication

The following table defines the cryptographic mechanism references for an AT CRT.

Table : Cryptographic Mechanism Reference for AT CRT

|  |  |  |
| --- | --- | --- |
| **b8 – b5** | **b4 to b1** | **Meaning** |
| 0 | - | RFU |
|  | 1 | 2 Key Triple DES |
| 2 | 3 Key Triple DES |
| 3 | AES 128 |
| 4 | AES 192 |
| 5 | AES 256 |
| 6 | RFU |
| 7 | RFU |
| 8 | RFU |
| 9 | RFU |
| A | ECC: Curve P-192 |
| B | ECC: Curve P-224 |
| C | ECC: Curve P-256 |
| D | ECC: Curve P-384 |
| E | ECC: Curve P-521 |
| Other | RFU |

**Note:**

* ECC curves parameters are defined in FIPS 186-3 appendix E.

### Confidentiality

The following tables define the cryptographic mechanism references for a CT CRT for both symmetric and asymmetric algorithms.

Table : Cryptographic Mechanism Reference for CT CRT with Symmetric Algorithm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **b8** | **b7** | **b6-b5** | **b4 - b1** | **Meaning** |
| 0 |  |  |  | No padding enforced by the card |
| 1 |  |  |  | ISO/IEC 9797 padding method 21 |
| - | 0 |  |  | CBC mode |
| - | 1 |  |  | ECB mode |
|  |  | 00 |  | RFU |
|  |  |  | 1 | 2-key triple DES |
|  |  |  | 2 | 3-key triple DES |
|  |  |  | 3 | AES 128 |
|  |  |  | 4 | AES 192 |
|  |  |  | 5 | AES 256 |
|  |  |  | Other | RFU |

1Mandatory padding on the right with 80 followed by as many zeros as needed.

Table : Cryptographic Mechanism Reference for CT CRT with Asymmetric Algorithm

|  |  |  |  |
| --- | --- | --- | --- |
| **b8-b7** | **b6-b5** | **b4-b1** | **Meaning** |
| 00 | - | - | No padding enforced by the card |
| 01 | - | - | RSAES-PKCS1-v1\_5 padding |
| 10 | - | - | RSAES-OAEP padding |
| 11 |  |  | RFU |
| - | 00 | - | RFU |
|  |  | 6 | RSA 1024-bit |
|  |  | 7 | RSA 2048-bit |
|  |  | 8 | RSA 3072-bit |
|  |  | 9 | RSA 4096-bit |
|  |  | Other | RFU |

**Notes:**

* When no padding is enforced, it is the responsibility of the off-card application to ensure that the data is block-aligned. The encryption/decryption process fails if data is not block-aligned.
* When a padding scheme is selected, padding is performed automatically by the card during encryption and verified during decryption.
* For optimal asymmetric encryption padding (OAEP), the following default parameters from PKCS#1 V2.1 are made mandatory by GIDS:

The value of the input Label L shall be an empty string as required by PKCS#1 V2.1.

The mask generation function (MGF) shall be MGF1 with SHA-1 as specified in PKCS#1 V2.1 Appendix B.2.1.

* Two encryption schemes are supported by GIDS: RSAES-OAEP and RSAESPKCS1-v1\_5. Whenever possible, the RSAES-OAEP should be preferred as per PKCS#1 v2.1 recommendations. RSAES-PKCS1-v1\_5 for encryption is included only for compatibility with legacy systems.
* If multiple cryptographic schemes are to be authorized for a given key, the EF FCP that is associated with that key may list one CRT per scheme to be supported. The card would then know which scheme to use by looking at the CRT that the previous MSE:SET command selected.
* This version of GIDS does not support asymmetric encryption that uses a public key that is supplied by the off-card application. Encryption and decryption can be done only with keys that are stored in a key EF with a CT CRT that defines the cryptographic scheme to be used.

### Digital Signature

The following table defines the cryptographic mechanism references that are supported by GIDS for a DST CRT.

Table : Cryptographic Mechanism Reference for DST CRT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **b8** | **B6** | **b5** | **b4 - b1** | **Meaning** |
| 00 |  |  | - | No padding enforced by the card. |
| 01 |  |  | - | RSASSA PKCS1-v 1\_5 padding scheme (for RSA only; otherwise, RFU) |
| 10 |  |  | - | RFU |
| 11 |  |  | - | RFU |
|  | 1 | - |  | RFU |
|  | - | 1 |  | Full SHA off-card authorized |
|  | 0 | 0 |  | RFU |
|  |  |  | 6 | RSA 1024 bit |
|  |  |  | 7 | RSA 2048 bit |
|  |  |  | 8 | RSA 3072 bit |
|  |  |  | 9 | RSA 4096 bit |
|  |  |  | A | ECDSA with Curve P-192 |
|  |  |  | B | ECDSA with Curve P-224 |
|  |  |  | C | ECDSA with Curve P-256 |
|  |  |  | D | ECDSA with Curve P-384 |
|  |  |  | E | ECDSA with Curve P-521 |
|  |  |  | Other | RFU |

Table : DigestInfo Values for Hash Functions (from RFC 3447)

|  |  |
| --- | --- |
| **Hash algorithm** | **DigestInfo value** |
| MD2 | 30 20 30 0c 06 08 2a 86 48 86 f7 0d 02 02 05 00 04 10 |
| MD4 | 30 20 30 0c 06 08 2a 86 48 86 f7 0d 02 04 05 00 04 10 |
| MD5 | 30 20 30 0c 06 08 2a 86 48 86 f7 0d 02 05 05 00 04 10 |
| SHA-1 | 30 21 30 09 06 05 2b 0e 03 02 1a 05 00 04 14 |
| SHA-256 | 30 31 30 0d 06 09 60 86 48 01 65 03 04 02 01 05 00 04 20 |
| SHA-384 | 30 41 30 0d 06 09 60 86 48 01 65 03 04 02 02 05 00 04 30 |
| SHA-512 | 30 51 30 0d 06 09 60 86 48 01 65 03 04 02 03 05 00 04 40 |

**Notes:**

* PKCS1-v1\_5 padding scheme is as defined in PKCS#1 version 2.1 with additional constraints thatare imposed by FIPS 186-3.
* When PKCS1-v1\_5 padding scheme is selected and the hash is computed completely off-card, DigestInfo that provides information on the hash that was used must be supplied together with the hash value. Refer to “Table 29: DigestInfo Values for Hash Functions.”
* When no padding is enforced, it is the responsibility of the off-card application to ensure that the data is block-aligned. The encryption/decryption process fails if data is not block-aligned.
* ECC curves parameters are defined in FIPS 186-3 appendix E.
* Two encryption schemes are supported by GIDS: RSAES-OAEP and RSAESPKCS1-v1\_5. PKCS #1 v2.1 recommends that RSAES-OAEP be used for new applications. RSAES‑PKCS1‑v1\_5 is included only for compatibility with existing applications and is not recommended for new applications.

# Authentication and Session Key Agreement Protocols

The following authentication protocols may be supported by a GIDS implementation. In addition to authentication, these mechanisms establish a pre-master secret that is shared between both sides that could be used for session key establishment if the authentication is to be followed by a secure messaging. If the off-card application does not want to establish secure messaging, the value of the pre-master secret can be safely ignored.

Establishment of session keys by using a pre-master secret will be specified in a future revision of this specification.

## Mutual Authentication with Symmetric Algorithm

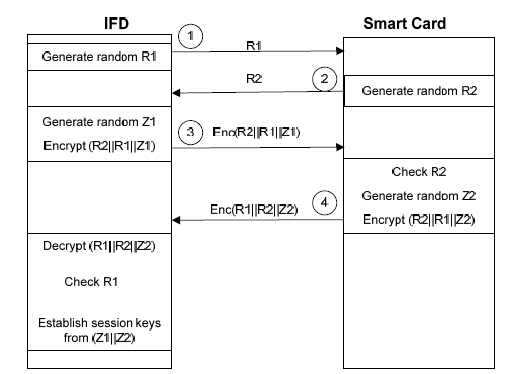
This section describes the mutual authentication protocol by using symmetric algorithm (3DES or AES). A successful authentication allows both sides to share a pre-master secret “Z” that may be subsequently used to open a secure messaging. This secret will be specified in a future revision of this specification. This protocol conforms to ISO11770-2 key establishment mechanism 6 with off-card application in role B, card in role A, and message 2 and 3 swapped. This swapping is necessary for the protocol to fit into GENERAL AUTH.

Figure : Mutual authenticate mechanism with key establishment using symmetric algorithm

**Notes:**

* The length of random R1 that was generated by the off-card application shall be 16 bytes.
* The random R2 that was generated by the card is of the same length as the incoming random R1.
* The length of the random Z1 that was generated by the off-card application depends on the cryptographic mechanism that was specified in the current AT CRT. It shall be half the length of the pre-master secret Z. See “Table 30: Length of Pre-master Secret Z as a Function of Authentication Key Used.”
* The length of the random Z2 that was generated by the card depends on the cryptographic mechanism that was specified in the current AT CRT. It shall be half the length of the pre-master secret Z. See “Table 30: Length of Pre-master Secret Z as a Function of Authentication Key Used.”
* The encryption uses the CBC mode of the symmetric algorithm with ISO/IEC 9797 padding, method 2.

Table : Length of Pre-master Secret Z as a Function of Authentication Key Used

|  |  |  |
| --- | --- | --- |
| **Authentication key type** | **Pre-master secret (Z) size** | **Session key type** |
| 2-key triple DES | 10 bytes (80 bits) | 2-key triple DES |
| 3-key triple DES | 14 bytes (112 bits) | 3-key triple DES |
| AES 128 | 16 bytes (128 bits) | AES 128 |
| AES 192 | 24 bytes (192 bits) | AES 192 |
| AES 256 | 32 bytes (256 bits) | AES 256 |
| ECC: Curve P-192 | 10 bytes (80 bits) | 2-key triple DES |
| ECC: Curve P-224 | 14 bytes (112 bits) | 3-key triple DES |
| ECC: Curve P-256 | 16 bytes (128 bits) | AES 128 |
| ECC: Curve P-384 | 24 bytes (192 bits) | AES 192 |
| ECC: Curve P-521 | 32 bytes (256 bits) | AES 256 |

## External Authentication with Symmetric Algorithm

This section describes the external authentication protocol by using symmetric algorithm (3DES or AES).

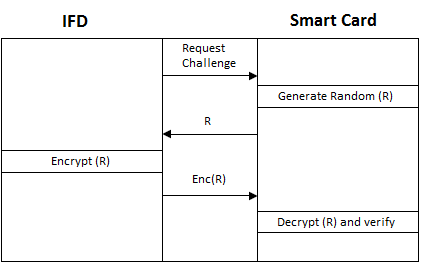


Figure : External authenticate mechanism using symmetric algorithm

**Notes:**

* The length of random R that was generated by the off-card application shall be 8 bytes.

## Key Establishment with Internal Authentication Using ECC

This is a session (ephemeral) key establishment protocol that allows further secure communication of sensitive user data from an off-card application to a card.

This protocol does not require terminal authentication, but maintains the confidentiality of user-entered credentials during their transport to the card. It is particularly suitable to transmit to the card-sensitive cardholder data such as PIN or biometric data for matching on-card from an off-card application but over an unsecured communication channel (such as contactless communication).

The protocol uses a static secret on ICC only, and no secret (keys) are required to be stored on the off-card application side. This removes the need for any key management on the terminal side.

The initiator (the off-card application) generates an ephemeral key pair but uses no static key pair. The responder (the card application) has only a static key pair. The ephemeral key pair is an ECC key pair from the same domain as the card static key pair. The static key pair provides authentication of the card application to the client application. The session key agreement protocol reproduces NIST SP800-56A 6.2.2.2 One-Pass Diffie-Hellman, C(1, 1, ECC CDH). It combines the ephemeral and static key pairs to form on both sides the symmetric session keys for MACing and encryption. Per NIST SP800-56A, session key confirmation from the responder to the initiator is also included.

