



**Global
Platform®**

The standard for
secure digital services
and devices

GlobalPlatform Technology

Cryptographic Service Provider Card Specification v2.3 – Amendment N

Version 0.0.0.40

Public Review

May 2025

Document Reference: GPC_SPE_230

Copyright © 2023-2025 GlobalPlatform, Inc. All Rights Reserved.

Recipients of this document are invited to submit, with their comments, notification of any relevant patents or other intellectual property rights of which they may be aware which might be necessarily infringed by the implementation of the specification or other work product set forth in this document, and to provide supporting documentation. This document (and the information herein) is subject to updates, revisions, and extensions by GlobalPlatform, and may be disseminated without restriction. Use of the information herein (whether or not obtained directly from GlobalPlatform) is subject to the terms of the corresponding GlobalPlatform license agreement on the GlobalPlatform website (the "License"). Any use (including but not limited to sublicensing) inconsistent with the License is strictly prohibited.

THIS SPECIFICATION OR OTHER WORK PRODUCT IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE COMPANY, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER DIRECTLY OR INDIRECTLY ARISING FROM THE IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT.

Contents

1	Introduction	23
1.1	Audience	23
1.2	Document Structure	24
1.3	IPR Disclaimer	24
1.4	References	24
1.5	Terminology and Definitions	31
1.5.1	Components	31
1.5.2	Data Structures	33
1.5.3	Roles	34
1.5.4	Interfaces	34
1.6	Abbreviations	35
1.7	Revision History	38
2	Use Cases and Requirements	39
2.1	Use Cases	39
2.2	Requirements	40
2.3	Modularity	43
2.4	Exclusions and Limitations	44
2.4.1	Post-Quantum Cryptography (PQC) Considerations	44
2.4.2	Legacy Cryptographic Algorithms	44
2.4.3	Interactions with Certificate Authorities	44
3	Features	45
3.1	Resource Management	45
3.1.1	Create Resources	45
3.1.2	Destroy Resources	45
3.1.3	Load Resource for Personalization	46
3.1.4	Configure Resources	46
3.1.5	Resource Identifiers	46
3.1.6	Streaming Resources for Offloading	47
3.1.7	Clear Resources	47
3.2	Access Control	48
3.2.1	Client Application Registration	48
3.2.2	Enforce Client Authentication	49
3.2.3	Resource Ownership	49
3.2.4	Access Control Rules	49
3.2.5	Usage Concept	50
3.2.6	Policies	50
3.3	Key Management	52
3.3.1	Key Generation	52
3.3.2	Key Derivation	53
3.3.3	Key Agreement	54
3.3.3.1	ECKA-DH	54
3.3.3.2	ECKA-EG	55
3.3.4	Import and Export Public Keys	55
3.3.5	Transient Keys	56
3.4	Certificate Management	57
3.4.1	Import Certificates	57
3.4.2	Export Certificates	58
3.4.3	Certificate Trust Chain Verification	58

3.4.4	Certificate Rollover	58
3.4.5	Extract Certificate Data	58
3.4.6	Limitations of Certificate Handling	58
3.4.6.1	Certificate Signing	58
3.4.6.2	Certificate Overlap	59
3.4.6.3	Certificate Revocation	59
3.5	Password Management	60
3.5.1	Password Verification.....	60
3.5.2	Retry Counter with Maximum Try Limit.....	61
3.5.3	Unblocking Passwords.....	61
3.5.4	Passwords in Transport	62
3.5.5	Updating Passwords	62
3.6	Cipher and Signatures	63
3.6.1	Padding Schemes	63
3.6.2	Hash and Message Digest.....	63
3.6.3	Cipher.....	64
3.6.4	Cipher Block Modes	64
3.6.5	Signature Suites.....	65
3.6.6	Proof of Association	65
3.6.7	Signatures with Counters & Timestamps.....	67
3.6.8	Encryption Transformations	67
3.7	Secure Messaging	68
3.7.1	Secure Communication Flow	68
3.7.2	Built-In Secure Channel Protocols	69
3.7.2.1	PACE	70
3.7.2.2	EAC for eID.....	70
3.7.2.3	EAC for MRTD	71
3.7.2.4	PACE-CAM.....	72
3.7.2.5	SCP03	73
3.7.2.5.1	Key-ENC	73
3.7.2.5.2	Key-MAC	73
3.7.2.6	SCP04	73
3.7.3	Building Blocks for Custom Secure Channel Protocols	73
3.7.4	Confidential Data Transfer	74
3.7.4.1	Motivation for Confidential Data Transfer	75
3.7.4.2	Receiving Confidential Data	75
3.7.4.3	Sending Confidential Data.....	76
3.7.4.4	Combination Transform Service with Confidential Data Transfer	77
3.8	Attestations	78
3.8.1	Platform Attestation.....	78
3.8.2	Config Attestation.....	78
3.8.3	Data Attestation.....	79
3.8.4	Proof of Possession Key Attestation.....	80
3.8.5	Generate Key Pair Attestation	81
3.9	Counter and Limits	82
3.9.1	Overview Counters.....	82
3.9.1.1	Manual Counter	83
3.9.1.2	Usage Counter.....	83
3.9.1.3	Usage Counter per Block	84
3.9.1.4	Success Counter	84
3.9.1.5	Failure Counter	84
3.9.1.6	Transport Counter	84

3.9.1.7	Timeout Counter	85
3.9.2	Maximum Counter Increment Capacities	85
3.10	Timer and Time Management	86
3.10.1	Overview Timers	86
3.10.1.1	Manual Timer	86
3.10.1.2	Validity Period	87
3.10.1.3	Validity Date for Keys and Passwords	87
3.10.1.4	Validity Date of Certificates	87
3.10.1.5	Authentication Timeout	87
3.10.1.6	Security Timeout	88
3.10.2	Time Synchronization	88
3.10.2.1	TA Certificates as Time Source	89
3.10.2.2	External Timekeeper	89
3.10.2.3	Time Verification Key	90
3.10.3	Time Not Available	90
3.10.3.1	Time Not Synchronized	91
3.10.3.2	No Time Support	91
3.11	Secure Auditing	92
3.11.1	Creating Log Messages	92
3.11.2	Processing Log Messages	93
3.11.3	Audit Event Queue	93
3.11.4	Selecting Audit Events	94
3.11.5	Limitations of Audit	94
3.11.5.1	Impact on Memory from Logging	94
3.11.5.2	Audit is deactivated by Default	95
3.11.5.3	Limited Event Buffering Capacity	95
4	Architecture and Instantiation	96
4.1	Architecture	96
4.2	Role Model	97
4.3	Deployment	99
4.4	Instantiation	100
4.4.1	AIDs of the CSP	100
4.4.2	Global Service ID for the CSP	100
4.4.3	Privileges Required by the CSP	101
4.4.4	CSP Install Parameters	101
4.4.5	CSP INSTALL Command	101
4.5	Lifecycle	102
5	Core Modules.....	103
5.1	System Module	103
5.1.1	System Definitions	103
5.1.1.1	Operation Modes	103
5.1.1.2	Modules	103
5.1.1.3	System Attestations	105
5.1.1.4	Core System Events	105
5.1.1.5	Error Modes	106
5.1.1.6	Error Types	107
5.1.1.7	Core Error Codes	108
5.1.2	System Configuration	109
5.1.3	System Operations	110
5.1.3.1	Access Restrictions on System Operations	111
5.1.3.2	API Level Compliance	112

5.1.3.3	Sensitive Results Checks	112
5.1.3.4	Sensitive Arrays	113
5.1.4	System Lifecycle	114
5.1.4.1	Internal System Parameters	114
5.1.5	System Structures	115
5.1.5.1	CSPCoreSupport	115
5.1.5.2	CSPSettings	115
5.1.5.3	CSPProtocolVersion	116
5.1.5.4	CSPConfiguration	116
5.1.5.5	CSPMode	117
5.1.5.6	CSPConfigVersion	118
5.1.5.7	CSPConfigName	118
5.1.5.8	CSPErrorMode	118
5.1.5.9	CSPPlatform	119
5.1.5.10	CSPAPIVersion	119
5.1.5.11	CSPELFVersion	119
5.1.5.12	CSPAID	120
5.1.5.13	CSPBoolean	120
5.1.5.14	CSPChallenge	120
5.1.5.15	CSPSignature	121
5.2	Resource Module	122
5.2.1	Resource Definitions	122
5.2.1.1	Resource Types	122
5.2.1.2	Usage Types	123
5.2.1.3	Resource States	126
5.2.1.4	Resource Identifiers	128
5.2.1.5	Core Resource Events	128
5.2.2	Resource Configuration	129
5.2.3	Resource Operations	130
5.2.3.1	Access Restrictions on Resource Operations	131
5.2.3.2	Sensitive Results computed by Resource Operations	132
5.2.4	Resource Lifecycle	132
5.2.4.1	Internal Resource Parameters	132
5.2.5	Resource Structures	132
5.2.5.1	CSPResourceSupport	133
5.2.5.2	CSPResource	133
5.2.5.3	CSPResourceId	134
5.2.5.4	CSPResourceState	134
5.2.5.5	CSPResourceValue	135
5.2.5.6	CSPResourceType	135
5.2.5.7	CSPUsageType	136
5.2.5.8	CSPAlgorithms	137
5.3	Access Module	139
5.3.1	Access Definitions	139
5.3.1.1	Client Authentication	139
5.3.1.2	Access Rights	139
5.3.1.3	ACR Bit Positions	141
5.3.2	Access Configuration	142
5.3.3	Access Operations	143
5.3.4	Access Lifecycle	143
5.3.5	Access Structures	143
5.3.5.1	CSPClient	143

5.3.5.2	CSPClientReference	144
5.3.5.3	CSPClientApplication	144
5.3.5.4	CSPAccessControl	145
5.3.5.5	CSPAccessControlRules	145
6	Optional Modules	148
6.1	Cipher Module	148
6.1.1	Cipher Definitions	148
6.1.1.1	Cipher Modes	148
6.1.1.2	Padding Algorithms	148
6.1.1.3	Cipher Algorithms	149
6.1.1.4	Cipher, Padding, and Key Size Combinations	151
6.1.1.5	Cipher Initialization Data	151
6.1.1.6	Cipher Events	152
6.1.1.7	Cipher Error Codes	153
6.1.2	Cipher Configuration	154
6.1.3	Cipher Operations	154
6.1.3.1	Access Restrictions on Cipher Operations	155
6.1.3.2	Sensitive Results Computed by Cipher Operations	156
6.1.3.3	Sensitive Arrays Required for Cipher Operations	156
6.1.4	Cipher Lifecycle	157
6.1.5	Cipher Structures	157
6.1.5.1	CSPCipherSupport	157
6.1.5.2	CSPCipherAlgorithms	158
6.1.5.3	CSPPaddingAlgorithm	158
6.1.5.4	CSPCipherAlgorithm	159
6.2	Signature Module	161
6.2.1	Signature Definitions	161
6.2.1.1	Signature Modes	161
6.2.1.2	Message Digest Algorithms	161
6.2.1.3	Signature Algorithms	162
6.2.1.4	Signature, Padding, Hash, Key Size and Curve Combinations	163
6.2.1.5	Signature Events	165
6.2.1.6	Signature Error Codes	165
6.2.2	Signature Configuration	166
6.2.3	Signature Operations	167
6.2.3.1	Access Restrictions on Signature Operations	168
6.2.3.2	Sensitive Results Computed by Signature Operations	168
6.2.3.3	Sensitive Arrays Required for Signature Operations	169
6.2.4	Signature Lifecycle	169
6.2.5	Signature Structures	170
6.2.5.1	CSPSignatureSupport	170
6.2.5.2	CSPMessageDigestAlgorithm	170
6.2.5.3	CSPSignatureAlgorithm	171
6.2.5.4	CSPSignatureAlgorithms	172
6.3	Transform Module	174
6.3.1	Transform Definitions	174
6.3.1.1	Transform Error Codes	174
6.3.2	Transform Configuration	174
6.3.3	Transform Operations	174
6.3.3.1	Access Restrictions on Transform Operations	175
6.3.3.2	Sensitive Results computed by Transform Operations	176
6.3.3.3	Sensitive Arrays required for Transform Operations	176

6.3.4	Transform Lifecycle	176
6.4	Secure Channel Module	178
6.4.1	Secure Channel Definitions	178
6.4.1.1	Protocol Types	178
6.4.1.2	Security Functions	181
6.4.1.3	Protocol, Resource, Size and Curve Combinations	184
6.4.1.4	Security States	185
6.4.1.4.1	PACE States	186
6.4.1.4.2	EAC eID States	186
6.4.1.4.3	EAC Passport States	186
6.4.1.4.4	PACE CAM States	187
6.4.1.4.5	SCP03 & SCP04 States	187
6.4.1.5	Secure Channel Events	187
6.4.1.6	Secure Channel Error Codes	188
6.4.2	Secure Channel Configuration	188
6.4.3	Secure Channel Operations	189
6.4.3.1	Access Restrictions on Secure Channel Operations	191
6.4.3.2	Sensitive Results Computed by Secure Channel Operations	192
6.4.3.3	Sensitive Arrays Required for Secure Channel Operations	192
6.4.4	Secure Channel Lifecycle	192
6.4.4.1	Internal Secure Channel Parameters	193
6.4.5	Secure Channel Structures	193
6.4.5.1	CSPSecureChannelSupport	194
6.4.5.2	CSPSecureChannelSettings	194
6.4.5.3	CSPProtocolType	194
6.4.5.4	CSPSecureChannelAlgorithms	195
6.4.5.5	CSPSecurityFunction	196
6.5	Confidential Data Transfer Module	198
6.5.1	Confidential Data Transfer Definitions	198
6.5.1.1	Confidential Data Transfer Error Codes	198
6.5.2	Confidential Data Transfer Configuration	199
6.5.3	Confidential Data Transfer Operations	199
6.5.3.1	Access Restrictions on Confidential Data Transfer Operations	200
6.5.3.2	Sensitive Results computed by Confidential Data Transfer Operations	201
6.5.3.3	No Sensitive Arrays required for Confidential Data Transfer Operations	201
6.5.4	Confidential Data Transfer Lifecycle	202
6.6	Attestation Module	203
6.6.1	Attestation Definitions	203
6.6.1.1	Resource Attestation Types	203
6.6.1.2	Attestation Type, Component and Algorithm Combinations	204
6.6.1.3	Attestation Error Codes	204
6.6.2	Attestation Configuration	205
6.6.3	Attestation Operations	205
6.6.3.1	Access Restrictions on Attestation Operations	207
6.6.3.2	Sensitive Results Computed by Attestation Operations	207
6.6.3.3	Sensitive Arrays Required for Attestation Operations	207
6.6.4	Attestation Lifecycle	208
6.6.5	Attestations Structures	208
6.6.5.1	CSPAttestationSupport	208
6.6.5.2	CSPSystemAttestationType	209
6.6.5.3	CSPResourceAttestationType	209
6.6.5.4	CSPAttestationAlgorithms	210

6.6.5.5	CSPPlatformAttestation	210
6.6.5.6	CSPConfigAttestation	212
6.6.5.7	CSPDataAttestation	213
6.6.5.8	CSPKeyPoPAAttestation	214
6.7	Key Module	216
6.7.1	Key Definitions	216
6.7.1.1	Key Types	216
6.7.1.2	Key Sizes	219
6.7.1.3	ECC Curves	220
6.7.1.4	Key Management Modes	221
6.7.1.5	Key Generation Processes	221
6.7.1.6	Key Derivation Algorithms	222
6.7.1.7	Key Derivation, Resource Type, Hash, Size and Curve Combinations	224
6.7.1.8	Key Derivation Additional Input Data	225
6.7.1.9	Key Agreement Schemes	226
6.7.1.10	Key Agreement, Key Type, Curve and Size, Combinations	226
6.7.1.11	Key Events	227
6.7.1.12	Key Error Codes	228
6.7.2	Key Configuration	228
6.7.3	Key Operations	229
6.7.3.1	Access Restrictions on Key Operations	232
6.7.3.2	Sensitive Results Computed by Key Operations	233
6.7.3.3	Sensitive Arrays Required for Key Operations	234
6.7.4	Key Lifecycle	234
6.7.5	Key Structures	235
6.7.5.1	CSPKeySupport	235
6.7.5.2	CSPKey	236
6.7.5.3	CSPKeyType	236
6.7.5.4	CSPKeySize	237
6.7.5.5	CSPCurve	239
6.7.5.6	CSPKeySizeOrCurve	240
6.7.5.7	CSPKeyDerivationAlgorithms	241
6.7.5.8	CSPKeyDerivationAlgorithm	241
6.7.5.9	CSPKeyAgreementScheme	242
6.8	Certificate Module	243
6.8.1	Certificate Definitions	243
6.8.1.1	Certificate Types	243
6.8.1.2	Certificate Management Modes	244
6.8.1.3	Certificate Events	244
6.8.1.4	Certificate Error Codes	244
6.8.2	Certificate Configuration	245
6.8.3	Certificate Operations	245
6.8.3.1	Access Restrictions on Certificate Operations	247
6.8.3.2	Sensitive Results Computed by Certificate Operations	248
6.8.3.3	Sensitive Arrays Required for Certificate Operations	248
6.8.4	Certificate Lifecycle	248
6.8.5	Certificate Structures	249
6.8.5.1	CSPCertificateSupport	249
6.8.5.2	CSPCertificate	249
6.8.5.3	CSPCertificateType	250
6.9	Password Module	251
6.9.1	Password Definitions	251

6.9.1.1	Password Types	251
6.9.1.2	Password Events	252
6.9.1.3	Password Error Codes	253
6.9.2	Password Configuration	254
6.9.3	Password Operations	255
6.9.3.1	Access Restrictions on Password Operations	257
6.9.3.2	Sensitive Results computed by Password Operations	258
6.9.3.3	Sensitive Arrays Required for Password Operations	258
6.9.4	Password Lifecycle	258
6.9.4.1	Internal Password Parameters	261
6.9.5	Password Structures	261
6.9.5.1	CSPPasswordSupport	261
6.9.5.2	CSPPassword	262
6.9.5.3	CSPPasswordType	262
6.10	Counter Module	264
6.10.1	Counter Definitions	264
6.10.1.1	Counter Modes	264
6.10.1.2	Counter Types	265
6.10.1.3	Counter Capacities	270
6.10.1.4	Counter, Resource Type and Capacity Combinations	271
6.10.1.5	Counter Events	271
6.10.1.6	Counter Error Codes	272
6.10.2	Counter Configuration	272
6.10.3	Counter Operations	274
6.10.3.1	Access Restrictions on Counter Operations	275
6.10.3.2	Sensitive Results computed by Counter Operations	275
6.10.3.3	Sensitive Arrays Required for Counter Operations	276
6.10.4	Counter Lifecycle	276
6.10.4.1	Internal Counter Parameters	277
6.10.5	Counter Structures	277
6.10.5.1	CSPCounterSupport	277
6.10.5.2	CSPCounterSettings	278
6.10.5.3	CSPCounterMode	278
6.10.5.4	CSPCounterType	279
6.10.5.5	CSPCounters	280
6.10.5.6	CSPCounter	280
6.10.5.7	CSPCounterCapacity	281
6.11	Time Module	282
6.11.1	Time Definitions	282
6.11.1.1	Time Modes	282
6.11.1.2	Time Synchronization Methods	283
6.11.1.3	Timer Types	284
6.11.1.4	Time Formats	288
6.11.1.5	Timeout Types	289
6.11.1.6	Timer Type, Format, Resource Type, and Timeout Type Combinations	290
6.11.1.7	Time Events	290
6.11.1.8	Time Error Codes	291
6.11.2	Time Configuration	291
6.11.3	Time Operations	294
6.11.3.1	Access Restrictions on Time Operations	294
6.11.3.2	Sensitive Results computed by Time Operations	295
6.11.3.3	Sensitive Arrays Required for Time Operations	295

6.11.4	Time Lifecycle	295
6.11.4.1	Internal Time Parameters	296
6.11.5	Time Structures	297
6.11.5.1	CSPTimeSupport	297
6.11.5.2	CSPTimeSettings	297
6.11.5.3	CSPTimeMode	298
6.11.5.4	CSPTimeSynchronization	298
6.11.5.5	CSPTimerType	299
6.11.5.6	CSPTimers	300
6.11.5.7	CSPTimestamp	301
6.11.5.8	CSPDuration	301
6.11.5.9	CSPTimeout	301
6.11.5.10	CSPTimeoutType	302
6.11.5.11	CSPManualTimer	302
6.12	Audit Module	304
6.12.1	Audit Definitions	304
6.12.1.1	Audit Modes	304
6.12.1.2	Audit Events	305
6.12.1.3	Audit Error Codes	306
6.12.2	Audit Configuration	306
6.12.3	Audit Operations	307
6.12.3.1	Access Restrictions on Audit Operations	308
6.12.3.2	Sensitive Results Computed by Audit Operations	308
6.12.3.3	Sensitive Arrays Required for Audit Operations	308
6.12.3.4	Audit Listener	308
6.12.4	Audit Lifecycle	309
6.12.5	Audit Structures	309
6.12.5.1	CSPAuditSupport	309
6.12.5.2	CSPAuditSettings	309
6.12.5.3	CSPAuditMode	310
6.12.5.4	CSPSystemEvent	310
6.12.5.5	CSPResourceEvent	311
6.12.5.6	CSPLogMessage	314
6.12.5.7	CSPEventData	314
6.12.5.8	CSPEventDataUpdateCSP	315
6.12.5.9	CSPEventDataUpdateConfig	316
6.12.5.10	CSPEventDataSetTime	316
6.12.5.11	CSPEventDataGeneralError	317
6.12.5.12	CSPEventDataResource	317
6.12.5.13	CSPEventDataKeyDerivation	318
6.12.5.14	CSPEventDataKeyAgreement	318
6.12.5.15	CSPEventDataPasswordFailure	319
6.13	Offloading Module	320
6.13.1	Offloading Definitions	320
6.13.1.1	Offloading Modes	320
6.13.1.2	Offloading Events	320
6.13.1.3	Offloading Error Codes	321
6.13.2	Offloading Configuration	321
6.13.3	Offloading Operations	321
6.13.3.1	Access Restrictions on Offloading Operations	323
6.13.3.2	Sensitive Results Computed by Offloading Operations	323
6.13.3.3	Sensitive Arrays Required for Offloading Operations	323