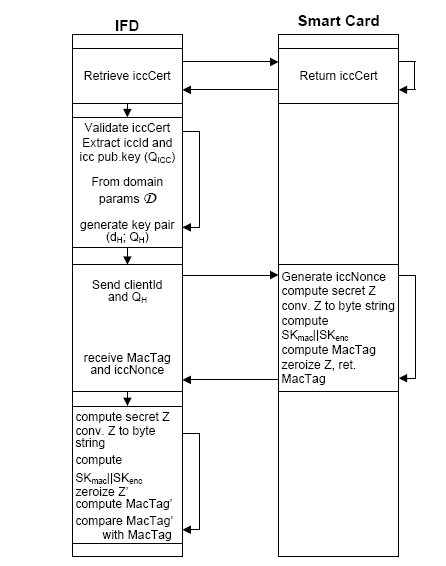
This protocol is the C1,1 ECC CDH from SP 800-56A. It is also defined in ISO/IEC 24727-3.

Figure : Establishment with internal authentication using elliptic curve cryptography

Table : C1,1 ECC CDH Protocol for Key Establishment with Internal Authentication

|  |  |
| --- | --- |
| **Step #** | **Description** |
| 1. | **INTERNAL AUTHENTICATE** for card authentication:  Off-card application sends the nesting authentication DO, which consists of QH (public key host) and IDH (identity host). |
| 2. | Card generates card nonce (NC).  Card computes secret *Z*= **ECC\_CDH** (dC, QH).  Card computes session keys: SKmac || SKenc= **KDF** (*Z*, *len*, *info*).  Card zeroizes *Z.*  *MacData* = “KC\_1\_V” || IDC || IDH || QH  Card computes MacTag= **MAC** (SKmac, *MacData*).  Card returns template with Nonce NC and MacTag. |
| 3. | Off-card application computes secret *Z*= **ECC\_CDH** (dH, QC).  Off-card application computes SKmac || SKenc= **KDF** (*Z*, len, info).  Off-card application zeroizes Z.  Off-card application computes MacTag= **MAC** (SKmac, *MacData*).  Off-card application now compares MacTag with the received MacTag; card is authenticated and the session keys are established good for subsequent SM if this comparison succeeds |

* **ECC\_CDH** refers to the elliptic curve co‑factor Diffie-Hellman primitive as specified in NIST SP800-56A (section 5.7.1.2). Each party uses its own private key dA and the other party’s public key QB and simply computes the point P = hdAQB. The shared secret Zis the x‑coordinate of *P*. To achieve SUITE-B compliance, the recommended elliptic curve domain parameters *D* are P-256 or P-384.
* **Nc** refers to the nonce that is generated by the card. It is a 4-byte random number.
* **KDF** refers to the key derivation function defined in 10.4.
* **MAC** refers to the message authentication code as specified in NIST SP800-56A. It uses the session key SKmac over *MacData.*
* **“KC\_1\_V”**is a predefined 6-byte message string. Its value is set to 00 00 00 00 00 00.
* **KDFHashAlgorithm** refers to SHA1.

For session key establishment, the inputs to the described KDF function are the following:

* Pre-master secret = Z
* IDH: 8-byte value
* IDC: 8-byte value
* NC: 4-byte value

# Key Management

## Key Selection

The selection of keys is performed by the MSE:SET command if not specified otherwise (such as in the parameters of the command header). An appropriate key reference is submitted and stored in the ICC to specify the key to be used with the next commands.

## Reserved Key References

The key reference (tag 83) is a 1-byte DO that complies with ISO/IEC 7816-4 table 65. The following table provides the values that are supported by GIDS.

Table : Key Reference

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **Meaning** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ISO for “No information given” |
| 0 | - | - | - | - | - | - | - | Global reference data |
| 1 | - | - | - | - | - | - | - | Specific reference data (DFspecific key) |
| - | 0 | 0 | - | - | - | - | - | 00 (any other value is reserved for future use) |
| - | - | - | x | x | x | x | x | Qualifier—that is, key reference |

As a result, the key reference for application keys is a 1-byte value that can range from 80 to 9E. Use of global keys in an on-card application is outside the scope of this specification.

## Administrative Key

The external or mutual authentication security conditions from the file FCP can be satisfied only with a successful external authentication by using GENERAL AUTHENTICATE. A key that can satisfy this authentication is called an ADMIN KEY.

To qualify for ADMIN KEY status, a key must be associated with a CRT of type A4 (AT) that has a usage qualifier byte that allows external or mutual authentication (bit b8 set to 1).

To prevent unauthorized creation of an Admin Key, the creation of a key that could be used as an Admin Key requires (when the application life cycle state is operational) a successful external or mutual authentication in addition to the create file security conditions in the DF FCP. See “Table 5: Access Mode Byte for ADF.”

# APDU References

## Command Response Pairs

APDU command-response pairs are handled as indicated in ISO/IEC 7816-3.

## CLASS Byte Coding

CLA indicates the class of the command. According to ISO/IEC 7816-4, FF is an invalid value. Bit 8 of CLA distinguishes between the interindustry class and the proprietary class. Bit 8 set to 0 indicates the interindustry class.

The values 000x xxxx are specified hereafter.

Table : CLASS Byte Interindustry Values

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **Meaning** | **Support** |
| 0 | 0 | 0 | x | - | - | - | - | **Command chaining control** | Yes |
| 0 | - | - | - | - | The command is the last or only command of a chain |
| 1 | - | - | - | - | The command is not the last command of a chain |
| - | x | x | - | - | **Secure messaging indication** | No |
| - | 0 | 0 | - | - | No SM |
| - | 0 | 1 | - | - | Proprietary SM format | No |
| - | 1 | 0 | - | - | SM, command header not processed |
| - | 1 | 1 | - | - | SM, command header authenticated | No |
| - | - | - | X | x | **Logical channel number from zero to three** | Yes |

## Data Fields

Multi-byte data is always formatted as big-endian.

## Status Bytes SW1 and SW2

SW1-SW2 indicates the processing state. Their coding follows the rules shown in the following table.

### General Meaning

Table : General Meaning of the Interindustry Values of SW1-SW2

|  | **SW1-SW2** | **Meaning** |
| --- | --- | --- |
| Normal processing | 9000 | No further qualification |
| 61XX | SW2 encodes the number of data bytes still available |
| Warning processing | 62XX | State of non-volatile memory is unchanged (further qualification in SW2) |
| 63XX | State of non-volatile memory has changed (further qualification in SW2) |
| Execution error | 64XX | State of non-volatile memory is unchanged (further qualification in SW2) |
| 65XX | State of non-volatile memory has changed (further qualification in SW2) |
| 66XX | Security-related issues |
| Checking error | 6700 | Wrong length; no further indication |
| 68XX | Functions in CLA not supported (further qualification in SW2) |
| 69XX | Command not allowed (further qualification in SW2) |
| 6AXX | Wrong parameters P1-P2 (further qualification in SW2) |
| 6B00 | Wrong parameters P1-P2 |
| 6CXX | Wrong Le field; SW2 encodes the exact number of available data bytes |
| 6D00 | Instruction code not supported or invalid |
| 6E00 | Class not supported |
| 6F00 | No precise diagnosis |

### Specific Interindustry Warning and Error Conditions

Table : Specific Interindustry Warning and Error Conditions

| **SW1** | **SW2** | **Meaning** |
| --- | --- | --- |
| 62 (warning) | 00 | No information given |
| 01 to 80 | RFU |
| 81 | Part of returned data possibly corrupted |
| 82 | End of file or record reached before reading Ne bytes |
| 83 | Selected file deactivated |
| 84 | File control information not correctly formatted |
| 85 | Selected file in termination state |
| 86 | No INPUT DATA available from a sensor on the card |
| 63 (warning) | 00 | No information given |
| 81 | File filled up by the last write |
| CX | Counter from 0 to 15 encoded by X (exact meaning depending on the command) |
| 64  (error) | 00 | Execution error |
| 01 | Immediate response required by the card |
| 02 to 80 | Triggering by the card |
| 65  (error) | 00 | No information given |
| 81 | Memory failure |
| 68  (error) | 00 | No information given |
| 81 | Logical channel not supported |
| 82 | Secure messaging not supported |
| 83 | Last command of the chain expected |
| 84 | Command chaining not supported |
| 69  (error) | 00 | No information given |
| 81 | Command incompatible with file structure |
| 82 | Security status not satisfied |
| 83 | Authentication method blocked |
| 84 | Reference data not usable |
| 85 | Conditions of use not satisfied |
| 86 | Command not allowed (no current EF) |
| 87 | Expected secure messaging DOs missing |
| 88 | Incorrect secure messaging DOs |
| 6A  (error) | 00 | No information given |
| 80 | Incorrect parameters in the command data field |
| 81 | Function not supported |
| 82 | File or application not found |
| 83 | Record not found |
| 84 | Not enough memory space in the file |
| 85 | Lc inconsistent with TLV structure |
| 86 | Incorrect parameters P1-P2 |
| 87 | Lc inconsistent with parameters P1-P2 |
| 88 | Referenced data or reference data not found (exact meaning depending on the command) |
| 89 | File already exists |
| 8A | DF name already exists |

### Status Word Treatment for Interoperability

Depending on each implementation, the status word values may vary. For interoperability, the following are the status words to verify:

* Normal ending: 9000 🡪 No error
* Warning, the command was executed but a concern was detected: 6200; 6281 to 629F / 6300; 6381 to 639F
* Error, command not executed: 6400 / 6800; 6881 to 688F / 6900; 6981 to 699F / 6A00; 6A80 to 6A9F / 6581 / 6700 / 6B00 / 6D00 / 6E00
* Error generated following a secret data verification that led to a ratification 63C*x*, (*x* indicating the remaining number of tries)

## Command Chaining

### GIDS Commands Supporting Command Chaining

The following commands support command chaining (CC) as described in this section:

* PUT DATA
* VERIFY
* GENERAL AUTHENTICATE
* PERFORM SECURITY OPERATION

### Description of the Command Chaining

CC is a transaction control mechanism introduced by ISO.IEC 7816-4 whereby in the interindustry class consecutive command response pairs can be chained. The mechanism may be used when executing a multistep process, such as transmitting a data string that is too long for a single command.

GIDS supports the CC mechanism as indicated in the EF.ATR. See “EF.ATR.”

However, ISO/IEC 7816-4 specifies the card behavior only in the case where, once initiated, a chain is terminated before initiating a command-response pair that is not part of the chain. Otherwise, the card behavior is not specified.

GIDS fills that gap by defining the card behavior on a failed chain as follows. The description uses the PUT DATA command for illustration, but applies to any of the commands in “GIDS Command Set APDU” where the value of the CLASS specifically allows CC.

A PUT DATA with CC (bit 5 of the CLA byte set to 1) starts a chain that extends until the first PUT DATA without CC (that is, bit 5 of the CLA byte set to 0) or a failed PUT DATA or any other command. Unless the last PUT DATA is successful, any command in the chain is canceled and memory is recovered. For example, if a chain were used to update multiple DOs through a sequence of chained PUT DATA and the chain fails after updating only the first few DOs, these first DOs would revert to their value before the start of the chain.

The following section describes how to use CC.

### Use of Command Chaining

CC is used as follows:

* Consider the following data field to transmit: Data.
* The data field Data to send is divided into several elementary blocks Data = D1 | D2|…|Dn.
* Each data block D1… Dn-1, has a given length Lblock, and the final one Dn has at most the length Lblock.
* Each elementary block is conveyed by using a chain of commands. Bit 5 of the CLA byte of the command is used to indicate if the APDU conveys a part of the chain or the last part of the chain.
* If the command is a part of the chain, bit 5 of the CLA byte shall be set to 1: CLA.chaining.
* If the command is the last of the chain, bit 5 of the CLA byte shall be set to 0: CLA.last.
* Therefore, in this example the data field Data may be conveyed that way:

CLA.chaining INS P1 P2 Lblock D1

CLA.chaining INS P1 P2 Lblock D2

- - - - - - - -

CLA.last INS P1 P2 L Dn

* Each command of the chain shall be sent in the correct order and shall be consecutive.
* There could be as many commands chained as required to convey all incoming data.
* Each correct command of the chain shall return the SW 0x9000 upon correct reception.
* If the chain is broken because an unexpected command has been sent, all intermediate data that is stored in the internal context of the ICC is lost.

#### Case without Secure Messaging

If the commands are to be sent without secure messaging, the length of block shall be set to a maximum of 0xFF = 255 bytes.

# GIDS Command Set APDU

The following table lists the ISO/IEC 7816 APDU that are supported by the GIDS card-edge.

Table : GIDS APDUs

|  |  |  |
| --- | --- | --- |
| **INS** | **Command name** | **Note** |
| 44 | ACTIVATE FILE |  |
| EO | CREATE FILE |  |
| 24,25 | CHANGE REFERENCE DATA | Allow reference data of variable lengths (that is, different from 8 to allow password). |
| E4 | DELETE FILE |  |
| 47 | GENERATE ASYMMETRIC KEY  PAIR | Includes information on how to encode the optional cryptographic mechanism parameters when the key pair to generate is an RSA key pair. |
| 87 | GENERAL AUTHENTICATE |  |
| CB | GET DATA | Includes a parser to retrieve only part of a structured DO or template. Allows P1-P2 = File identifier as well as P1-P2 = 3F FF. |
| CB | GET PUBLIC KEY | Extension of the GET DATA command to support retrieval of a public key. |
| 88 | INTERNAL AUTHENTICATE | Allows internal authentication schemes without witness. |
| 22 | MANAGE SECURITY ENVIRONMENT |  |
| 2A | PERFORM SECURITY OPERATION | Supports on-card signature generation/verification and encryption/decryption functions with on-card formatting. |
| DB | PUT DATA | Allows loading DOs of any tags. |
| DB | PUT KEY | Extension of the PUT DATA command to support the loading DOs that represent keys. |
| 2C,2D | RESET RETRY COUNTER | Allow referenced data of variable length. Provides the capability to execute the command without resetting code when a secure messaging with mutual authentication has been established. |
| A4 | SELECT | The returned FCI can be customized to meet application needs. Allows SELECT APDU response absent (without FCI or FCP returned) when explicitly requested by the calling application (for example, to speed up transaction over contactless interface). |
| E6 | TERMINATE DF |  |
| 20,21 | VERIFY | Allow referenced data of variable length. Provides the capability to clear the application security status without leaving the application. |

## ACTIVATE FILE

### Description

The ACTIVATE FILE command initiates the transition of a file state from the creation to the operational states.

Activating a correctly created file is always allowed.

P1-P2 = 00 00 and the command data field is absent. Therefore, the command applies only to a file that has been selected by the command that was executed directly before the file.

Access control rules on DOs that are stored in an EF are enforced as soon as the file has been activated. See “Security Architecture.”

### Command APDU

Table : ACTIVATE FILE APDU

|  |  |
| --- | --- |
| **CLA** | 00 |
| **INS** | 44 |
| **P1-P2** | 00 00 Activate current file |
| **Lc** | Absent |
| **Data Field** | Absent |
| **Le** | Absent |

### Status Word

|  |  |
| --- | --- |
| **SW1 SW2** | **Meaning** |
| 67 00 | Wrong Lc |
| 69 82 | Security status not satisfied |
| 6A 82 | File not found (no EF selected) |
| 69 85 | Incorrect P1-P2 |
| 90 00 | Successful execution |

### Conditional Usage

* There are no access conditions to execute ACTIVATE FILE APDU.
* If the file has already been activated, ACTIVATE FILE APDU returns 9000 without triggering any modifications of the LCS file.
* When a key EF that has just been created (that is, PUT KEY or GENERATE ASYMETRIC KEY PAIR is not yet performed on the key EF) is activated by using the ACTIVATE command, the LCS of the key EF enters the operational—deactivated state.
* When a key EF that has already been loaded with a key value (that is, through PUT KEY or GENERATE ASYMETRIC KEY PAIR) is activated, the LCS enters the operational—activated state.
* Only an ACTIVATE command can switch the LCS from the initialization to the operational—activated/deactivated states. The PUT KEY and GENERATE ASYMETRIC KEY PAIR commands do not automatically switch the LCS from an initialization to an operational—activated state.
* The key that is associated with the key EF can be used (such as by GENERAL AUTHENTICATE or PERFORM SECURITY OPERATION) when the key EF's LCS is in the initialization or the operational—activated state.
* Access control rules to load, generate, or use a key are enforced as soon as the key EF’s LCS is in the operational state.
* The ACTIVATE command can be applied to a key EF that has no key in it.

## CREATE FILE

### Description

The CREATE FILE command initiates the creation of an EF directly under the current DF.

After successful creation, the created file becomes the current file and other applicable file-related commands could be issued without specifying a target file. After an EF fails to be created, the current EF is unchanged.

Because GIDS supports only one level of EF in DFs, it is not required to select the DF level between each EF creation.

With the GIDS command set, memory for data need not be allocated during file creation to provide greater flexibility to store DOs. Memory could be taken from the pool that was allocated to the on-card application (DF) whenever a new DO is written into the file or an existing one is updated by using the PUT DATA command.

The GIDS command set has only one DF per application (the application DF or ADF) and does not support the creation of a DF under the ADF.

### Command APDU

Table : CREATE FILE APDU

|  |  |
| --- | --- |
| **CLA** | 00 |
| **INS** | E0 |
| **P1-P2** | 00 00 |
| **Lc** | Length of data field |
| **Data Field** | FCP template for EF; see “Table 12: FCP Template Assignment Data Objects for Data Object and Binary EF” |
| **Le** | Absent |

### Command Data Field

The command data field shall contain the FCP template as described in “Table 12: FCP Template Assignment Data Objects for Data Object and Binary EF” (for DO files and GIDS key files).

The file life cycle (tag 8A) shall not be included in the command data field for the CREATE FILE command.

### Response Data Field

Absent

### Status Word

| **SW1 SW2** | **Meaning** |
| --- | --- |
| 69 82 | Security status not satisfied |
| 69 85 | Condition of use not satisfied |
| 6A 86 | Incorrect P1-P2 |
| 67 00 | Wrong Lc |
| 6A 85 | Lc incompatible with TLV structure (invalid TLV length) |
| 6A 80 | Invalid FCP template |
| 6A 84 | Not enough EEPROM available |
| 6A 89 | File already exists (EFID or SFI already assigned) |
| 90 00 | Successful execution |

### Conditional Usage

The following conditions must be fulfilled:

* The EFID that was specified for the new file must not already be assigned to an existing file in the current DF.
* The EFID must not be a value reserved by ISO (00 00, 2F 00, 2F 01, 3F 00, 3F FF or FF FF).

The command can be performed only if the security status satisfies the security attributes for the current DF. The CREATE FILE command does not support CC:

* CREATE FILE is an atomic command, meaning that if the command is aborted before its completion, the card memory shall revert to its content immediately before the command execution. CREATE FILE with EFID = 2F00, 2F01 is not authorized because these EFIDs are reserved for EF.DIR and EF.ATR and are already present in the card structure.
* GIDS supports only the creation of transparent EFs.
* An invalid FCP template error status is returned in the following cases:

The FCP structure does not comply with BER-TLV encoding rules.

The FCP contains undefined tags.

The FCP is incomplete.

The value of file descriptor byte (tag 82) is not supported by the card.

The value of the security attribute (tag 8C) is not supported by the card.

The value of tag A5 when present is not supported by the card.

## CHANGE REFERENCE DATA

### Description

The CHANGE REFERENCE DATA command initiates the comparison of the password that is stored in the card with the password that is sent from the interface device and then conditionally replaces the password that is stored in the card with the new password that is sent from the interface device.

The CHANGE REFERENCE DATA command is available only for passwords.

### Command APDU

Table : CHANGE REFERENCE DATA APDU

|  |  |
| --- | --- |
| **CLA** | 00 |
| **INS** | 24 |
| **P1** | 00 or 01 |
| **P2** | Reference data ID (see “Table 73: Reference Data Qualifier”) |
| **Lc** | Length of command data field |
| **Data Field** | <password> || <new password> if P1 = 00  <new password> if P1 = 01 |
| **Le** | Absent |

P1 = 01 is available only during the initialization phase to set the reference data that should be available to the application. The value of the reference data can be updated during personalization.

### P2 Parameter

Table : Reference Data ID for CHANGE REFERENCE DATA P2 Parameter

|  |  |
| --- | --- |
| **P2** | **Password** |
| 00 | Card Global Password |
| 80 | Application Password |
| 81 | Application Resetting Password |

### Command Data Field

When INS = 24, the data field is constructed by the representation of the actual password followed without delimitation by the new password.

The length of the existing reference data is known in the card, so that neither a delimiter nor padding for filling up fixed formats is necessary. The length of the new password therefore computes to Lnew = Lc — Lold.

### Response Data Field

Absent

### Status Word

|  |  |  |
| --- | --- | --- |
| **SW1 SW2** | | **Meaning** |
| 63 C*x* | | Comparison failed and *x* tries remain |
| 63 CF | Reference data retry counter non-deterministic | | |
| 69 83 | | Authentication method blocked |
| 6A 80 | | Incorrect parameter in command data field |
| 6A 86 | | Bad parameter P1 or P2 |
| 6A 88 | | Reference data not found |
| 90 00 | | Successful execution |

### Conditional Usage

CHANGE REFERENCE DATA is an atomic command—that is, if the command is aborted before its completion, the card memory state shall revert to the content immediately before the command execution.

The CHANGE REFERENCE DATA command cannot be used to set or change biometric data for match on card. Biometric data are enrolled by using a PUT DATA command.

The result of the comparison updates the security status of the reference data that is being changed.

## DELETE FILE

### Description

The DELETE FILE command initiates the deletion of a file that has been selected by the command that was executed directly before the command.

After successful completion of this command, the deleted file can no longer be selected. The current file after deletion of an EF is the current DF (application) because GIDS supports only one level of DF. The resources that the file holds shall be released and the memory that the file uses shall be set to the logical erased state.

### Command APDU

Table : DELETE FILE APDU

|  |  |
| --- | --- |
| **CLA** | 00 |
| **INS** | E4 |
| **P1-P2** | 00 00 delete current file |
| **Lc** | Absent |
| **Data Field** | Absent |
| **Le** | Absent |

### Status Word

|  |  |
| --- | --- |
| **SW1 SW2** | **Meaning** |
| 67 00 | Wrong L**c** |
| 69 82 | Security status not satisfied |
| 69 85 | Condition of use not satisfied |
| 6A 86 | Incorrect P1-P2 |
| 90 00 | Successful execution |

### Conditional Usage

The deletion of a key EF zeroizes the associated key value in memory. DELETE FILE does not apply to DFs.

## GENERATE ASYMMETRIC KEY PAIR

### Description

The GENERATE ASYMMETRIC KEY PAIR card command initiates the generation of an asymmetric cryptographic key pair and stores it in the card for future use.

The public part of the generated key pair is returned as the response to the command.

The private part is stored in the non-volatile memory of the card. If the key reference that is indicated in P2 already exists, the card updates the previous key.

See “Access Mode Byte for Keys.”

### Command APDU

|  |  |
| --- | --- |
| **CLA** | 00 |
| **INS** | 47 |
| **P1** | 00 |
| **P2** | 00 for key with CRT |
| **Lc** | Length of data field |
| **Data Field** | Control reference template; see “Table 42: GENERATE ASYMMETRIC KEY PAIR Data Field” |
| **Le** | Length of public key of DO template |

### Command Data field

The command data field of GENERATE ASYMMETRIC KEY is a structured BER-TLV that is formatted as shown in the following table.

Table : GENERATE ASYMMETRIC KEY PAIR Data Field

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Tag** | **Len** | **Data** | **Notes** |
| AC | Var. | 80 | 01 | Cryptographic mechanism identifier | Mandatory; see the following table for a key with CRT and for a key without CRT |
|  | | 83 | Var. | Key reference identifier | Mandatory |

#### Cryptographic Mechanism Identifier for Key with CRT

The following table provides the cryptographic mechanism identifier that is supported by GIDS for the generation of an asymmetric key pair.

Table : Cryptographic Mechanism Identifiers for Generation of an Asymmetric Key with CRT

|  |  |  |
| --- | --- | --- |
| **Cryptographic mechanism identifier** | **Description** | **Parameters** |
| 06 | RSA 1024 bit | None |
| 07 | RSA 2048 bit |
| 08 | RSA 3072 bit |
| 09 | RSA 4096 bit |
| 0A | ECC: Curve P-192 | None |
| 0B | ECC: Curve P-224 | None |
| 0C | ECC: Curve P-256 | None |
| 0D | ECC: Curve P-384 | None |
| 0E | ECC: Curve P-521 | None |
| Other values | RFU | |

For RSA, a default public exponent with value 65,537 (216 + 1) is used.