6.13.4	Offloading Lifecycle	324
6.14	Field Module	325
6.14.1	Field Definitions	325
6.14.1.1	Fields	325
6.14.1.2	Field Sources	327
6.14.1.3	Field Modes	327
6.14.1.4	Field Error Codes	328
6.14.2	Field Configuration	328
6.14.3	Field Operations	329
6.14.4	Field Lifecycle	329
6.14.5	Field Structures	329
6.14.5.1	CSPFieldSupport	330
6.14.5.2	CSPFieldSettings	330
6.14.5.3	CSPFieldMode	330
6.14.5.4	CSPField	331
6.14.5.5	CSPFieldType	331
6.14.5.6	CSPFieldSource	333
6.14.5.7	CSPFieldValue	333
6.15	Policy Module	335
6.15.1	Policy Definitions	335
6.15.1.1	Policy Modes	335
6.15.1.2	Policy Types	335
6.15.1.3	Policy and Resource Combinations	338
6.15.2	Policy Configuration	339
6.15.3	Policy Operations	339
6.15.4	Policy Lifecycle	340
6.15.5	Policy Structures	340
6.15.5.1	CSPPolicySupport	340
6.15.5.2	CSPPolicySettings	340
6.15.5.3	CSPPolicyMode	340
6.15.5.4	CSPPolicy	341
6.15.5.5	CSPPolicyType	341
6.16	Random Data Module	343
7	CSP Protocol	344
7.1	CSPAdminCommand	344
7.1.1	CSPEnforce	346
7.1.2	CSPRegisterClient	348
7.1.3	CSPUnregisterClient	349
7.1.4	CSPCreateResource	349
7.1.5	CSPDestroyResource	350
7.1.6	CSPConfigureResource	350
7.1.7	CSPSetup	351
7.1.8	CSPActivate	352
7.1.9	CSPDeactivate	353
7.1.10	CSPGetConfiguration	353
7.1.11	CSPSetValue	354
7.2	CSPClientCommand	356
7.2.1	CSPProcessSecurity	357
7.2.2	CSPSign	358
7.2.3	CSPVerifySignature	360
7.2.4	CSPEncrypt	361
7.2.5	CSPDecrypt	362

7.2.6	CSPResourceAttestation	363
7.3	Shared Command Structures	365
7.3.1	CSPClearResource	365
7.3.2	CSPSystemAttestation	365
7.3.3	CSPGenerateKey	367
7.3.4	CSPComputePublicKey	367
7.3.5	CSPDeriveKey	368
7.3.6	CSPSetTime	370
8	CSP API	371
8.1	Prerequisite for Using the CSP API	372
8.2	Retrieve the CSP Instance	373
8.3	Byte Buffer Parameters	375
8.4	Listener Mechanism	376
Annex A	Custom Secure Messaging (Informative)	377
A.1	ECDH-AES / ECKA-DH-AES	377
A.2	ECKA-EG-AES	377
A.3	ECIES	378
A.4	Mobile Driving Licence	379
A.5	FIDO2	380

Tables

Table 1-1: Normative References	24
Table 1-2: Informative References	30
Table 1-3: Terminology and Definitions: Components	31
Table 1-4: Terminology and Definitions: Data Structures	33
Table 1-5: Terminology and Definitions: Roles	34
Table 1-6: Terminology and Definitions: Interfaces	34
Table 1-7: Abbreviations	35
Table 1-8: Revision History	38
Table 2-1: Optional Modules, Algorithms, and Types	43
Table 3-1: Access Control Rules Bit Positions	50
Table 3-2: Algorithm Overview	74
Table 3-3: Attestation Key Parameters	79
Table 3-4: Time Verification Key Parameters	90
Table 3-5: Audit Signing Key Parameters	93
Table 4-1: AIDs of the CSP	100
Table 4-2: Global Service ID of the CSP	100
Table 4-3: GlobalPlatform Privileges Required by the CSP	101
Table 4-4: CSP Install Parameters	101
Table 4-5: CSP INSTALL [for install and make selectable] Command	102
Table 5-1: CSP Operation Modes	103
Table 5-2: CSP Modules	103
Table 5-3: System Attestations Types	105
Table 5-4: Core System Events for Auditing	106
Table 5-5: CSP Error Modes	107
Table 5-6: CSP Error Types	107
Table 5-7: Core Error Codes	108
Table 5-8: CSP Configuration Parameters	109
Table 5-9: CSP Operations	110
Table 5-10: CSP Operations Access Restrictions	111
Table 5-11: CSP API Level Compliance	112
Table 5-12: CSP Assert Sensitive Results Operations	113
Table 5-13: CSP-internal Parameters	114
Table 5-14: Resource Types	122
Table 5-15: Resource Usage Types	124
Table 5-16: Resource States	127
Table 5-17: Resource ID Ranges	128
Table 5-18: Resource Events for Auditing	129
Table 5-19: Resource Configuration Parameters	129
Table 5-20: Resource Operations	131
Table 5-21: Resource Operations Access Restrictions	131
Table 5-22: Resource Operations Requiring Sensitive Results Checks	132
Table 5-23: Resource-internal Parameters	132
Table 5-24: Access Rights	140
Table 5-25: Access Control Rules	141
Table 5-26: Access Control Rules Bit Positions	142
Table 5-27: Access Configuration Options	142
Table 6-1: Cipher Modes	148
Table 6-2: Padding Algorithms	149
Table 6-3: Cipher Algorithms	150

Table 6-4: Cipher, Padding, and Key Size Combinations	151
Table 6-5: Cipher Initialization Data	152
Table 6-6: Cipher Events for Auditing.....	152
Table 6-7: Cipher Error Reason Codes	153
Table 6-8: Cipher Configuration Parameters.....	154
Table 6-9: Cipher Operations.....	155
Table 6-10: Cipher Operations Access Restrictions.....	156
Table 6-11: Cipher Operations Requiring Sensitive Results Checks.....	156
Table 6-12: Cipher Operations Requiring Sensitive Arrays.....	156
Table 6-13: Signature Modes.....	161
Table 6-14: Message Digest Algorithms	162
Table 6-15: Signature Algorithms.....	163
Table 6-16: Signature, Padding, Hash, and Key Size Combinations	164
Table 6-17: Signature Events for Auditing	165
Table 6-18: Signature Error Reason Codes	166
Table 6-19: Signature Configuration Parameters	166
Table 6-20: Signature Operations	167
Table 6-21: Signature Operations Access Restrictions	168
Table 6-22: Signature Operations Requiring Sensitive Results Checks	169
Table 6-23: Signature Operations Requiring Sensitive Arrays	169
Table 6-24: Transform Error Reason Codes	174
Table 6-25: Transform Operations	175
Table 6-26: Transform Operations Access Restrictions	175
Table 6-27: Transform Operations Requiring Sensitive Results Checks	176
Table 6-28: Transform Operations Requiring Sensitive Arrays	176
Table 6-29: Protocol Types	179
Table 6-30: Security Functions	181
Table 6-31: Protocol, Resource, Size, and Curve Combinations	184
Table 6-32: Security States	185
Table 6-33: Security States PACE	186
Table 6-34: Security States EAC eID	186
Table 6-35: Security States EAC v1	186
Table 6-36: Security States PACE CAM	187
Table 6-37: Security States SCP03 & SCP04.....	187
Table 6-38: Secure Channel Events for Auditing.....	188
Table 6-39: Secure Channel Error Reason Codes.....	188
Table 6-40: Secure Channel Configuration Parameters.....	189
Table 6-41: Secure Channel Operations.....	189
Table 6-42: Secure Channel Operations Access Restrictions	191
Table 6-43: Secure Channel Operations Requiring Sensitive Results Checks.....	192
Table 6-44: Secure Channel Operations Requiring Sensitive Arrays.....	192
Table 6-45: Secure Channel-internal Parameters	193
Table 6-46: Confidential Data Transfer Error Reason Codes	198
Table 6-47: Confidential Data Transfer Operations.....	199
Table 6-48: Confidential Data Transfer Access Restrictions.....	201
Table 6-49: Confidential Data Transfer Operations Requiring Sensitive Results Checks.....	201
Table 6-50: Resource Attestation Types.....	203
Table 6-51: Attestation Type, Component, Key, and Algorithm Combinations	204
Table 6-52: Attestation Error Reason Codes.....	205
Table 6-53: Attestation Configuration Parameters.....	205
Table 6-54: Attestation Operations	206
Table 6-55: Attestation Operations Access Restrictions.....	207

Table 6-56: Attestation Operations Requiring Sensitive Results Checks	207
Table 6-57: Attestation Operations Requiring Sensitive Arrays.....	207
Table 6-58: Key Types.....	217
Table 6-59: Key Sizes.....	219
Table 6-60: Curves for ECC Curves	220
Table 6-61: Key Management Modes.....	221
Table 6-62: Key Generation Processes	222
Table 6-63: Key Derivation Algorithms.....	223
Table 6-64: Key Derivation, Resource Type, Hash, Size, and Curve Combinations	224
Table 6-65: Key Derivation Additional Input Data.....	225
Table 6-66: Key Agreement Schemes.....	226
Table 6-67: Key Agreement, Key Type, Curve, and Size Combinations.....	226
Table 6-68: Key Events for Auditing	227
Table 6-69: Key Error Reason Codes.....	228
Table 6-70: Key Configuration Parameters.....	228
Table 6-71: Key Operations	230
Table 6-72: Key Operations Access Restrictions.....	233
Table 6-73: Key Operations Requiring Sensitive Results Checks	233
Table 6-74: Key Operations Requiring Sensitive Arrays	234
Table 6-75: Certificate Types.....	243
Table 6-76: Certificate Management Modes.....	244
Table 6-77: Certificate Events for Auditing	244
Table 6-78: Certificate Error Reason Codes.....	245
Table 6-79: Certificate Configuration Parameters	245
Table 6-80: Certificate Operations	246
Table 6-81: Certificate Operations Access Restrictions.....	247
Table 6-82: Certificate Operations Requiring Sensitive Results Checks	248
Table 6-83: Certificate Operations Requiring Sensitive Arrays	248
Table 6-84: Password Types	251
Table 6-85: Password Events for Auditing.....	252
Table 6-86: Password Error Reason Codes.....	253
Table 6-87: Password Configuration Parameters.....	254
Table 6-88: Password Operations.....	255
Table 6-89: Password Operations Access Restrictions	257
Table 6-90: Password Operations Requiring Sensitive Results Checks.....	258
Table 6-91: Password Operations Requiring Sensitive Arrays.....	258
Table 6-92: Password-internal Parameters	261
Table 6-93: Counter Modes.....	264
Table 6-94: Counter Types.....	266
Table 6-95: Counter Capacities.....	270
Table 6-96: Counter, Resource Type, and Capacity Combinations.....	271
Table 6-97: Counter Events for Auditing	272
Table 6-98: Counter Error Reason Codes	272
Table 6-99: Counter Configuration Parameters	272
Table 6-100: Counter Operations.....	274
Table 6-101: Counter Operations Access Restrictions	275
Table 6-102: Counter Operations Requiring Sensitive Results Checks.....	276
Table 6-103: Counter Operations Requiring Sensitive Arrays.....	276
Table 6-104: Counter-internal Parameters	277
Table 6-105: Time Modes.....	282
Table 6-106: Time Synchronization Methods Bit Positions	283
Table 6-107: Time Synchronization Methods	284

Table 6-108: Timer Types	285
Table 6-109: Time Formats	288
Table 6-110: Timeout Types	289
Table 6-111: Timer Type, Format, Resource Type, and Timeout Type Combinations	290
Table 6-112: Time Events for Auditing	290
Table 6-113: Time Error Reason Codes	291
Table 6-114: Time Configuration Parameters	292
Table 6-115: Time Operations	294
Table 6-116: Time Operations Access Restrictions	294
Table 6-117: Time Operations Requiring Sensitive Results Checks	295
Table 6-118: Time Operations Requiring Sensitive Arrays	295
Table 6-119: Time-internal Parameters	296
Table 6-120: Audit Operation Modes	304
Table 6-121: Audit Error Reason Codes	306
Table 6-122: Audit Configuration Parameters	306
Table 6-123: Audit Operations	307
Table 6-124: Audit Operations Access Restrictions	308
Table 6-125: Audit Operations Requiring Sensitive Results Checks	308
Table 6-126: Audit Operations Requiring Sensitive Arrays	308
Table 6-127: Audit Listener	308
Table 6-128: Key Management Modes	320
Table 6-129: Offloading Events for Auditing	321
Table 6-130: Offloading Error Reason Codes	321
Table 6-131: Offloading Configuration Parameters	321
Table 6-132: Offloading Operations	322
Table 6-133: Offloading Operations Access Restrictions	323
Table 6-134: Offloading Operations Requiring Sensitive Results Checks	323
Table 6-135: Offloading Operations Requiring Sensitive Arrays	324
Table 6-136: Fields	325
Table 6-137: Field Sources	327
Table 6-138: Field Modes	328
Table 6-139: Field Error Reason Codes	328
Table 6-140: CSP Configuration Parameters	329
Table 6-141: Policy Modes	335
Table 6-142: Policy Types	336
Table 6-143: Policy and Resource Combinations	338
Table 6-144: Policy Configuration Options	339
Table 6-145: Random Data Operations	343
Table 7-1: CSP Protocol: CSP-Admin Commands	344
Table 7-2: CSP Protocol: CSP-Client Commands	356

Figures

Figure 2-1: CSP Concept.....	39
Figure 3-1: Proof of Association Key Attestation	66
Figure 3-2: Secure Messaging Flow Chart	69
Figure 3-3: Confidential Data Personalization Flow Chart.....	76
Figure 3-4: Confidential Data Accessing Flow Chart.....	77
Figure 4-1: Architecture of a CSP Platform	96
Figure 4-2: Roles	97
Figure 4-3: Roles and Applications Interacting with the CSP	98
Figure 4-4: CSP Deployment Diagram.....	99
Figure 5-1: CSP Instance-internal Lifecycle	114
Figure 6-1: Cipher Lifecycle	157
Figure 6-2: Signature Lifecycle	170
Figure 6-3: Transform Lifecycle.....	177
Figure 6-4: Secure Channel Lifecycle.....	193
Figure 6-5: Confidential Data Transfer Lifecycle.....	202
Figure 6-6: Attestation Lifecycle	208
Figure 6-7: Key Lifecycle	235
Figure 6-8: Certificate Lifecycle	249
Figure 6-9: Password Lifecycle.....	260
Figure 6-10: Counter Lifecycle.....	277
Figure 6-11: Timer Lifecycle.....	296
Figure 6-12: Offloading Lifecycle.....	324
Figure 8-1: CSP API	371
Figure 8-2: CSP API Listener.....	376

ASN.1 Definitions

ASN 5-1: Access: ASN.1 Definition for CSPCoreSupport.....	115
ASN 5-2: Core: ASN.1 Definition for CSPSettings	115
ASN 5-3: Core: ASN.1 Definition for CSPProtocolVersion.....	116
ASN 5-4: Core: ASN.1 Definition for CSPConfiguration	116
ASN 5-5: Core: ASN.1 Definition for CSPMode.....	117
ASN 5-6: Core: ASN.1 Definition for CSPConfigVersion.....	118
ASN 5-7: Core: ASN.1 Definition for CSPConfigName.....	118
ASN 5-8: Core: ASN.1 Definition for CSPErrorMode.....	118
ASN 5-9: Core: ASN.1 Definition for CSPPlatform.....	119
ASN 5-10: Core: ASN.1 Definition for CSPAPIVersion.....	119
ASN 5-11: Core: ASN.1 Definition for CSPELFVersion.....	120
ASN 5-12: Core: ASN.1 Definition for CSPAID	120
ASN 5-13: Core: ASN.1 Definition for CSPBoolean.....	120
ASN 5-14: Core: ASN.1 Definition for CSPChallenge.....	120
ASN 5-15: Core: ASN.1 Definition for CSPSignature	121
ASN 5-16: Resource: ASN.1 Definition for CSPResourceSupport	133
ASN 5-17: Resource: ASN.1 Definition for CSPResource	133
ASN 5-18: Resource: ASN.1 Definition for CSPResourceId.....	134
ASN 5-19: Resource: ASN.1 Definition for CSPResourceState.....	135
ASN 5-20: Resource: ASN.1 Definition for CSPResourceValue	135
ASN 5-21: Resource: ASN.1 Definition for CSPResourceType	136
ASN 5-22: Resource: ASN.1 Definition for CSPUsageType.....	136
ASN 5-23: Resource: ASN.1 Definition for CSPAlgorithms	137
ASN 5-24: Core: ASN.1 Definition for CSPClient	143
ASN 5-25: Core: ASN.1 Definition for CSPClientReference	144
ASN 5-26: Core: ASN.1 Definition for CSPClientApplication.....	144
ASN 5-27: Access: ASN.1 Definition for CSPAccessControl.....	145
ASN 5-28: Access: ASN.1 Definition for CSPAccessControlRules	146
ASN 6-1: Cipher: ASN.1 Definition for CSPCipherSupport.....	157
ASN 6-2: Cipher: ASN.1 Definition for CSPCipherAlgorithms	158
ASN 6-3: Cipher: ASN.1 Definition for CSPPaddingAlgorithm	158
ASN 6-4: Cipher: ASN.1 Definition for CSPCipherAlgorithm	160
ASN 6-5: Signature: ASN.1 Definition for CSPSignatureSupport	170
ASN 6-6: Signature: ASN.1 Definition for CSPMessageDigestAlgorithm	171
ASN 6-7: Signature: ASN.1 Definition for CSPSignatureAlgorithm	171
ASN 6-8: Signature: ASN.1 Definition for CSPSignatureAlgorithms.....	172
ASN 6-9: SecChannel: ASN.1 Definition for CSPSecureChannelSupport	194
ASN 6-10: SecChannel: ASN.1 Definition for CSPSecureChannelSettings.....	194
ASN 6-11: SecChannel: ASN.1 Definition for CSPProtocolType	194
ASN 6-12: SecChannel: ASN.1 Definition for CSPSecureChannelAlgorithms	195
ASN 6-13: SecChannel: ASN.1 Definition for CSPSecurityFunction.....	196
ASN 6-14: Attestation: ASN.1 Definition for CSPAttestationSupport.....	208
ASN 6-15: Attestations: ASN.1 Definition for CSPSystemAttestationType	209
ASN 6-16: Attestations: ASN.1 Definition for CSPResourceAttestationType.....	209
ASN 6-17: Signature: ASN.1 Definition for CSPAttestationAlgorithms.....	210
ASN 6-18: Attestations: ASN.1 Definition for CSPPlatformAttestation.....	210
ASN 6-19: Attestations: ASN.1 Definition for CSPConfigAttestation	212
ASN 6-20: Attestations: ASN.1 Definition for CSPDataAttestation	213
ASN 6-21: Attestations: ASN.1 Definition for CSPKeyPoPAttestation	214