#### Optional Parameters for ECC Keys

For ECC key pair generation, there are no optional parameters to define in the command data field because all parameters can be derived from the ECC curve being used—that is, from the cryptographic mechanism identifier.

### Response Data Field

The response data field contains the public part of the generated key in BER TLV format as described in the following table.

#### For RSA Keys

Table : GENERATE ASYMMETRIC KEY PAIR Response Field for RSA Keys

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **Len.** | **Value** | | |
| 7F49 | Var. | **Public Key Template** (ISO/IEC 7816-8 table 3) | | |
| **Tag** | **Length** | **Value** |
| 81 | BER-TLV Length | Public key modulus |
| 82 | BER-TLV Length  01 to 108 (1024-bit key) or  01 to 228 (2048-bit key) | Public key exponent |

#### For ECC Keys

Table : GENERATE ASYMMETRIC KEY PAIR Response Field for ECC Keys

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **Len.** | **Value** | | |
| 7F49 | Var. | **Public Key Template** (ISO/IEC 7816-8 table 3) | | |
| **Tag** | **Length** | **Value** |
| 86 | BER-TLV Length  01 to 108 (1024-bit key) or  01 to 228 (2048-bit key) | Public Key P (also called Point)  in uncompressed format |

### Status Word

|  |  |
| --- | --- |
| **SW1 SW2** | **Meaning** |
| 61 XX | Successful execution where SW2 encodes the number of response data bytes still available; 6100 means that at least 256 bytes of data are still available and can be retrieved with the next GET RESPONSE APDU |
| 69 82 | Security status not satisfied |
| 6A 80 | Incorrect parameter in command data field such as unrecognized cryptographic mechanism |
| 6A 86 | Cryptographic mechanism of reference data to be generated differently from cryptographic mechanism of reference data of given key reference |
| 90 00 | Successful execution |

### Conditional Usage

GENERATE ASYMETRIC KEY PAIR is an atomic command—that is, if the command is aborted before its completion, the card memory state shall revert to the content immediately before the command execution.

## GENERAL AUTHENTICATE

### Description

The GENERAL AUTHENTICATE card command performs a cryptographic operation such as an authentication protocol by using the data in the data field of the command and returns the result of the cryptographic operation in the response data field.

The GENERAL AUTHENTICATE command is used to authenticate the card or a card application to the off-card application (INTERNAL AUTHENTICATE), to authenticate an external entity to the card (EXTERNAL AUTHENTICATE), and to perform a mutual authentication between the card and an entity external to the card (MUTUAL AUTHENTICATE).

The function can be performed only if the security status satisfies the security attributes for this operation. The result (yes or no) of a control that is performed by the card may conditionally update the security status. The card may record the number of times that the function is issued to limit the number of further uses of the relevant secret or the algorithm. The card may record unsuccessful authentications (such as to limit the number of further uses of the reference data).

### Command APDU

Table : GENERAL AUTHENTICATE APDU

|  |  |
| --- | --- |
| **CLA** | 00 or 10 indicating CC |
| **INS** | 87 |
| **P1-P2** | 00 00 |
| **Lc** | Length of data field |
| **Data Field** | See “Table 47: Data Field of the GENERAL AUTHENTICATE APDU” |
| **Le** | Absent or length of expected response |

### P1-P2 Parameters

P1-P2 = 00 00. A CRT must have been previously set with MANAGE SECURITY ENVIRONMENT.

### Command Data Field

The command data field of the GENERAL AUTHENTICATE command must be encapsulated in a BER-TLV object with tag 7C, whose data field contains one or more BER-TLV objects that conform to the following table. For more information, refer to 7816-4:2005, Table 71: Dynamic Authentication DOs.

Table : Data Field of the GENERAL AUTHENTICATE APDU

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | **Description** |
| 7C | Var. |  | Set of dynamic authentication DOs with the following tags |
| 80 | Var. | Witness | Demonstration of knowledge of a fact without revealing the fact; an empty witness is a request for a witness |
| 81 | Var. | Challenge | One or more random numbers or byte sequences to be used in the authentication protocol |
| 82 | Var. | Response | Sequences of bytes that encode a response step in an authentication protocol |
| 83 | Var. | Committed Challenge | The hash code of a large random number including one or more challenges |
| 84 | Var. | Authentication Code | The hash-code of one or more data field and a witness DO |
| 85 | Var. | Exponential | A positive number for establishing an ephemeral key by a key agreement technique |
| A0 | Var. | Identification | Identification data template |

The following rules apply within the interindustry template for dynamic authentication:

* If a DO is empty in a template, it shall be complete in the template in the next data field.
* In the first command data field, the template indicates the dynamic authentication function as follows:

A witness request, such as an empty witness, denotes an INTERNAL AUTHENTICATE function.

A challenge request, such as an empty challenge, denotes an EXTERNAL AUTHENTICATE function.

The absence of an empty DO denotes a MUTUAL AUTHENTICATE function. Then unless the card aborts the process, the template in the response data field shall contain the same DOs as the template in the command data field. The MUTUAL AUTHENTICATE function allows two entities to agree on an ephemeral key by using a pair of “exponential” data elements that are referenced by tag 85. See key agreement techniques in ISO/IEC 11770-3.

The dynamic authentication may protect data fields that are exchanged during a session. Both entities maintain a current hash-code, updated by including one command or response data field at a time. The DO with tag 84 conveys an authentication code that results from updating the current code by including a witness DO with tag 80. The verifier successively reconstructs a witness and an authentication code. If the reconstructed witness is not zero and if the two codes are identical, the authentication is successful.

### Response Data Field

The response data field is presented in the same way as the command data field—that is, if present, the data is encapsulated in a constructed BER-TLV object with tag 7C.

In case of an ECDSA signature, the response data shall be formatted as follows:

EC Signature:: = SEQUENCE OF {

r INTEGER

s INTEGER

}

### Status Word

|  |  |
| --- | --- |
| **SW1 SW2** | **Meaning** |
| 61 XX | Successful execution where SW2 encodes the number of response data bytes still available; 6100 means that at least 256 bytes of data are still available and can be retrieved with the next GET RESPONSE APDU |
| 69 82 | Security status not satisfied (such as authentication failed) |
| 69 85 | Command not authorized at this time (such as out of sequence or key usage counter reached or no security environment set) |
| 6A 80 | Incorrect data field (badly formatted) |
| 6A 86 | Bad P1-P2 |
| 6A 88 | Reference data is missing |
| 90 00 | Successful execution |

### Conditional Usage

The GENERAL AUTHENTICATE command is available only to keys for which the CRT, as defined in the FCP of the EF where the key is stored, includes authentication. See “Control Reference Template (CRT).”

The SE must have been previously set with the MSE-SET APDU.

As stated in ISO/IEC 7816-4, a witness must be used when the GENERAL AUTHENTICATE command is to perform an internal authentication.

Witness, challenge, and response are BER -TLV encoded and must comply with the following encoding rules:

* Only lengths coded on 1 to 3 bytes are accepted.
* No “minimal coding” is enforced—that is, a length of 80 can be coded as 82 00 80 or as 81 80.

Some authentication schemes require more than one GENERAL AUTHENTICATE command. If a command other than a GENERAL AUTHENTICATE is received by the application before the termination of the chain, the application rolls back to the state it was before the first GENERAL AUTHENTICATE command.

As described in ISO 7816-4:2005 section 7.5.5, the GENERAL AUTHENTICATE card command refines the EXTERNAL, INTERNAL, and MUTUAL AUTHENTICATE functions. It has not been designed to perform encryption or signature. For these functions, the ISO/IEC 7816-8 PERFORM SECURITY OPERATION (PSO) that was described in “PERFORM SECURITY OPERATION” must be used. The use of authentication code 84 allows enforcing the role separation. For further details, refer to ISO/IEC 7816-4.

CC may be available only on cards that do not support extended length as per ISO/IEC 7816-3. Card capability to support extended-length APDUs can be retrieved from the third byte of the card capability DO that is stored in EF.ATR.

## GET DATA

### Description

The GET DATA card command retrieves the data content of the DOs whose tag is given in the data field.

### Command APDU

Table : GET DATA APDU

|  |  |
| --- | --- |
| **CLA** | 00 |
| **INS** | CB |
| **P1-P2** | 3F FF, 00 00, or file identifier |
| **Lc** | Length of data field |
| **Data Field** | See “Table 49: GET DATA Command Data Field” |
| **Le** | Number of data content bytes to be retrieved |

### P1-P2 Parameters

If P1-P2 = 3F FF, GET DATA searches for application DOs in all EFs in which life cycle phase is operational. It also searches for global DOs that are stored at the MF level regardless of the DF that is currently selected because global DOs are made available to all on-card applications. This is the only value of P1-P2 that allows retrieving global DOs that are stored at the MF level.

If P1-P2 = 00 00, GET DATA searches for application DOs in the current EF.

If P1-P2 provides a file identifier, the GET DATA command searches only in the specified file. If the file identifier that is provided in P1-P2 is found, this file becomes the currently selected EF even if no data is available.

### Command Data Field

Table : GET DATA Command Data Field

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | **Comment** |
| 5C | Length of value field | Tag of the DO to retrieve | Only one tag list per command data field is supported by GIDS (5C). A maximum of one tag is allowed for the tag list (5C). |

### Response Data Field for BER-TLV Data Object

For BER-TLV DOs (that is, DOs stored in a DO EF) the response data field is formatted as shown in the following table.

Table : Response Data Field for BER-TLV Data Object

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | **Comment** |
| Tag of the  DO requested | Length of the value field of the DO | Value field of the requested DO | The DO is returned as a whole and encoded in compliance with BER-TLV rules. |

### Status Word

|  |  |
| --- | --- |
| **SW1 SW2** | **Meaning** |
| 61 XX | Successful execution where SW2 encodes the number of response data bytes still available; 6100 means that at least 256 bytes of data are still available and can be retrieved with the next GET RESPONSE APDU |
| 62 82 | End of file or record reached before reading Le bytes |
| 69 82 | Security status not satisfied |
| 6A 80 | Incorrect parameters in the command data field |
| 6A 82 | File or application not found |
| 6A 88 | Referenced data not found |
| 69 85 | Condition of use not satisfied |
| 90 00 | Successful execution |

### Conditional Usage

The retrievable data is under the access conditions from the EF into which they are stored. See “Security Architecture.”

GET DATA command data must be a tag list or extended header list, which indicates a single DO type.

A successful execution of GET DATA automatically selects the EF whose EFID is listed in the command parameters and *not* necessary the EF in which the DO is actually stored because the response could include DOs from multiple EFs.

GET DATA with 5C00 in the command data field retrieves all DOs that are available in the indicated file in P1-P2 according to the current access conditions that were established in the card-application at the time of request. When used with P1-P2 = 3F FF, GET DATA gets all available DOs from the current DF (whole card application) in one command. However, remember that a DO that is referenced in multiple templates may be returned more than once.

GET DATA with 5C00 or 5C015C in the command data field processes binary as if they were BER DOs.

If more than one accessible DO matches the requested type, all DOs are returned concatenated without wrapping.

GET DATA with P1-P2 = 3F FF does not return a reference DO. This allows avoiding duplicate DOs in the return data field.

If the GET DATA command cannot return any DO because the requested DOs are all unavailable because of access control rules, the command shall fail with a status 6982 (security status not satisfied) and the currently selected EF shall remain unchanged.

If the GET DATA command cannot return any DO because the requested DOs do not exist in the file that is specified by P1-P2, the command shall fail with a status 6A88 (referenced data not found) and the currently selected EF shall remain unchanged. But if at least one DO can be returned with the current access condition, it is returned followed by a status 9000 and the currently selected EF shall be updated accordingly.

GET DATA on an empty DO returns the DO with a length set to zero.

## GET PUBLIC KEY

### Description

The GET PUBLIC KEY is a GET DATA command with a special command data field that allows retrieval of a public key to request a certificate to be generated.

### Command APDU

See “GET DATA Command APDU.”

### Command Data Field

The following table provides the command data field to retrieve a public key.

Table : Command Data field to Retrieve a Public Key Value

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | | | | | **Comment** |
| A3 | Var. | 84 | 01 | **Key References** | | |  |
|  | | A5 | 03 |  | | | Key value template |
|  | | 7F49 | 80 | Empty | Specifies that the public key part of a key pair is to be retrieved (length = 80 because 7F49 is a constructed tag) |

### Response Data Field

The following table provides the response data field.