ASN 6-22: Key: ASN.1 Definition for CSPKeySupport	235
ASN 6-23: Key: ASN.1 Definition for CSPKey	236
ASN 6-24: Key: ASN.1 Definition for CSPKeyType.....	236
ASN 6-25: Key: ASN.1 Definition for CSPKeySize.....	237
ASN 6-26: Key: ASN.1 Definition for CSPCurve.....	239
ASN 6-27: Key: ASN.1 Definition for CSPKeySizeOrCurve.....	240
ASN 6-28: Cipher: ASN.1 Definition for CSPKeyDerivationAlgorithms.....	241
ASN 6-29: Key: ASN.1 Definition for CSPKeyDerivationAlgorithm.....	241
ASN 6-30: Key: ASN.1 Definition for CSPKeyAgreementScheme	242
ASN 6-31: Certificate: ASN.1 Definition for CSPCertificateSupport	249
ASN 6-32: Certificate: ASN.1 Definition for CSPCertificate	250
ASN 6-33: Certificate: ASN.1 Definition for CSPCertificateType	250
ASN 6-34: Password: ASN.1 Definition for CSPPasswordSupport	261
ASN 6-35: Password: ASN.1 Definition for CSPPassword	262
ASN 6-36: Password: ASN.1 Definition for CSPPasswordType	262
ASN 6-37: Counter: ASN.1 Definition for CSPCounterSupport.....	277
ASN 6-38: Counter: ASN.1 Definition for CSPCounterSettings	278
ASN 6-39: Counter: ASN.1 Definition for CSPCounterMode.....	278
ASN 6-40: Counter: ASN.1 Definition for CSPCounterType	279
ASN 6-41: Counter: ASN.1 Definition for CSPCounters.....	280
ASN 6-42: Counter: ASN.1 Definition for CSPCounter.....	281
ASN 6-43: Counter: ASN.1 Definition for CSPCounterCapacity	281
ASN 6-44: Time: ASN.1 Definition for CSPTimeSupport	297
ASN 6-45: Time: ASN.1 Definition for CSPTimeSettings	297
ASN 6-46: Time: ASN.1 Definition for CSPTimeMode	298
ASN 6-47: Time: ASN.1 Definition for CSPTimeSynchronization	299
ASN 6-48: Time: ASN.1 Definition for CSPTimerType	299
ASN 6-49: Time: ASN.1 Definition for CSPTimers	300
ASN 6-50: Time: ASN.1 Definition for CSPTimestamp.....	301
ASN 6-51: Time: ASN.1 Definition for CSPDuration	301
ASN 6-52: Time: ASN.1 Definition for CSPTimeout.....	301
ASN 6-53: Time: ASN.1 Definition for CSPTimeoutType	302
ASN 6-54: Time: ASN.1 Definition for CSPManualTimer	302
ASN 6-55: Audit: ASN.1 Definition for CSPAuditSupport.....	309
ASN 6-56: Audit: ASN.1 Definition for CSPAuditSettings.....	309
ASN 6-57: Audit: ASN.1 Definition for CSPAuditMode	310
ASN 6-58: Audit: ASN.1 Definition for CSPSystemEvent.....	311
ASN 6-59: Audit: ASN.1 Definition for CSPResourceEvent.....	311
ASN 6-60: Audit: ASN.1 Definition for CSPLogMessage	314
ASN 6-61: Audit: ASN.1 Definition for CSPEventData	314
ASN 6-62: Audit: ASN.1 Definition for CSPEventDataUpdateCSP	315
ASN 6-63: Audit: ASN.1 Definition for CSPEventDataUpdateConfig	316
ASN 6-64: Audit: ASN.1 Definition for CSPEventDataSetTime.....	316
ASN 6-65: Audit: ASN.1 Definition for CSPEventDataGeneralError	317
ASN 6-66: Audit: ASN.1 Definition for CSPEventDataResource	317
ASN 6-67: Audit: ASN.1 Definition for CSPEventDataKeyDerivation.....	318
ASN 6-68: Audit: ASN.1 Definition for CSPEventDataKeyAgreement	318
ASN 6-69: Audit: ASN.1 Definition for CSPEventDataPasswordFailure.....	319
ASN 6-70: Access: ASN.1 Definition for CSPFieldSupport.....	330
ASN 6-71: Audit: ASN.1 Definition for CSPFieldSettings.....	330
ASN 6-72: Signature: ASN.1 Definition for CSPFieldMode.....	330
ASN 6-73: Signature: ASN.1 Definition for CSPField	331

ASN 6-74: Signature: ASN.1 Definition for CSPFieldType.....	332
ASN 6-75: Signature: ASN.1 Definition for CSPFieldSource.....	333
ASN 6-76: Signature: ASN.1 Definition for CSPFieldValue.....	333
ASN 6-77: Access: ASN.1 Definition for CSPPolicySupport.....	340
ASN 6-78: Access: ASN.1 Definition for CSPPolicySettings.....	340
ASN 6-79: Access: ASN.1 Definition for CSPPolicyMode.....	341
ASN 6-80: Access: ASN.1 Definition for CSPPolicy.....	341
ASN 6-81: Access: ASN.1 Definition for CSPPolicyType.....	342
ASN 7-1: CSP Protocol: ASN.1 Definition for the CSPAdminCommand.....	345
ASN 7-2: Admin: ASN.1 Definition for CSPEnforce.....	346
ASN 7-3: Admin: ASN.1 Definition for CSPRegisterClient.....	348
ASN 7-4: Admin: ASN.1 Definition for CSPUnregisterClient.....	349
ASN 7-5: Admin: ASN.1 Definition for CSPCreateResource.....	349
ASN 7-6: Admin: ASN.1 Definition for CSPDestroyResource.....	350
ASN 7-7: Admin: ASN.1 Definition for CSPConfigureResource.....	351
ASN 7-8: Admin: ASN.1 Definition for CSPSetup.....	352
ASN 7-9: Admin: ASN.1 Definition for CSPActivate.....	353
ASN 7-10: Admin: ASN.1 Definition for CSPDeactivate.....	353
ASN 7-11: Admin: ASN.1 Definition for CSPGetConfiguration.....	354
ASN 7-12: Admin: ASN.1 Definition for CSPGetConfigurationResponse.....	354
ASN 7-13: Resource: ASN.1 Definition for CSPSetValue.....	354
ASN 7-14: CSP Protocol: ASN.1 Definition for the CSPClientCommand.....	356
ASN 7-15: Secure Channel: ASN.1 Definition for CSPProcessSecurity.....	358
ASN 7-16: Attestations: ASN.1 Definition for CSPProcessSecurityResponse.....	358
ASN 7-17: Signature: ASN.1 Definition for CSPSign.....	359
ASN 7-18: Attestations: ASN.1 Definition for CSPSignResponse.....	359
ASN 7-19: Signature: ASN.1 Definition for CSPVerifySignature.....	360
ASN 7-20: Attestations: ASN.1 Definition for CSPVerifySignatureResponse.....	360
ASN 7-21: Cipher: ASN.1 Definition for CSPEncrypt.....	361
ASN 7-22: Attestations: ASN.1 Definition for CSPEncryptResponse.....	362
ASN 7-23: Cipher: ASN.1 Definition for CSPDecrypt.....	362
ASN 7-24: Attestations: ASN.1 Definition for CSPDecryptResponse.....	363
ASN 7-25: Attestations: ASN.1 Definition for CSPResourceAttestation.....	363
ASN 7-26: Attestations: ASN.1 Definition for CSPResourceAttestationResponse.....	364
ASN 7-27: Resource: ASN.1 Definition for CSPClearResource.....	365
ASN 7-28: Attestations: ASN.1 Definition for CSPSystemAttestation.....	365
ASN 7-29: Attestations: ASN.1 Definition for CSPSystemAttestationResponse.....	366
ASN 7-30: Key: ASN.1 Definition for CSPGenerateKey.....	367
ASN 7-31: Key: ASN.1 Definition for CSPComputePublicKey.....	368
ASN 7-32: Key: ASN.1 Definition for CSPDeriveKey.....	369
ASN 7-33: Time: ASN.1 Definition for CSPSetTime.....	370
ASN 7-34: Attestations: ASN.1 Definition for CSPSetTimeResponse.....	370

Samples

Sample 4-1: CSP Install Parameters	101
Sample 4-2: CSP INSTALL Command.....	101
Sample 8-1: Retrieve CSP Instance	373

1 INTRODUCTION

This document defines an extension to the *GlobalPlatform Card Specification* ([GP Card Spec]) that specifies the Cryptographic Service Provider (CSP). The CSP provides an API for high-level cryptographic building blocks, such as authentication protocols and secure data management. Its purpose is to support Application developers by providing high-level cryptography and reducing dependencies on platform-specific security guidance.

Further, the CSP aims to simplify the certification process for Applications by eliminating the need for composite certification and enabling Application certification regardless of the underlying platform, while still achieving a security level comparable to Common Criteria EAL4+AVA_VAN.5 for the combination of Application and CSP Platform ([ENISA-EUICC]). These Applications, referred to as Client Applications, are on-card Applications, such as Java Card applets, that rely on cryptography provided solely by the CSP. While the CSP is primarily designed for on-card use, it may also support off-card Clients. However, since off-card Clients require explicit authentication mechanisms to interact with the CSP, their evaluation requires additional assurance measures.

To achieve these objectives, the CSP provides a standardized API for cryptographic services along with general guidance rules for Application developers, thereby eliminating platform-specific adjustments within on-card Applications. While the API is standardized, some functionalities specified in this document are optional. To ensure consistency between different CSP implementations and to prevent the unexpected absence of certain functionalities, this specification supports modularity, as discussed in section 2.3. This approach uses GlobalPlatform configuration documents to harmonize CSP feature sets (e.g., CSP eID Configuration) and allows the automatic detection of supported features and algorithms on a specific platform.

This document is part of a CSP scheme and contains the technical CSP specification for Secure Element (SE) platforms, detailing the functional parts of the CSP; its architecture, services, and cryptography; and its expected behavior in general. The CSP scheme is further composed of:

- CSP interfaces, consisting of a CSP API toward the Client Application [GP CSP API] and the CSP Protocol toward off-card entities, e.g., a management backend [GP CSP ASN1]
- GlobalPlatform configuration documents providing the full listing of mandatory features and algorithms for specific use cases
- CSP Common Criteria Protection Profile (CSP PP) that corresponds with this specification as a module of the GlobalPlatform Secure Element Protection Profile [GP PP SE]
- General CSP Guidance rules for Application Developers [GP CSP Gdnc]
- The CSP Executable Load File (CSP ELF), which is the actual implementation of this CSP specification, as well as its CSP Applications – separate instantiations of the CSP ELF on a given platform, each representing an independent CSP Instance that covers a specific use case

The term CSP refers to the CSP Application, while a CSP Platform includes the CSP ELF, CSP interfaces and the platform functionality utilized by the CSP Application.

Note: This specification exclusively covers SE-based CSP Platforms. Other CSP implementations are not considered in this document.

1.1 Audience

The CSP is intended for applications that require a high level of security assurance. This specification is meant for:

- Application developers using cryptography provided by the CSP

- Card manufacturers and developers implementing a CSP
- Assessment laboratories evaluating a CSP

1.2 Document Structure

This document is structured into the following chapters:

- Chapter 2 - *Use Cases and Requirements* explains the motivation for the CSP.
- Chapter 3 - *Features* provides a high-level overview of the CSP functionalities.
- Chapter 4 - *Architecture and Instantiation* describes the CSP system architecture.
- Chapter 5 - *Core Modules* and 6 - *Optional Modules* contain detailed specifications of CSP functionality, structured similarly to Chapter 3 but with full technical details.
- Chapter 7 - CSP Protocol and 8 - CSP API provide details on the CSP interfaces.

This document contains cross-references to navigate between the high-level overview in Chapter 3 and the detailed specifications in other chapters. Cross-references to other sections are *italicized*. Constants and method names appear in Consolas font and link to their definitions.

Example cross-reference: section 3.6.5, *Signature Suites* → Detailed in 6.2.1.3, *Signature Algorithms*.

Example constant and method: Algorithm SIG_ECDSA used through `sig.sign`.