Table : Response Data Field with Public Key Value

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | **Comment** |
| 7F49 | Var. | See “Table 44: GENERATE ASYMMETRIC KEY PAIR Response Field for RSA Keys” and “Table 45: GENERATE ASYMMETRIC KEY PAIR Response Field for ECC Keys” | Template that contains the cardholder's public key DOs for digital signature functionality using asymmetric mechanisms (defined in ISO/IEC 7816-8) |

### Status Word

|  |  |
| --- | --- |
| **SW1 SW2** | **Meaning** |
| 61 XX | Successful execution where SW2 encodes the number of response data bytes still available; 6100 means that at least 256 bytes of data are still available and can be retrieved with the next GET RESPONSE APDU |
| 69 82 | Security status not satisfied |
| 6A 80 | Incorrect parameters in the command data field |
| 69 85 | Condition of use not satisfied |
| 90 00 | Successful execution |

### Conditional Usage

Only public key values can be retrieved. The retrievable data are under the access conditions from the EF that is associated with the key. See “Security Architecture.”

## INTERNAL AUTHENTICATE

### Description

The INTERNAL AUTHENTICATE card command initiates the computation of authentication data by the card by using the challenge data sent by the interface device and a relevant secret (such as a key) stored in the card.

### Command APDU

Table : INTERNAL AUTHENTICATE APDU

|  |  |
| --- | --- |
| **CLA** | 00 |
| **INS** | 88 |
| **P1-P2** | 00 00 |
| **Lc** | Length of data field |
| **Data Field** | See “Table 46: GENERAL AUTHENTICATE APDU” |
| **Le** | Absent or length of expected response |

### Command Data Field

The command data field of the INTERNAL AUTHENTICATE command must be encapsulated in a BER-TLV object with tag 7C, whose data field contains one or more BER-TLV objects that conform to the following table.

Table : Command Data Field of the INTERNAL AUTHENTICATE APDU

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | | | | **Description** | | |
| 7C | Var. |  | | | | Set of dynamic authentication DOs with the following tags | | |
|  | | **Tag** | **Len** | **Value** | | **Description** | | |
| 81 | Var. | Chal­lenge | | One or more random numbers or byte sequences to be used in the authentication protocol | | |
| 85 | Var. | Expo­nential | | A positive number for establishing an ephemeral key by a key agreement technique; this DO is optional and used only for internal authentication protocol 9.4 | | |
| A0 | Var. | Identi­fication | | Identification data template; this DO is optional and used only for internal authentication protocol 9.4 | | |
|  | | | | **Tag** | **Len** | | **Value** | **Description** |
| 80 | Var. | | Identity |  |
| 81 | Var. | | Nonce |  |

### Response Data Field

The response data field is constructed as shown in the following table.

Table : Response Data Field of the INTERNAL AUTHENTICATE APDU

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | | | | **Description** | | |
| 7C | Var. |  | | | | Set of dynamic authentication DOs with the following tags | | |
|  | | **Tag** | **Len** | **Value** | | **Description** | | |
| 82 | Var. | Response | | Sequences of bytes that encode a response step in an authentication protocol | | |
| A0 | Var. | Identification | | Identification data template; this DO is optional and used only for internal authentication protocol 9.4 | | |
|  | | | | **Tag** | **Len** | | **Value** | **Description** |
| 80 | Var. | | Identity |  |
| 81 | Var. | | Nonce |  |

**Note:**

In case of an ECDSA signature, the response data shall be formatted as follows:

EC Signature:: = SEQUENCE OF {  
r INTEGER  
s INTEGER  
}

### Status Word

|  |  |
| --- | --- |
| **SW1 SW2** | **Meaning** |
| 61 XX | Successful execution where SW2 encodes the number of response data bytes still available; 6100 means that at least 256 bytes of data are still available and can be retrieved with the next GET RESPONSE APDU |
| 69 82 | Security status not satisfied (such as authentication failed) |
| 69 85 | Command not authorized at this time (such as out of sequence or key usage counter reached or no security environment set) |
| 6A 80 | Incorrect data field (badly formatted) |
| 6A 86 | Bad P1-P2 |
| 6A 88 | Reference data missing |
| 90 00 | Successful execution |

### Conditional Usage

Unlike GENERAL AUTHENTICATE when used to perform an internal authentication function, the INTERNAL AUTHENTICATE APDU does not require a witness from the card.

## MANAGE SECURITY ENVIRONMENT

### Description

MANAGE SECURITY ENVIRONMENT (MSE) prepares the GENERAL AUTHENTICATE and PERFORM SECURITY OPERATION security commands.

### Command APDU

Table : INTERNAL AUTHENTICATE APDU

|  |  |
| --- | --- |
| **CLA** | 00 or 10 indicating CC |
| **INS** | 22 |
| **P1** | x1—see the following section |
| **P2** | Tag of the control reference template present in the command data field |
| **Lc** | Length of data field |
| **Data Field** | Concatenation of control reference DOs |
| **Le** | Absent for encoding Ne = 0 |

### P1Parameter

Only the values of P1 in the following table are supported.

Table : MSE P1 Parameter

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **Meaning** |
| - | - | 1 | 1 | - | - | - | - | RFU |
| - | 1 | - | - | - | - | - | - | Computation, decipherment, internal authenticate, and key agreement |
| 1 | - | - | - | - | - | - | - | Verify, encipherment, external authenticate, and key agreement |
| - | - | - | X | 0 | 0 | 0 | 1 | SET |

### P2 Parameter

The P2 parameter is the tag of the control reference template to set. Only values in “Table 23: CRT Tags” are supported.

### Command Data Field

The command data field is the concatenation of the control reference DOs to set.

GIDS supports only one CRT per command. That CRT remains active until another CRT of the same type (same tag) is set.

The following table, extracted from ISO/IEC 7816-4:2005 table 33, provides the control reference DO that is supported by GIDS for the MSE-SET command.

Table : Control Reference Data Objects in Control Reference Template for MSE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | **AT** | **DST** | **CT** |
| 80 | 01 | Cryptographic mechanism reference (AlgoID) | X | X | X |
| 83 | 01 | Reference of a secret key (for direct use) or reference of a public key if applicable | X | X | X |
| 84 | 01 | Reference of a private key if applicable | X | X | X |

The presence of a cryptographic mechanism reference (tag 80) to set a CRT of type AT is optional. If absent, the default value that is defined in “Table 25: Cryptographic Mechanism Reference for AT CRT” is used.

### Status Word

|  |  |
| --- | --- |
| **SW1 SW2** | **Meaning** |
| 69 82 | Security status not satisfied |
| 6A 86 | Incorrect P1-P2 |
| 67 00 | Wrong Lc |
| 90 00 | Successful execution |

### Conditional Usage

Only the SET command is supported (P1 = x1), that is, the command can be used only to set or replace one component of the current SE.

CC may be available only on cards that do not support extended length per ISO/IEC 7816-3. Card capability to support extended-length APDUs can be retrieved from the third byte of the card capability DO that is stored in EF.ATR.

## PERFORM SECURITY OPERATION

### Description

The PERFORM SECURITY OPERATION (PSO) command initiates the following security operations, according to the DOs that are specified in P1-P2:

* Computation of a cryptographic checksum.
* Computation of a digital signature.
* Calculation of a hash-code.
* Verification of a cryptographic checksum.
* Verification of a digital signature.
* Encipherment.
* Decipherment.

If the security operation requires several commands to complete, CC shall apply.

### Command APDU

Table : PSO APDU

|  |  |
| --- | --- |
| **CLA** | 00 or 10, which indicates CC |
| **INS** | 2A |
| **P1** | Tag (the response data field is the data element, if present) or 00 (the response data field is always absent) |
| **P2** | Tag (the command data field is the data element, if present) or 00 (the command data field is always absent) |
| **Lc** | Absent for encoding Nc = 0, present for encoding Nc > 0 |
| **Data Field** | Absent or value of the DO specified in P2 |
| **Le** | Absent for encoding Ne = 0, present for encoding Ne > 0 |

### P1-P2 Parameters

The combinations of P1‑P2 in the following table are supported.

Table : PSO P1-P2 Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function** | **P1** | **P2** | **Command data field** | **Response data field** |
| PSO: COMPUTE DIGITAL SIGNATURE | 9E | 9A | DER-encoded digest information (PKCS 1.5 only) || data to be signed (precomputed hash value) | Plain signature without further TLV coding |
| PSO: ENCIPHER | 86 | 80 | Data to be enciphered (plain value) | Enciphered value without further TLV coding |
| PSO: DECIPHER | 80 | 86 | Data to be deciphered  (Pl || cryptogram) | Deciphered value without further TLV coding |

### Command Data Field

See “Table 60: PSO P1-P2 Parameters.”

### Status Word

|  |  |
| --- | --- |
| **SW1 SW2** | **Meaning** |
| 69 82 | Security status not satisfied |
| 6A 86 | Incorrect P1-P2 |
| 67 00 | Wrong Lc |
| 90 00 | Successful execution |

### Conditional Usage

The PERFORM SECURITY OPERATION command is available only in the modes that are authorized by the CRT, as defined in FCP of the EF where the key is stored. See “Cryptopgraphic Algorithms”.

GIDS does not define default algorithm. As a result, the key to use with a PSO command must have been explicitly specified initially with a MSE SET command.

The final value and all intermediate values are cleared after the completion of the last command of the chain.

When performing a decryption, the format of the padding is verified by the card. If the verification fails, the response data field is absent and an error status is returned.

When the security operation to perform requires multiple PSO commands, CC shall be used. Intermediate results are not returned by the card when PSO is used in CC. The response data field is present only outside CC.

CC may be available only on cards that do not support extended length as per ISO/IEC 7816-3. Card capability to support extended-length APDUs can be retrieved from the third byte of the card capability DO that is stored in EF.ATR

## PUT DATA

### Description

The PUT DATA card command creates or replaces the contents of a single DO in the current application.

### Command APDU

Table : PUT DATA APDU

|  |  |
| --- | --- |
| **CLA** | 00 or 10 indicating CC |
| **INS** | DB |
| **P1-P2** | 3F FF, 00 00 or file Identifier |
| **Lc** | Length of data field |
| **Data Field** | See “P1-P2 Parameters” |
| **Le** | Absent |

If the administration usage requires more than one PUT DATA command—that is, the length of the data to be stored in the card (including mandatory encapsulation and secure messaging if required) is greater than the size of the I/O buffer (as declared in the EF.ATR), the off-card application must use CC. See “Command Chaining.”

### P1-P2 Parameters

P1-P2 = EFID where the DO is to be created, or 00 00 for current EF. If the DO is to be updated and there is no ambiguity on its location within the application (DF), P1-P2 can also be set to 3F FF. Otherwise, a status error shall be returned.

### Command Data Field

For BER-TLV DO (that is, DOs stored in a DO EF), the command data field complies with ISO/IEC 7816-4 and is formatted as shown in the following table.

Table : Data field of PUT DATA Command for a BER-TLV data object

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | **Comment** |
| Tag of the DO to store or extended header list | Length of the DO value field to store | Absent or value of the DO | GIDS does not support concatena­tion of DOs in the command data field |

### Response Data Field

The PUT DATA command does not return a data field.

### Status Word

|  |  |
| --- | --- |
| **SW1 SW2** | **Meaning** |
| 69 82 | Security status not satisfied |
| 6A 80 | Incorrect parameters in the command data field |
| 6A 81 | Function not supported |
| 6A 84 | Not enough memory space |
| 6A 85 | Nc inconsistent with TLV structure |
| 69 85 | Condition of use not satisfied |
| 6A 86 | Incorrect parameter in P1-P2 |
| 6A 88 | Referenced key does not exist |
| 90 00 | Successful execution |

### Conditional Usage

The access condition is linked to the EF in which the DO is being written. See “Security Architecture.”

The command data in PUT DATA may be a tag with no value. In that case, a DO with empty value is created. Also note that the very same PUT DATA issued a second time would delete the empty DO. See “Single DO deletion.”

DOs can be created only in a valid EF. A valid EF must be referenced by the P1-P2 parameters to create a new DO. However, the DO can be created with a value length set to zero.

The value of an existing DO can be updated even if P1-P2 = 3F FF if the DO tag that was supplied in the command data field uniquely identifies the DO (such as not having two first-level DOs with the same tag in two different EFs).