1.3 IPR Disclaimer

Attention is drawn to the possibility that some of the elements of this GlobalPlatform specification or other work product may be the subject of intellectual property rights (IPR) held by GlobalPlatform members or others. For additional information regarding any such IPR that have been brought to the attention of GlobalPlatform, please visit <https://globalplatform.org/specifications/ip-disclaimers>. GlobalPlatform shall not be held responsible for identifying any or all such IPR, and takes no position concerning the possible existence or the evidence, validity, or scope of any such IPR.

1.4 References

The tables below list references applicable to this specification. The latest version of each reference applies unless a publication date or version is explicitly stated.

Table 1-1: Normative References

Standard / Specification	Description	Ref
ANSI X9.62	ANSI X9.62-2005 Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA) November 2005	[X9.62]
ANSI X9.63	ANSI X9.63-2011 (R2017) Public Key Cryptography For The Financial Services Industry – Key Agreement And Key Transport Using Elliptic Curve Cryptography January 2011	[X9.63]

Standard / Specification	Description	Ref
BSI AIS20/31	BSI AIS 20 / AIS 31 A proposal for: Functionality classes for random number generators September 2011 Version 2.0	[AIS 20/31]
BSI TR-03110 Part 1	BSI Technical Guideline Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token Part 1 – eMRTDs with BAC/PACEv2 and EACv1 February 2015 Version 2.20	[TR-03110-1]
BSI TR-03110 Part 2	BSI Technical Guideline Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token Part 2: Protocols for electronic IDentification, Authentication and trust Services (eIDAS) December 2016 Version 2.21	[TR-03110-2]
BSI TR-03110 Part 3	BSI Technical Guideline Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token Part 3: Common Specifications December 2016 Version 2.21	[TR-03110-3]
BSI TR-03111	BSI Technical Guideline Elliptic Curve Cryptography (ECC) June 2018 Version 2.10	[TR-03111]
CEN/EN 419212 Part 1	Application Interface for smart cards used as Secure Signature Creation Devices Part 1: Introduction and common definitions September 2017	[EN-419212-1]
Fiat Shamir	Amos Fiat, Adi Shamir How To Prove Yourself: Practical Solutions to Identification and Signature Problems CRYPTO '86 Lecture Notes in Computer Science Volume 263, Springer, 1987, ISBN 3-540-18047-8	[Fiat-Shamir]
FIDO Alliance CTAP	FIDO Alliance - FIDO2 Client to Authenticator Protocol (CTAP) https://fidoalliance.org/specs/fido-v2.0-ps-20190130 January 2019 Version 2.0	[FIDO2-CTAP]
FIPS PUB 81	Federal Information Processing Standards Publication DES Modes of Operation December 1980	[FIPS 81]

Standard / Specification	Description	Ref
FIPS PUB 180-4	Federal Information Processing Standards Publication Secure Hash Standard (SHS) August 2015	[FIPS 180-4]
FIPS PUB 186-5	Federal Information Processing Standards Publication Digital Signature Standard (DSS) February 2023	[FIPS 186-5]
FIPS PUB 197	Federal Information Processing Standards Publication Advanced Encryption Standard (AES) November 2001	[FIPS 197]
FIPS PUB 198-1	Federal Information Processing Standards Publication The Keyed-Hash Message Authentication Code (HMAC) July 2008	[FIPS 198-1]
GlobalPlatform Algorithm Recommendations	GlobalPlatform Technology (GPC_TEN_053) Cryptographic Algorithm Recommendations April 2025 Version 3.0 or higher	[GP Crypto Rec]
GP Card Specification	GlobalPlatform Technology (GPC_SPE_034) Card Specification March 2018 Version 2.3.1 or higher	[GP Card Spec]
GP Amendment A Card Content Management	GlobalPlatform Technology (GPC_SPE_007) Card Specification Amendment A: Confidential Card Content Management July 2019 Version 1.2 or higher	[GP Amd A]
GP Amendment D SCP03	GlobalPlatform Technology (GPC_SPE_014) Card Specification Amendment D: Secure Channel Protocol '03' April 2020 Version 1.2 or higher	[GP Amd D]
GP Amendment K SCP04	GlobalPlatform Technology (GPC_SPE_182) Card Specification Amendment K: Secure Channel Protocol '04' January 2024 Version 1.0.1 or higher	[GP Amd K]
GP API Java Card API	GlobalPlatform API (org.globalplatform) Card Specification Java Card API and Export File for Card Specification Version 1.8 or higher	[GP API]
GP API Java Card CSP API	GlobalPlatform API (org.globalplatform.csp) Card Specification Java Card API and Export File for CSP Version 1.0 or higher	[GP CSP API]

Standard / Specification	Description	Ref
GP API ASN.1 CSP Protocol	GlobalPlatform API (csp-protocol.asn) Card Specification ASN.1 Definition File for CSP Version 1.0 or higher	[GP CSP ASN1]
GP DLOA	Global Platform Technology (GPC_SPE_095) Digital Letter of Approval December 2015 Version 1.0 or higher	[GP DLOA]
GP Privacy Framework	GlobalPlatform Technology (GPC_SPE_100) Card Specification Privacy Framework November 2019 Version 1.0.1 or higher	[GP Privacy Fmwk]
ICAO Doc 9303 Part 11	International Civil Aviation Organization (ICAO) Doc 9303 Machine Readable Travel Documents Part 11: Security Mechanisms for MRTDs September 2021 8 th Edition	[ICAO 9303-11]
IEEE 1363-2000	Institute of Electrical and Electronics Engineers Standard Specification for Public-Key Cryptography January 2000	[IEEE 1363-2000]
IEEE 1619-2018	Institute of Electrical and Electronics Engineers Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices January 2019	[IEEE 1619-2018]
IETF RFC: 3610	Internet Engineering Task Force Counter with CBC-MAC (CCM) https://www.ietf.org/rfc/rfc3610.txt September 2003	[RFC 3610]
IETF RFC: 5639	Internet Engineering Task Force Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation https://www.ietf.org/rfc/rfc5639.txt March 2010	[RFC 5639]
IETF RFC: 5652	Internet Engineering Task Force Cryptographic Message Syntax (CMS) https://www.ietf.org/rfc/rfc5652.txt September 2009	[RFC 5652]
IETF RFC: 5758	Internet Engineering Task Force Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA https://www.ietf.org/rfc/rfc5758.txt January 2010	[RFC 5758]

Standard / Specification	Description	Ref
IETF RFC: 5869	Internet Engineering Task Force HMAC-based Extract-and-Expand Key Derivation Function (HKDF) https://www.ietf.org/rfc/rfc5869.txt May 2010	[RFC 5869]
IETF RFC: 7748	Internet Engineering Task Force Elliptic Curves for Security https://www.ietf.org/rfc/rfc7748.txt January 2016	[RFC 7748]
IETF RFC: 8017	Internet Engineering Task Force PKCS #1: RSA Cryptography Specifications Version 2.2 https://www.ietf.org/rfc/rfc8017.txt November 2016	[RFC 8017]
IETF RFC: 8235	Internet Engineering Task Force Schnorr Non-interactive Zero-Knowledge Proof https://www.ietf.org/rfc/rfc8235.txt September 2015	[RFC 8235]
IETF RFC: 8017 PKCS #1	Internet Engineering Task Force PKCS #1: RSA Cryptography Specifications Version 2.2 https://www.ietf.org/rfc/rfc8017.txt November 2016	[PKCS #1]
IETF RFC: 2898 PKCS #5	Internet Engineering Task Force PKCS #5: Password-Based Cryptography Specification https://www.ietf.org/rfc/rfc2898.txt September 2000 Version 2.0	[PKCS #5]
ISO/IEC 7816 Part 4	Identification cards – Integrated circuit card Part 4: Organization, security and commands for interchange https://www.iso.org/standard/77180.html May 2020 4 th Edition	[ISO 7816-4]
ISO/IEC 7816 Part 5	Identification cards – Integrated circuit card Part 5: Registration of application providers https://www.iso.org/standard/34259.html May 2004 2 nd Edition	[ISO 7816-5]
ISO/IEC 9797 Part 1	Information technology, Security techniques Message Authentication Codes (MACs) Part 1: Mechanisms using a block cipher https://www.iso.org/standard/50375.html March 2011 2 nd Edition	[ISO 9797-1]

Standard / Specification	Description	Ref
ISO/IEC 10116:2017	Information technology, Security techniques Modes of operation for an n-bit block cipher https://www.iso.org/standard/64575.html July 2017 4 th Edition	[ISO 10116]
ISO/IEC 18013 Part 5	Information technology, Security techniques Personal identification - ISO-compliant driving licence Part 5: Mobile driving licence (MDL) application https://www.iso.org/standard/69084.html September 2021 1 st Edition	[ISO 18013-5]
ITU-T X.509	Information technology – Open Systems Interconnection The Directory: Public-key and attribute certificate frameworks https://www.itu.int/rec/T-REC-X.509-201910-I/en October 2019	[ITU-T X.509]
Java Card API	Java Card™ Platform Application Programming Interface, Classic Edition https://docs.oracle.com/javacard/3.0.5/api/index.html Version 3.0.5 or higher	[JCAPI]
Java Card JCRE	Java Card™ Platform, Oracle Corporation Runtime Environment Specification, Classic Edition Version 3.0.5 or higher	[JCRE]
NIST SP 800-38B	NIST Special Publication SP 800-38B: Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication https://csrc.nist.gov/pubs/sp/800/38/b/upd1/final June 2016	[SP800-38B]
NIST SP 800-38C	NIST Special Publication SP 800-38C: Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality https://csrc.nist.gov/pubs/sp/800/38/c/upd1/final July 2007	[SP800-38C]
NIST SP 800-38D	NIST Special Publication SP 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC https://csrc.nist.gov/pubs/sp/800/38/d/final November 2007	[SP800-38D]
NIST SP 800-56C	NIST Special Publication SP800-56C: Recommendation for Key-Derivation Methods in Key-Establishment Schemes https://csrc.nist.gov/pubs/sp/800/56/c/r2/final August 2020	[SP800-56C]

Standard / Specification	Description	Ref
NIST-SP 800-186	NIST Special Publication SP800-186: Recommendation for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters https://csrc.nist.gov/pubs/sp/800/186/final February 2023	[SP800-186]
TCA eUICC Profile	Trusted Connectivity Alliance eUICC Profile Package: Interoperable Format Technical Specification https://trustedconnectivityalliance.org/technology_library/euicc-profile-package-technical-specifications/ July 2023, Version 3.3.1	[TCA eUICC]

Table 1-2: Informative References

Standard / Specification	Description	Ref
BSI CSP Whitepaper	Overview of the concept for a Cryptographic Service Provider. December 2022 Version 1.0 or higher	[BSI-CSP-WP]
BSI TR-03181	BSI Technical Guideline of the Cryptographic Service Provider 2 (CSP2). April 2023 Version 0.94 or higher	[TR-03181]
ENISA eUICC Certification	Specifications for eUICC Certification under the EUCC scheme. June 2024 Version 1.0 or higher https://certification.enisa.europa.eu/publications/specifications-related-certification-embedded-universal-integrated-circuit-card_en	[ENISA-EUICC]
Common Criteria Part 3	Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components November 2022, CC:2022 Revision 1	[CC-3]
Attestation Proof of Association	Eric Verheul, Logius Attestation Proof of Association - provability that attestation keys are bound to the same hardware and person Cryptology ePrint Archive, Paper 2024/1444 https://eprint.iacr.org/2024/1444 January 2025 Version 1.2 or higher	[Verheul-PoA]
GP CSP Guidance for Applet Developer	GlobalPlatform Technology (GPC_GUI_237) CSP Guidance for Applet Developer	[GP CSP Gdnc]
GP SE PP	GlobalPlatform Technology (GPC_SPE_174) Secure Element Protection Profile	[GP PP SE]

Standard / Specification	Description	Ref
JIL Composite Evaluation	Joint Interpretation Library Composite product evaluation for Smartcards and similar devices May 2018 Version 1.5.1	[JCPE]

1.5 Terminology and Definitions

Selected terms used in this document are listed in Table 1-3 for components, Table 1-4 for data structures, Table 1-5 for roles, and Table 1-6 for interfaces.

1.5.1 Components

Table 1-3 defines terms covering components of the CSP.

Table 1-3: Terminology and Definitions: Components

Term	Definition
Application	Instance of an Executable Module after it has been installed. ([GP Card Spec]) Note: This means the Application is installed and operated on-card, with each Application having a unique Application Identifier (AID) used for selection and communication on the Secure Element. The AID consists of 5 to 16 bytes, with the first 5 bytes being the Registered Application Provider Identifier (RID), assigned by ISO through a global registry to ensure uniqueness ([ISO 7816-5]).
Client Application	An Application, installed and operated on the same platform as the CSP Application, that utilizes cryptographic services provided by the CSP Application, which are specifically tailored to a particular use case. Each Client Application, identified by its AID, must be registered with the CSP Application to access these services. Once registered, Client Applications can invoke the CSP Services via the CSP API, using CSP Resource Identifiers that refer to the cryptographic keys and algorithms preconfigured by the CSP Admin.
CSP Application (aka CSP)	An Application that provides CSP Services according to this specification. It is instantiated from the CSP ELF through the GlobalPlatform INSTALL Command ([GP Card Spec]), using a standardized CSP ELF AID. Multiple CSP Applications can coexist on the same Secure Element, each within its own Security Domain (SD), representing a dedicated use case. The CSP Application is configured by the CSP Admin via the CSP Protocol over the GlobalPlatform Personalization interface ([GP Card Spec]), customizing it for the specific use case, with the necessary cryptographic resources. Its cryptographic services can be utilized by registered Client Applications via the CSP API.
CSP ELF	The Executable Load File (as defined in [GP Card Spec]) containing the compiled source code of the CSP Application. The CSP ELF exists only once on each Secure Element and has a fixed AID.

Term	Definition
CSP Instance	A specific object instance of the CSP API interface (<code>org.globalplatform.csp.api.CSP</code>) that is accessible to registered and authorized Client Applications, enabling them to invoke CSP Services that have been pre-configured for a dedicated use case. Essentially, 'CSP Instance' and 'CSP Application' can be used synonymously because, in this specification, each CSP Application supports exactly one CSP Instance. Thus, distinct use cases that wish to use a CSP according to this specification require the instantiation of dedicated CSP Applications, each providing exactly one CSP Instance.
CSP Module (aka module)	A unit of code that extends the core functionality of the CSP Application with specific cryptographic or security-related operations. It can be developed, tested, and maintained independently, allowing for modularity and customization of CSP implementations. This specification defines, for example, the Cipher Module, Signature Module, Secure Channel Module, Attestation Module, Time Module, and Audit Module.
CSP Platform	The term 'CSP-enabled Secure Element Platform' (CSP Platform) refers to the entire CSP implementation, including the CSP ELF, the CSP interfaces, and the underlying platform functionalities. The CSP Platform manages cryptographic keys, certificates, and passwords as CSP Resources that are configurable by CSP Admins through the CSP Protocol. Distinct use cases are handled by different CSP Instances. Each CSP Instance offers pre-configured services tailored for a specific use case, which can be utilized by Client Applications through the CSP API.
CSP Service (aka service)	An instance of a CSP Module. For example, the Cipher Service is an instance of the Cipher Module. Client Applications can invoke operations provided by such a service using specific Resource Identifiers that refer to a part of the CSP Configuration, such as a cipher configuration containing a cryptographic key with a defined key type, size, and value, along with the padding and cipher algorithm. The CSP executes service operations within its secure environment, isolating and safeguarding cryptographic keys from the Client Applications and other Applications.
module	See <i>CSP Module</i> .
off-card Client	An off-card entity that uses CSP Services through the <code>CSPClientCommand</code> of the CSP Protocol. It must authenticate to the CSP Application before it can access these services.
Secure Element (SE)	A tamper-resistant secure hardware component that is used in a device to provide the security, confidentiality, and multiple application environment required to support various business models. May exist in any form factor, such as embedded or integrated SE, SIM/UICC, smart card, smart microSD, etc.
Security Domain (SD)	Application having the Security Domain privilege. This on-card entity provides support for the control, security, and communication requirements of an off-card entity such as the Card Issuer, an Application Provider, or a Controlling Authority. Note: In essence, the Security Domain is a specific, secure area within the Secure Element, dedicated to a particular use case, ensuring the isolation and protection of sensitive operations and data.
service	See <i>CSP Service</i> .

1.5.2 Data Structures

Table 1-4 defines terms covering data structures related to the CSP.

Table 1-4: Terminology and Definitions: Data Structures

Term	Definition
certificate	See <i>CSP Certificate</i> .
configuration	See <i>CSP Configuration</i> .
CSP Certificate (aka certificate)	A CSP Resource of type <code>RESOURCE_CERTIFICATE</code> . It is a container holding the configuration of a digital certificate.
CSP Configuration (aka configuration)	A security configuration that contains all cryptographic keys, certificates, and passwords required for a specific use-case, along with their purpose, algorithm, access control, and other parameters necessary for the CSP to provide use-case-specific cryptographic services through the CSP API. Each CSP Application has exactly one CSP Instance with one CSP Configuration.
CSP Key (aka key)	A CSP Resource of type <code>RESOURCE_KEY</code> . It is a container holding the configuration parameters and the value of a cryptographic key.
CSP Password (aka password)	A CSP Resource of type <code>RESOURCE_PASSWORD</code> . It is a container holding the configuration of a passphrase or PIN.
CSP Resource (aka resource)	A CSP-internal data structure that represents either a key, certificate, password, counter, or timer along with all parameters required to process a specific service provided by the CSP. Depending on the purpose of the resource, these parameters may vary. For example, an attestation key used within the CSP's Attestation Service requires the configuration of its key size, curve, signature algorithm, and access-restriction for attestation usage by Client Applications, preventing its use within the Signature Service.
CSP Resource Identifier (aka resourceId)	The unique identifier of a CSP Resource, referring to its cryptographic and security-related parameters stored in the CSP Configuration of a CSP Instance.
CSP Resource Value (aka resource value)	The value of a CSP Resource. Depending on the type of the resource, this may include a symmetric, private, or public key value; the content of a certificate; or a specific PIN or passphrase.
key	See <i>CSP Key</i> .
password	See <i>CSP Password</i> .
resource	See <i>CSP Resource</i> .
resource value	See <i>CSP Resource Value</i> .
resourceId	See <i>CSP Resource Identifier</i> .

1.5.3 Roles

Table 1-5 defines terms covering roles related to the CSP.

Table 1-5: Terminology and Definitions: Roles

Term	Definition
CSP Admin	An entity authorized to access the Security Domain associated with the CSP Application. This entity is responsible for setting up and configuring a CSP Instance before it can be used by CSP Clients. This configuration process utilizes the GlobalPlatform Personalization interface ([GP Card Spec]) to, among other actions, create keys, certificates, and passwords as CSP Resources; configure their cryptographic algorithms; and grant usage permissions.
CSP Client	An on-card or off-card entity that uses the cryptographic services provided by a CSP Application. Both Client Applications and off-card Clients are considered CSP Clients.
SE Admin	An administrative entity authorized to access the Authorized Management Security Domain (AMSD) of the SE. It is responsible for creating new Security Domains (SDs) for both Client and CSP Applications and for instantiating the CSP Application from the CSP ELF into a dedicated SD for the CSP. Examples of SE Admins include SE Issuer, Original Equipment Manufacturer (OEM), Mobile Network Operator (MNO), Trusted Service Manager (TSM), or Secured Applications for Mobile Manager (SAM Manager).

1.5.4 Interfaces

Table 1-6 defines terms covering interfaces related to the CSP.

For further information, see section 4.1, Architecture.

Table 1-6: Terminology and Definitions: Interfaces

Term	Definition
APDU Interface	The communication layer through which commands and data are exchanged directly between an on-card Application and an off-card application, e.g., a terminal or mobile app. It allows the off-card application to send APDUs directly to the on-card Application using its unique Application Identifier (AID) ([ISO 7816-4]), enabling direct interaction without an intermediary Security Domain.
CSP API	An interface definition used for communication between a Client Application and a CSP Instance. It provides cryptographic and security-related services based on CSP Resource Identifiers that refer to a specific resource within the CSP Configuration. The CSP API is delivered as a Global Service and is composed of two packages: <code>org.globalplatform.csp</code> and <code>org.globalplatform.csp.api</code> .
CSP Protocol	A standardized data encoding scheme based on Tag Length Value (TLV) structures, used for communication between an off-card entity and a CSP Instance. It provides administrative commands to configure the CSP Instance, including resource creation, Client Application registration, and CSP activation for operational use, along with structures for selected CSP service operations (e.g., Platform Attestation).
Global Service	The GlobalPlatform mechanism that provides services to other Applications on the SE, as specified in [GP Card Spec].

Term	Definition
GlobalPlatform Installation Procedures	Standardized commands and processes for securely installing and managing Applications on a Secure Element. Includes commands such as INSTALL [for load] and INSTALL [for install] to load and install Application packages. (See [GP Card Spec].)
GlobalPlatform Personalization interface	The communication interface between off-card application and on-card Application through a Security Domain (SD) using GlobalPlatform's Secure Channel Protocols (SCP). This interface enables the off-card application to perform administrative tasks such as storing and updating personalized data by sending APDUs (e.g., Store Data APDU) securely through the SD ([GP Card Spec]).
Store Data APDU	A command structure used to transmit data to an Application, either through the APDU Interface or the GlobalPlatform Personalization interface. It is typically used to store or update data within the Secure Element ([ISO 7816-4]).