The PUT DATA command guarantees the atomicity of the update. If the command is aborted before its completion, the card memory shall revert to its content immediately before command execution.

By using CC, it is possible to load, update, or delete multiple objects within a single atomic transaction. However, a command data field cannot provide data from two different DOs. A new APDU within the chain should start with every DO. If any of the operations in the chain fails, the complete set of updates is canceled and previous values, if any, remain unaffected. When the command is chained, only the first command of the chain starts its data field with the tag (DO, 5C). For all subsequent commands, the data field starts with the remaining part of the actual data to be stored.

When CC is to be used, the full DO tag name must be present in the first APDU. DO value may be divided anywhere and in as many parts as required.

An existing DO can be deleted by issuing a PUT DATA of that DO with no value—that is, with an object BER-TLV length set to zero.

A PUT DATA command, with a DO that does not already exist and a DO length set to zero, creates a new empty DO in the file that is specified in P1-P2. This allows subsequent use of PUT DATA with P1-P2 = 3F FF to personalize the newly created DO regardless of the EF in which the DO is actually stored.

Status 6A 84 shall be returned when not enough memory is available to perform the atomicity of the PUT DATA instruction. This can be solve by the middleware by performing a PUT DATA of the same DO but with a length set to zero to erase the DO in the card first and then reissuing the PUT DATA with the initial value. If CC was used to store multiple DOs in one atomic transaction, the problem could be solved by reducing the number of DOs in one atomic transaction.

To be consistent with the GET DATA command, a successful execution of PUT DATA automatically selects the EF whose EFID is listed in the command parameters and *not* necessarily the EF in which the DO is actually stored (such as the reference DO in PUT DATA).

To avoid data corruption, a PUT DATA command on a reference that truncates the size of the referred DO updates the full DO, regardless of the truncation that was introduced by the reference.

## PUT KEY

### Key Usage Template

The GIDS command set supports the use of the ISO/IEC 7816-4 key usage template to inject application keys into the card by using the PUT DATA command. Therefore, the PUT KEY command is actually a PUT DATA APDU with P1-P2 = 3F FF.

The PUT KEY command data field is a key usage template as defined in the following table.

Table : Data Field of PUT DATA Command to Load Keys

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | **Comment** |
| 70 | Var. | Key usage template | See “Table 64: Key Usage Template” |

The key usage template is formatted as shown in the following table.

Table : Key Usage Template

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | **Comment** |
| 84 | 01 | Key Reference | Shall be unique in the DF, and an EF.KEY with that key reference should exist |
| A5 | Var. | Key value template (constructed) | See “Key Value Templates” |

If the key value template A5 has a length set to zero, the key referenced is zeroized.

### Key Value Templates

The key value template is formatted as shown in the following table to allow for unencrypted key import or compatibility with PKCS#11 encryption mechanisms for key import.

Table : Key Value Template for PKCS#11 Key Import

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag** | **Len** | **Value** | **Comment** |
| 83 | 01 | Key type | Type of key being imported: 01 = symmetric; 02 = RSA; 03 = ECC |
| 84 | 01 | Key reference | Reference of the transport key. Use 0x00 if key material is not encrypted. If not present, KEK from global platform is used. |
| 87 | Var. | Key value | If transport key is specified or not present, PKCS#11 encrypted value of the key to import. If transport key is 0x00, it is the unencrypted value of the key. |
| 88 | 03 | Key check value | Key check value (for symmetric keys only) |

#### Tag 83

The key type that is provided with tag 83 shows information on the format of the key value that is being imported.

##### Symmetric Key

For symmetric key (type = 01), the key value is used without formatting.

##### RSA Key Pair

For RSA key Pair (type = 02), the key value is encoded as follows:

RSAPrivateKey ::= SEQUENCE {

version Version,

modulus INTEGER, -- n

publicExponent INTEGER, -- e

privateExponent INTEGER, -- d

prime1 INTEGER, -- p (“p” shall be the smallest of the two primes “p” and “q”)

prime2 INTEGER, -- q

exponent1 INTEGER, -- d mod (p-1)

exponent2 INTEGER, -- d mod (q-1)

coefficient INTEGER, -- (inverse of q) mod p

otherPrimeInfos OtherPrimeInfos OPTIONAL

}

The following is an example of encoding for an RSA key pair:

30 82 01 36

02 01 00 version = 0

02 40 modulus = n

0a 66 79 1d c6 98 81 68 de 7a b7 74 19 bb 7f b0

c0 01 c6 27 10 27 00 75 14 29 42 e1 9a 8d 8c 51

d0 53 b3 e3 78 2a 1d e5 dc 5a f4 eb e9 94 68 17

01 14 a1 df e6 7c dc 9a 9a f5 5d 65 56 20 bb ab

02 03 01 00 01 publicExponent = e

02 40 privateExponent = d

01 23 c5 b6 1b a3 6e db 1d 36 79 90 41 99 a8 9e

a8 0c 09 b9 12 2e 14 00 c0 9a dc f7 78 46 76 d0

1d 23 35 6a 7d 44 d6 bd 8b d5 0e 94 bf c7 23 fa

87 d8 86 2b 75 17 76 91 c1 1d 75 76 92 df 88 81

02 20 prime1 = p

33 d4 84 45 c8 59 e5 23 40 de 70 4b cd da 06 5f

bb 40 58 d7 40 bd 1d 67 d2 9e 9c 14 6c 11 cf 61

02 20 prime2 = q

33 5e 84 08 86 6b 0f d3 8d c7 00 2d 3f 97 2c 67

38 9a 65 d5 d8 30 65 66 d5 c4 f2 a5 aa 52 62 8b

02 20 exponent1 = d mod p-1

04 5e c9 00 71 52 53 25 d3 d4 6d b7 96 95 e9 af

ac c4 52 39 64 36 0e 02 b1 19 ba a3 66 31 62 41

02 20 exponent2 = d mod q-1

15 eb 32 73 60 c7 b6 0d 12 e5 e2 d1 6b dc d9 79

81 d1 7f ba 6b 70 db 13 b2 0b 43 6e 24 ea da 59

02 20 coefficient = q-1 mod p

2c a6 36 6d 72 78 1d fa 24 d3 4a 9a 24 cb c2 ae

92 7a 99 58 af 42 65 63 ff 63 fb 11 65 8a 46 1d

##### ECC Key Pair

For ECC key Pair (type = 03), the key value is coded as follows:

ECPrivateKey ::= SEQUENCE {

Version INTEGER { ecPrivkeyVer1(1) }

(ecPrivkeyVer1),

privateKey OCTET STRING,

pointP OCTET STRING,

}

The point P must be in uncompressed format.

#### Tag 84

After the value of the key is encoded as shown in the preceding example, the value to import into the card is encrypted with the transport key that is referenced by tag 84 within A5.

When the transport key is a symmetric key, ISO/IEC 9797 Padding Method 2 shall be applied to the key value before encryption.

When the transport key is an RSA key, RSAES-OAEP padding shall be applied to the key value before encryption.

#### Tag 88

The key value check (KVC) is computed by encrypting binary zeros with the key and keeping the first 3 bytes of the result (MSB).

It is present only for symmetric keys. For asymmetric key, the card shall perform a pair-wise consistency check on the key pair that is being imported.

## RESET RETRY COUNTER

### Description

The RESET RETRY COUNTER card command resets the reference data retry counter to its initial value and replaces the reference data with new reference data.

The RESET RETRY COUNTER command allows the execution of the command without resetting code when a secure messaging with mutual authentication has been established.

After a successful presentation of the resetting code (PUK), the retry counter of the referenced password is automatically reset to its initial value.

### Command APDU

Table : RESET RETRY COUNTER APDU

|  |  |
| --- | --- |
| **CLA** | 00 |
| **INS** | 2C |
| **P1** | 00 |
| **P2** | Reference data qualifier to reset; see “Table 40: Reference Data ID for CHANGE REFERENCE DATA P2 Parameter” |
| **Lc** | (m +n) |
| **Data Field** | Resetting code (PUK) followed without delimitation by new reference data if P1 = 00 |
| **Le** | Absent |

The security status *shall not* be reset after this command.

### Command APDU when External or Mutual Authentication with an Administrative Key is used as authentication method

If the Reference Data ID specified in P2 parameter is the *Application Password* and Local PUK status is not configured, External or Mutual authentication performed using GENERAL AUTHENTICATE with the Administrative Key is used for authentication. The command APDU is as follows:

Table : RESET RETRY COUNTER APDU FOR EXTERNAL OR MUTUAL AUTHENTICATION WITH AN ADMINISTRATIVE KEY

|  |  |
| --- | --- |
| **CLA** | 00 |
| **INS** | 2C |
| **P1** | 02 (data field contains new data) |
| **P2** | 80 (Application PIN) |
| **Lc** | Length of the new reference data |
| **Data Field** | New reference data |
| **Le** | Absent |

The security status *shall* be reset after this command.

### Command Data Field

If P1 = 00, the data field is constructed by the representation of the resetting code followed without delimitation by a new value for the password that is being reset. The length of the resetting code is known in the card, so that neither a delimiter nor padding for filling up fixed formats is necessary. The length of the new password therefore computes Lnew = Lc — LResetting code.

### Status Word

|  |  |  |
| --- | --- | --- |
| **SW1 SW2** | **Meaning** | |
| 63 C*x* | Verification failed; *x* indicates the number of further allowed retries upon resetting code verification | |
| 63 CF | Reference data retry counter non-deterministic |
| 69 83 | Authentication method blocked | |
| 6A 80 | Incorrect parameter in command data field | |
| 6A 86 | Incorrect P1-P2 parameters | |
| 69 82 | Security status not satisfied | |
| 69 85 | Condition of use not satisfied | |
| 6A 88 | Reference data missing | |
| 90 00 | Successful execution | |

### Conditional Usage

The reference data that is being replaced must already exist.

If the card command fails, the security status of the resetting code is set to FALSE and the reset counter that is associated with the resetting code is decremented by one.

If the verification failed on the last attempt, the card returns 63C0. If the number of remaining tries is already equal to 0 before the APDU is sent, the card returns the status word 6983.

The maximum consecutive number of unsuccessful attempts to reset a given reference data may be defined during the application initialization (prepersonalization).

A successful execution resets both reference data and resetting code presentation counters. The RESET RETRY COUNTER command is available regardless of whether the associated retry counter is blocked.

## SELECT

### Description

The SELECT command of the GIDS command set allows the selection of an application in the card by providing its AID or right-truncated AID.

While within an application that uses the GIDS command set, SELECT can be used to select an EF by its EFID or to select the parent DF (root of the application).

The SELECT command shall always be realized in clear mode (no secure messaging).

A GIDS application may be default-selected during the card power-on. In this case, the ATR shall specify the AID of the default-selected application as per ISO/IEC 7816.

### Command APDU

Table : SELECT APDU

|  |  |
| --- | --- |
| **CLA** | 00 |
| **INS** | A4 |
| **P1** | See “Table 69: SELECT P1 Parameter” |
| **P2** | See “Table 70: SELECT P2 Parameter” |
| **Lc** | Length of application identifier or EFID |
| **Data Field** | Application identifier (full or partial AID) or EFID |
| **Le** | Length of expected returned data or absent if P2 =0C |

### P1 Parameter

Table : SELECT P1 Parameter

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **Meaning** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SELECT DF or EF with EFID |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | SELECT by DF name (matching a [truncated] application AID) |

### P2 Parameter

Table : SELECT P2 Parameter

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **Meaning** |
| 0 | 0 | 0 | 0 | - | - | 0 | 0 | First or only occurrence |
| 0 | 0 | 0 | 0 | - | - | 1 | 0 | Next occurrence |
| 0 | 0 | 0 | 0 | 0 | 0 | - | - | Return FCI template without use of FCI tag and length |
| 0 | 0 | 0 | 0 | 0 | 1 | - | - | Return FCP template, mandatory use of FCP tag and length |
| 0 | 0 | 0 | 0 | 1 | 0 | - | - | Return FMD template, mandatory use of FMD tag and length |
| 0 | 0 | 0 | 0 | 1 | 1 | - | - | No response data if Le field is absent |

### Command Data Field

The command data field contains the full AID or a right-truncated AID of the application to select or the EFID of the DF or EF to select.

When the command data field is set to EF ID = 00 00, the returned information is related to the currently selected EF if any. An error code 6A 82 is returned if no EF is currently selected.

When the command data field is set to EF ID = 3F FF, the returned information is related to the currently selected DF.

The use of the MF reference (EF ID = 3F 00) in the SELECT command is not defined in this specification. Any use of such reference would lead to implementations not interoperable with this specification. For interoperability purposes, the MF may not be selectable

The use of the SELECT command with absent data field is not supported.

### Response Data Field

If P2=00, the FCI of the selected entity is returned (without tag 6F). If the entity that is being selected is a DF, the returned FCI is made up of the application template DO. If the entity that is being selected is an EF, the returned FCI is the EF FCP.

If P2 = 04, the FCP of the selected entity is returned. See “FCP Templates.”

If P2 = 08, the response data field contains the FMD of the DF (see “FMD Template”) or an empty FMD template (64 00) if an EF was selected (P1 = 02).

If P2 = 0C, no data is returned.

### Status Word

|  |  |
| --- | --- |
| **SW1 SW2** | **Meaning** |
| 62 85 | Selection of a termination state DF |
| 6A 82 | File or application not found; current status left unchanged |
| 6A 86 | P1 P2 combination not supported |
| 6A 87 | Incorrect data field length |
| 90 00 | Successful execution |

### Conditional Usage

No prerequisites are required. The SELECT APDU is always accepted if it is performed with the complete or right-truncated AID.

A SELECT APDU with the AID or a right-truncated AID of the currently selected application does *not* change the currently selected application or any of its security status. It nevertheless always resets the current selected EF to a ”No selected EF” status.

The use of the SELECT command with b2–b1 of P2 set to 1–0 (next occurrence) can be used for the external application to retrieve the FCP/FMD of all the files that are present in the application (discovery mechanism).

SELECT Previous and SELECT Last are not supported by GIDS.

SELECT of the currently selected DF (DF by Name or FID with 3F FF) does not reset the security status.

Selection of DF other than current DF (EFID = 3F FF) can be done only by using Name. GIDS DFs should never be selected by using an EFID because such behavior is not defined in this specification and could lead to incompatible behaviors.

Selection by short EF is not supported (GIDS does not support short EF).

Selection by path is not supported (DFs do not have EFID).

Application selection with partial name is always supported as stated in EF.ATR (see “ACD”).

### Channel Selection

A channel may be implicitly selected with a CLA byte of the SELECT command.

Explicit selection with the MANAGE CHANNEL command is not supported.

GIDS does not support concurrent selections of the same application through different logical channels. Therefore, selecting an application that is already selected on a different logical channel shall automatically deselect the application on the previously used logical channel and reset local security status.

## TERMINATE DF

### Description

The TERMINATE DF command initiates the irreversible transition of a DF (current application) into the termination state. After successful completion of the command, the DF is in a terminated state and the available functionality from the DF and its subtree is reduced. The DF shall be selectable and, if selected, the warning status SW1-SW2 = 6285 (selected file in termination state) shall be returned.

**Note:** The intent of DF termination is generally to make the application unusable by the cardholder.

For security reasons, the same functionality may be achieved by proprietary means.

GIDS supports only the TERMINATE DF that is used with P1-P2 = 00 00 and an absent command data field, which means that the command applies to the DF that has been selected by the command that was executed directly before the command executed. Other values of P1-P2 (EFID or 3FFF) are not supported.

According to ISO/IEC 7816-9 section 6.5, “Terminate DF command,” secure messaging should be used. If the response APDU is not protected by secure messaging, the way to check that the function has been properly executed is not defined within the scope of ISO/IEC 7816. However, GIDS enforces only security conditions that are defined in the DF AMB.

### Command APDU

Table : TERMINATE DF APDU

|  |  |
| --- | --- |
| **CLA** | 00 or 0C if within secure messaging |
| **INS** | E6 |
| **P1-P2** | 00 00 terminate current DF |
| **Lc** | Absent |
| **Data Field** | Absent |
| **Le** | Absent |

### Status Word

|  |  |
| --- | --- |
| **SW1 SW2** | **Meaning** |
| 67 00 | Wrong L**c** |
| 69 82 | Security status not satisfied |
| 69 85 | Condition of use not satisfied |
| 6A 86 | Incorrect P1-P2 |
| 90 00 | Successful execution |

### Conditional Usage

Terminate DF is not available on the MF.

The following restrictions apply to a terminated DF:

* Cryptographic functions are no longer available.
* Modification of a DO is no longer authorized.
* Only DOs that have free access in reading can be retrieved by using the GET DATA command.

## VERIFY

### Description

The command initiates the comparison in the card of stored reference data with verification data sent from the interface device (such as knowledge-based information).

For knowledge-based information, the reference data is a number (PIN), a word (password), or even a sequence of words or sentence (passphrase).

In the scope of this document, PIN, password, and passphrase are used equally to knowledge-based information that the cardholder submits to the card as a witness of its identity.

If verification is successful, the security status that is associated with the reference data is set to verified.

If verification fails, the retry counter that is associated with the reference data that is being verified is decremented. Verification is no longer possible when the value of this counter reaches zero.

### Command APDU

Table : VERIFY APDU

|  |  |
| --- | --- |
| **CLA** | 00 or 10, which indicates CC |
| **INS** | 20 |
| **P1** | 00 |
| **P2** | Reference data qualifier; see “Table 73: Reference Data Qualifier” |
| **Lc** | Absent for encoding Nc=0, present for encoding Nc>0 |
| **Data Field** | Password or absent (INS=20) |
| **Le** | Absent for encoding Ne = 0 |

### P2 Parameter

The P2 parameter indicates the reference data qualifier. The values in the following table are supported by the GIDS card-edge.

Table : Reference Data Qualifier

|  |  |  |
| --- | --- | --- |
| **INS** | **P2** | **Meaning** |
| 20 | 00 | Card global password |
| 20 | 01 | RFU |
| 20 | 80 | Application password (optional) |
| 20 | 81 | Application resetting password (optional) |
| 20 | 82 | Application security status resetting code |
| 21 | 00 | No information provided by P2; information may be retrieved from the BER-TLV structure of the command data field |

### Application Security Status Resetting Code

P2 = 82 defines a special reference data called “Application security status resetting code.” This reference data is always successfully verified regardless of the value in the command data field. A verification of this reference data clears the application security status. The command data field shall be ignored when P2 = 82.

### Command Data Field

When INS = 20, the data field is constructed by the n-byte representation of the reference data to verify. The value of n is application-dependent (n ≤ 127). The application can accept reference data of variable length.

### Response Data Field

VERIFY does not return any data field.

### Status Word

|  |  |
| --- | --- |
| **SW1 SW2** | **Meaning** |
| 63 Cx | Reference data comparison failed and *x* tries remain |
| 63 CF | Reference data retry counter non-deterministic |
| 69 83 | Authentication method blocked |
| 6A 80 | Incorrect parameter in command data field |
| 6A 86 | Bad parameter P1-P2 |
| 6A 88 | Reference data not found (such as no application password) |
| 90 00 | Successful execution |

### Conditional Usage

An attempt to verify the reference data by providing a reference code of the wrong length or of the wrong format is viewed as failed verification, and the associated retry counter is decremented.

The maximum consecutive number of unsuccessful attempts to verify the reference data is defined by the application and is outside the scope of this specification.

There is one retry counter per reference data.

If the card command succeeds, the security status of the reference data is set to TRUE and the retry counter that is associated with the reference data is reset to its initial value.

If the card command fails, the security status of the reference data is set to FALSE and the reset counter that is associated with the reference data is decremented by one.

If the verification failed on the last attempt, the card returns 63C0. If the number of remaining tries is already equal to 0 before the APDU is sent, the card returns the status word 6983.

When chaining is used with the verify command, a new APDU must start with every template to verify. The command data field is normally present for conveying verification data. The absence of command data field is used to check whether the reference data has already been successfully verified. The commands returns either 90 00 (reference data already verified) or 63 CX (reference data not verified and *X* tries remaining) as required by ISO/IEC 7816-4:2005 section 7.5.6.

# APDU Mapping

This section describes how the cryptographic and authentication mechanisms that were described in “Cryptopgraphic Algorithms” and “Authentication and Session Key Agreement Protocols” are mapped to the ISO/IEC 7816 APDU that was defined in section 13.

The CLASS byte in the following APDU must be modified if the communication is done within a secure channel or a logical channel other than the default one.

## Authentication Mechanisms

The mapping in the following section uses GENERAL AUTHENTICATE or INTERNAL AUTHENTICATE commands in compliance with commonly accepted best cryptographic practices.

The CLASS byte in the following APDU must be modified if the communication is done within a secure channel or a logical channel other than the default one. Chained APDUs (CLA = 0x10) shall be used when necessary.

In the following tables, *LL* is used to represent the length of the following data and may have different values even within an APDU.

### Mutual Authentication with Symmetric Algorithm

For details on the cryptograms, see “Mutual Authentication with Symmetric Algorithm.”

Table : Mutual Authentication with Session Key Agreement Using Symmetric Algorithm

|  |  |  |
| --- | --- | --- |
| **Command** | **Response data field** | **Comments** |
| 00 87 00 00 LL  7C *LL* 81 *LL* RR RR RR |  | GENERAL AUTHENTICATE to send off-card application challenge (random) |
|  | 7C *LL*.81 LL rr…rr | Card challenge (random) |
| 00 87 00 00 *LL*  7C *LL* 82 *LL* CC…CC |  | GENERAL AUTHENTICATE to send off-card application response; cryptogram length depends on AT key type |
|  | 7C *LL*.82 *LL* cc…cc | Card response (cryptogram length depends on AT key type) |

### External Authentication with Symmetric Algorithm

For details on the cryptograms, see “External Authentication with Symmetric Algorithm.”

Table : External Authentication Using Symmetric Algorithm

|  |  |  |
| --- | --- | --- |
| **Command** | **Response data field** | **Comments** |
| 00 87 00 00 04  7C *02* 81 00 |  | GENERAL AUTHENTICATE to send empty off-card application challenge |
|  | 7C *LL*.81 LL rr…rr | Card challenge (random) |
| 00 87 00 00 *LL*  7C *LL* 82 *LL* CC…CC |  | GENERAL AUTHENTICATE to send off-card application response; cryptogram length depends on AT key type |

### Key Establishment with Internal Authentication Using ECC

For details on the cryptograms, see “Key Establishment with Internal Authentication Using ECC.”

Table : Internal Authentication with Session Key Agreement Using ECC Algorithm

|  |  |  |
| --- | --- | --- |
| **Command** | **Response** | **Comments** |
| 00.CB.3F.FF.*05* 5C.*03*.TT.TT.TT |  | GET DATA to retrieve card-certificate TT TT TT maps to the tag of the DO that contains the certificate of the card’s static public key. |
|  | TT TT TT.*LL*.xx.xx….xx | xx xx …xx maps to the card certificate. |
| 00.88.00.00.*LL* 7C.*LL*.85.*LL*.QQ…QQ A0.*LL* 80 *LL* xx…xx |  | INTERNAL AUTHENTICATE for card authentication: QQ…QQ maps to QH and xx xx xx xx xx xx xx xx maps to IDH for KDF. |
|  | 7C.*LL*. 82.*LL*.MM..MM A0 *LL* 81.*LL*.nn…nn | MM…MM maps to the MacTag nn…nn maps to the nonce Nc. |

# Transport Protocol Management

## Communication Interface and Supported Protocols

The GIDS defines a command set at the APDU level. It can map directly into TPDU for cards that communicate in T = 1 in contact mode and in T = CL in contactless mode.

Other communication protocols are possible as long as the interface at the APDU level complies with this specification.

## Extended Length

The extended length feature is optional. The historical bytes of the ATR, which are stored in the EF.ATR (see “EF.ATR”), indicates whether the GIDS platform can handle extended length.

## Sending More than 255 bytes to the ICC

GIDS handles the following incoming commands that may contain data whose length is greater than FF = 255 bytes:

* PUT DATA
* GENERAL AUTHENTICATE
* PSO

This section clarifies the behavior of the application in such cases. The following two options are available:

* Use of extended length if supported by the card.
* Use of CC if extended length is not supported or if the length of the data field still exceeds the maximum capacity listed in section “Use of Extended Length for Incoming Data” in this section.

### Use of Extended Length for Incoming Data

See ISO/IEC 7816-3.

### Use of Command Chaining for Incoming Data

See “Command Chaining.”

## Command Returning More Than 256 Bytes

The following commands may return data greater than 256 bytes (with or without secure messaging):

* GET DATA
* GENERAL AUTHENTICATE
* PSO

When these commands are correctly processed:

* The outgoing data are prepared (signature computed).
* Depending on the security level required, the data is wrapped or not wrapped though the secure messaging layer, whichever is required.

Then the data is made available to the off-card application. This amount of data must be retrieved from the ICC.

The following three scenarios may be considered:

* The ICC supports extended length for data retrieval (outgoing data), and the data to return fits within the maximum length for the response data that is set in the EF.ATR (see “EF.ATR”).
* The ICC does not support extended length for data retrieval (outgoing data).
* The data to return does not fit within the maximum length for the response data (which was set in the EF.ATR).

### Case 1

If the ICC supports extended length for data retrieval (outgoing data) and the data to return fits within the maximum length for the response data, the data shall be returned by using this feature.

To do so, the incoming command shall be sent by using the extended length (see ISO/IEC 7816-3), that is:

* the Lc field shall be encoded over 3 bytes : 00 XX YY (for Case 4 command).
* the Le field shall be encoded over 2 bytes set to zero: 00 00.

The ICC returns all available data to the off-card application by using extended APDU (see ISO/IEC 7816-3).

### Case 2

If the ICC does not support extended length, the outgoing data shall be retrieved by using short length. The data retrieval is performed by using the GET RESPONSE command (see “GET RESPONSE”). The off-card application shall recover the data that way.

|  |  |  |
| --- | --- | --- |
| **Incoming data** | **Outgoing data** | **Status word returned** |
| Incoming command CLA INS P1 P2 Lc Data Le | Return 256 bytes | SW = 61  LDataLData indicates the length of data to retrieve.  LData = 00 means 256 bytes are available. |
| Incoming command GET RESPONSE LData (see “GET RESPONSE”) | Return LData bytes | If there are still remaining data, SW = 61 LData.  LData indicates the length of data to retrieve.  LData = 00 means 256 bytes are available.  If all data was retrieved, SW = 9000. |
| Carry on the sequence of GET RESPONSE command until the SW 9000 is returned | | |

After the incoming command is received, the outgoing data can be retrieved only by using the GET RESPONSE command (see ISO/IEC 7816-4) as described earlier.

The GET RESPONSE command does not modify the internal status of the GIDS-based application (authentication status). This command is just handled by the transport layer.

After the sequence of data retrieval by using the GET RESPONSE command has begun, it shall be continued until the entire completion of the sequence. If a command different from the GET RESPONSE command is sent during the data retrieval sequence, all remaining data to recover is erased and is not retrievable.

### Case 3

The data to return does not fit within the maximum length for the response data when extended lengths are used. In this case, the data shall be returned by using extended length combined with a GET RESPONSE command. To do so, the incoming command shall be sent by using the extended length (see ISO/IEC 7816-3), that is:

* The Lc field shall be encoded over 3 bytes : 00 XX YY (for Case 4 command).
* The Le field shall be encoded over 2 bytes set to zero : 00 00.

However, all data to return might not fit within the maximum length for the response data field. The data retrieval is performed by using the GET RESPONSE command. The off-card application shall recover the data that way.

Let Lmax be the maximum length for the response data.

|  |  |  |
| --- | --- | --- |
| **Incoming data** | **Outgoing data** | **Status word returned** |
| Incoming command  CLA INS P1-P2 Lc Data Le | Return Lmax bytes | SW = 6100  It means at most Lmax bytes are still available. |
| Incoming command  GET RESPONSE 00 Lmax  Or  GET RESPONSE 00 00 00  (see “GET RESPONSE”) | Return at most Lmax bytes | If there is still remaining data, SW = 6100.  It means at most Lmax bytes are still available.  If all data was retrieved, SW = 9000. |
| Carry on the sequence of GET RESPONSE command until the SW 9000 is returned | | |

After the incoming command is received, the outgoing data can be retrieved only by using the GET RESPONSE command (see “GET RESPONSE”) as described earlier.

The GET RESPONSE command does not modify the internal status of the card application (authentication status). This command is just handled by the transport layer.

After the sequence of data retrieval by using the GET RESPONSE command begins, it shall be continued until the entire completion of the sequence. If a command different from the GET RESPONSE command is sent during the data retrieval sequence, all remaining data to recover is erased and is not retrievable.

### GET RESPONSE

|  |  |
| --- | --- |
| **CLA** | 00 |
| **INS** | C0 |
| **P1** | 00 |
| **P2** | 00 |
| **Le Field** | LL |
|  |  |
| **Data Field** | Data to retrieve from the ICC |
| **SW1-SW2** | 6985 - There was no data to retrieve.  61xx - There are still *xx* bytes to retrieve from the ICC.  9000 - Correct processing–the data retrieval sequence is completed. |

Depending on the use case, the Le field (LL) may be a short length (encoded over 1 byte) or an extended length (encoded over 3 bytes 00 xx yy).

# Appendix A: Technical Limitations

## Technical Minima

This section describes minimums for interoperability:

* Because of the parsing that is involved to process the GET DATA command, card performance decreases as the number of DOs rise. For acceptable performances, we recommend to keep the number of DOs below 100.
* The data size of a primitive tag is limited to a maximum of 16 KB.
* The maximum number of DOs that a file can store has been set to 100.
* GIDS supports logical channels. However, the first version does not support concurrent selections through different logical channels.

## Compliance with ISO 7816

This specification was written to ensure close compatibility with ISO/IEC 7816 standards. However, some of the GIDS functionalities could not be achieved by using exclusively the ISO standards. This section lists the discrepancies between GIDS and ISO/IEC 7816 standards. These discrepancies will be communicated to WG4 in charge of maintaining the ISO/IEC 7816 standards to be considered for future revisions of their standards.

Discrepancies that were introduced by functionality that was used exclusively for backward compatibility with NIST SP800-73 are not listed in this section because such noncompliance aspects disappear when the card is personalized in strict GIDS mode.

### Discrepancies in APDU Behavior

#### SELECT APDU

When the P1-P2 values are 00-00, 2F-00, 2F-01, and 3F-FF, the implicit selection mechanism is not activated. Therefore, for these specific values, the current files and security status are unchanged after the execution of a GET DATA or PUT DATA command. This allows an application to retrieve the contents of EF.ATR and EF.DIR that are in the MF without losing the currently selected application or any current security status. See “P1-P2 Parameters in the GET DATA and PUT DATA Commands.”

### Tag Created by GIDS

The following tag has been created by GIDS: 7F68 in the FMD template: Succession of OIDs under which the application is known. See “FMD Template.”

# Appendix B: Definitions and Acronyms

## Definitions

constructed DO

A DO whose value is constructed. The BER-encoding of the value is a concatenation of BER-TLV encoding component DO.

data object (DO)

A byte string that represents typed data. The byte string contains a BER-TLV where the tag encodes the DO type. The type defines the encoding of the subsequent value field.

primitive DO

A DO whose value is primitive. As such, the BER-encoding of the value is entirely specified by the type and inner structure, if any, is unknown to the command set.

SC DO

A DO that is stored in the command set under a single reference. A SC DO is created, retrieved, or updated as a single block. If the stored DO is constructed, its components cannot be updated individually.

template DO

A constructed DO that has individually accessible component DO.

top-level DO

A DO that has been created without specifying a constructed DO (template) to which it belongs.

## Acronyms

PIV

Personal Identity and Verification Card

PIV End Point

Personal Identity and Verification card that has a command set that complies with the end point command set specified in SP 800-73.

TWIC

Transportation Worker Identification Credential

RT

Registered Traveler

FUPAC

Florida Uniform Port Access Credential

FRAC

First Responder Authentication Credential

## Notation

The concatenation operation on bit strings is denoted ||; for example, 001 || 10111 = 00110111.

# Appendix C: References

ISO/IEC 7816 Identification Cards – Integrated Circuit Cards

Part 3: Cards with Contacts: Electrical Interface and Transmission Protocols (2006)

Part 4: Organization, Security and Commands for Interchange (2005)

Part 5: Registration of Application Providers (2004)

Part 6: Interindustry Data Elements for Interchange (2004 with corrigendum 2006)

Part 8: Commands for Security Operations (2004)

Part 9: Commands for Card Management (2004)

Part 11: Personal Verification through Biometric Methods (2004)

Part 13: Commands for Application Management in Multi-application Environment (2007)

ISO/IEC 24727 Identification Cards – Integrated Circuit Cards Programming Interfaces [9] Part 1: Architecture (2007)

Part 2: Generic Card Interface (2008)

Part 3: Application Interface (2008)

ISO/IEC 24787 Identification Cards – On-Card Matching

CD 24727 dated 2008-05-20

ISO/IEC 8825 Information Technology – ASN1 Encoding Rules

Part 1: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) Data element specification (2002)

ISO/IEC 19785 Information Technology – Common Biometric Exchange Formats Framework

Part 1: Data element specification (2006)

ISO/IEC 19794 Information Technology – Biometric Data Interchange formats

Part 2: Finger minutia data (2005)

ISO/IEC 15946 Information Technology – Security Techniques – Cryptographic techniques based on elliptic curves

Part 1: General (2002)

Part 2: Digital Signature (2002)

Part 3: Key Establishment (2002)

• ISO/IEC 9797-1 Information Technology – Security Techniques – Message Authentication Codes (MACs)

Part 1: Mechanisms using a block cipher (1999)

• NIST References

FIPS 180-2 Secure Hash Standard (2002)

FIPS 186-2 Digital Signature Standard (2000)

FIPS 201-1 Change Notice, Personal Identity Verification (PIV) of Federal Employees and Contractors (2006)

SP 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, (2005)

SP 800-56A revision 1, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, (2007)

SP 800-73-2 - Interfaces for Personal Identity Verification, (2008)

SP 800-76-1 - Biometric Data Specification for Personal Identity Verification (2007)

SP 800-78-1 - Cryptographic Standards and Key Sizes for Personal Identity Verification, (2007)

NISTIR 7284 - Personal Identity Verification Card Management Report, (2006)

Test Approach v0.3 for Secure Biometric Match On Card feasibility study. (2007)

MINEX II -Performance of Fingerprint Match-on-Card Algorithms Phase II Report – NIST Interagency Report 7477, (2008). See <http://fingerprint.nist.gov/minexII>

• RSA References

PKCS #1 v2.1: RSA cryptography Standard. (2002)

• ANSI References

ANS X9.62-2005 Public Key service for the Financial Service Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)

• Other References

Karger, P.A. Privacy and Security Threat Analysis of the Federal Employee Personal Identity Verification (PIV) Program. in Proceedings of the 2nd Symposium on Usable Privacy and Security. 12-14 July 2006, Pittsburgh, PA: ACM Press. p. 114-121. <http://cups.cs.cmu.edu/soups/2006/proceedings/p114_karger.pdf>

Kc, G.S. and P.A. Karger, Preventing Attacks on Machine Readable Travel Documents (MRTDs), RC 23909 (W0603-079), 10 March 2006, IBM T. J. Watson Research Center: Yorktown Heights, NY. <http://www.research.ibm.com/resources/paper_search.html>

Scherzer, H., R. Canetti, P.A. Karger, H. Krawczyk, T. Rabin, and D.C. Toll. Authenticating Mandatory Access Controls and Preserving Privacy for a High-Assurance Smart Card. in8th European Symposium on Research in Computer Security (ESORICS 2003). 13-15 October 2003, Gjøvik, Norway: Lecture Notes in Computer Science Vol. 2808. Springer Verlag. p. 181-200.

“The order of encryption and authentication for protecting communications (or: How secure is SSL?)” by Hugo Krawczyk. In Advances in Cryptology — CRYPTO 2001, volume 2139 of Lecture Notes in Computer Science, pages 310-331. Springer-Verlag, 2001. <http://www.iacr.org/archive/crypto2001/21390309.pdf>

1. 9F17 or 97 are valid values for this tag because 9F17 is not a tag value that can be retrieved from decoding and then encoding the resulting value. 9F17 is the multi-byte encoding for a single byte tag (97). When this tag is decoded, the resulting value is 97 and encoding on this value results in a value of 97. [↑](#footnote-ref-1)