1.6 Abbreviations

Selected abbreviations used in this document are included in Table 1-7.

Table 1-7: Abbreviations

Abbreviation	Meaning
AAD	Additional Associated Data
ACR	Access Control Rules
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
AID	Application Identifier
AMSD	Authorized Management Security Domain
APDU	Application Protocol Data Unit
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
AVA	Assurance Vulnerability Assessment
AVA_VAN	Assurance Vulnerability Assessment – Vulnerability Analysis
AVA_VAN.5	Assurance Vulnerability Assessment – Vulnerability Analysis Level 5
CA	Chip Authentication
CAN	Card Access Number
CASD	Controlling Authority Security Domain
CBC	Cipher Block Chaining Mode
CC	Common Criteria
CCM	Counter CBC Mode
CFB	Cipher Feedback Mode
CLA	Class Byte

Abbreviation	Meaning
CMAC	Cipher-Based Message Authentication Code
C-MAC	Command Message Authentication Code
COR	Clear On Reset
CS	Cryptographic Service
CSP	Cryptographic Service Provider
CTAP2	Client to Authenticator Protocol
CTS	Cipher Text Stealing
CVC	Card Verifiable Certificates
DAP	Data Authentication Pattern
DEK	Data Encryption Key
DES	Data Encryption Standard
DLOA	Digital Letter Of Approval
EAC	Extended Access Control
EAL	Evaluation Assurance Level
EAL4	Evaluation Assurance Level 4
EAL4+AVA_VAN.5	Evaluation Assurance Level 4 augmented with AVA_VAN.5
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman key agreement
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
ECKA-DH	Elliptic Curve Key Agreement – Diffie Hellman
ECKA-EG	Elliptic Curve Key Agreement – ElGamal scheme
ECSDSA	Elliptic Curve Based Schnorr Digital Signature Algorithm
eID	Electronic (digital) Identity
ELF	Executable Load File
EM	Executable Module
eSE	Embedded Secure Element
eUICC	Embedded Universal Integrated Circuit Card
FIDO2	Fast Identity Online 2
GCM	Galois Counter Mode
GP	GlobalPlatform
HKDF	HMAC Key Derivation Function
HMAC	Hash-based Message Authentication Code
INS	Instruction Byte

Abbreviation	Meaning
IV	Initialization Vector
KDF	Key Derivation Function
LFDBH	Load File Data Block Hash
LSB0	Least Significant Bit at position 0
MAC	Message Authentication Code
MDL	Mobile Driving Licence
MRTD	Machine Readable Travel Documents
MRZ	Machine Readable Zone
NIZK	(Schnorr) Non-Interactive Zero-Knowledge
OAEP	Optimal Asymmetric Encryption Padding
OCE	Off-card Entity
OFB	Output Feedback Mode
OTP	One-Time Password
PACE	Password Authenticated Connection Establishment
PACE-CAM	PACE Connection with Chip Authentication Mapping
PBKDF	Password-Based Key Derivation Function
PKCS	Public-Key Cryptography Standards
PKI CRL	Public Key Infrastructure Certificate Revocation List
PoA	Proof of Association for cryptographic keys
PoP	Proof of Possession for cryptographic keys
PP	Protection Profile
PUK	Personal Unblocking Key
PWD	Password
R-ENC	Response Encryption
RFU	Reserved for Future Use
RID	Registered Application Provider Identifier
R-MAC	Response MAC
RNG	Random Number Generator
RSA	Rivest Shamir Adleman
SAM	Secured Applications for Mobile
SCP	Secure Channel Protocol
SD	Security Domain
SE	Secure Element
S-ENC	Secure Channel Session Encryption Key

Abbreviation	Meaning
SIO	Shareable Interface Object
S-MAC	Secure Channel C-MAC Session Key
S-RMAC	Secure Channel R-MAC Session Key
SRNG	Secure Random Number Generator
SW	Status Word
TA	Terminal Authentication
TLV	Tag Length Value
TSM	Trusted Service Manager
UTF-8	8-Bit Universal Character Set Transformation Format
XTS	XEX-based Tweaked Codebook mode with ciphertext Stealing
ZKP	Zero-Knowledge-Proof

1.7 Revision History

GlobalPlatform technical documents numbered $n.0$ are major releases. Those numbered $n.1$, $n.2$, etc., are minor releases where changes typically introduce supplementary items that do not impact backward compatibility or interoperability of the specifications. Those numbered $n.n.1$, $n.n.2$, etc., are maintenance releases that incorporate errata and clarifications; all non-trivial changes are indicated, often with revision marks.

Table 1-8: Revision History

Date	Version	Description
May 2024	0.0.0.22	Committee Review
Jan 2025	0.0.0.30	Member Review
May 2025	0.0.0.40	Public Review
TBD	v1.0	Public Release

2 USE CASES AND REQUIREMENTS

This specification defines a platform that simplifies the usage of typical high-level cryptography, prevents misuse of cryptographic functions, minimizes the need for usage guidance, separates the use of cryptographic functionality from governing asset management, and streamlines the certification process for solutions requiring a high level of assurance.

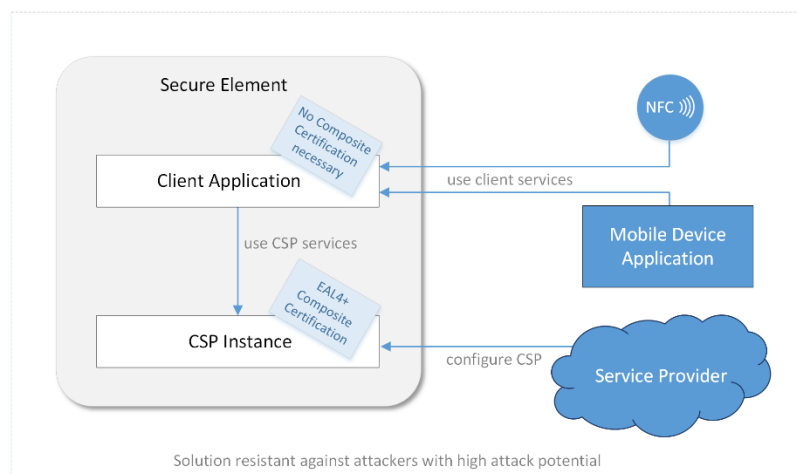
2.1 Use Cases

The CSP scheme is relevant for on-card Applications that require proof of protection against attackers with an attack potential of up to 'high'. To obtain this proof, these Applications must undergo an Assessment of Vulnerabilities (AVA) as outlined in [CC-3] section 14.3, including a Vulnerability Analysis (VAN) at the appropriate level.

The CSP scheme enables certification of Client Applications independently from the underlying platform, i.e., without requiring composite certification according to [JCPE]. In the scheme, Client Applications are certified for use with a CSP-enabled Secure Element platform that provides self-contained and pre-configured cryptographic services. For Client Applications that exclusively rely on CSP functionality for their cryptographic needs and do not implement custom cryptography, security functions, or sensitive operations, the entire solution is expected to remain resistant against attackers with high attack potential, as illustrated in Figure 2-1.

The cryptographic abstraction layer provided by the CSP is designed to serve a wide range of applications, including digital identity (eID), payment systems, ticketing, access control, healthcare, automotive, and other use cases with enhanced security assurance requirements.

Figure 2-1: CSP Concept



The introduction of a CSP is especially beneficial for Client Applications operating on an unknown number of different platforms, which is a typical scenario for mobile devices equipped with embedded Secure Elements (eSE) or embedded Universal Integrated Circuit Cards (eUICC). Considering the complexity, difficulty, length, and cost associated with composite certification for every possible platform, the use of CSP would address these issues by allowing a once-certified Client Application to be loaded on any CSP-enabled SE platform while still maintaining the certification status of the Client Application.

2.2 Requirements

The CSP serves as a cryptographic abstraction layer to SE platforms to simplify the Common Criteria (CC) certification process for Client Applications using the CSP Application ([BSI-CSP-WP]). To separate the use of cryptographic assets from their management, the CSP Application operates in two phases: a configuration phase and an operational phase. During the configuration phase, a CSP Admin sets up keys, certificates, and passwords, along with their cryptographic values, algorithms, constraining policies, and other parameters for a specific use case. This CSP Configuration is applied to a dedicated CSP Instance that exclusively serves the use case implemented by the appropriate CSP Client(s). In the operational phase, those CSP Clients can utilize the pre-configured cryptographic services provided by the CSP Instance.

One design decision of this specification is to realize the CSP as an extension to the GlobalPlatform Card Specification, without modifying any core features of the card platform. Consequently, the CSP Application manages key, certificate, and password configurations on top of the GlobalPlatform Security Domain layer. Thereby, it utilizes the Security Domain (SD) mechanism by enabling CSP Admins to configure the CSP Application using the GlobalPlatform Personalization interface ([GP Card Spec]). Such a CSP Admin is authorized to access the SD associated with the CSP Application, allowing the CSP Admin to create the CSP Configuration for its CSP Instance. Multiple CSP Applications may be administered by independent CSP Admins, supporting use-case-specific cryptographic services for different Client Applications on the same SE platform without interfering with each other.

This specification combines key management with selected cryptographic functionalities carefully chosen to provide CSP Services (services), covering widespread security-related scenarios as described in [TR-03181], including:

- *Cipher and Signatures*: Similar to those offered by [JCAPI], but when using the CSP, a CSP Admin configures cryptographic keys and algorithms specifically for a defined purpose, i.e., by selecting appropriate algorithms, key sizes, access constraints, and policies. The Client Application then simply invokes the pre-configured Cipher and/or Signature Service, without the need to maintain any cryptographic keys or algorithms.
- *Encryption Transformations*: Transform the encryption of data directly from one key and algorithm to another key and/or algorithm, preventing Client Applications from accessing the data in plain format.
- *Secure Messaging*: Similar to the [GP Card Spec] and [GP Privacy Fmwk], the CSP manages the complete secure channel protocol flow. However, the CSP supports multiple protocol instances using non-global key sets and is compatible with existing GlobalPlatform card platforms.
- *Confidential Data Transfer*: Transfers confidential data from session encryption to long-term storage encryption, ensuring that the Client Application never accesses the confidential data, i.e., preserving confidentiality of data-in-use and data-in-transit on a high assurance level.
- *Attestations*: Provides various attestation mechanism for different components, such as CASD-based attestation to prove the authenticity and identity of the SE platform; attestation of the CSP Instance using a use-case-specific CSP attestation key; attestations of public keys, timers, and counters managed by the CSP; a Proof of Possession (PoP) to demonstrate that a corresponding private key is also held by the same CSP Instance; and Proof of Association (PoA) that uses Zero-Knowledge Proof (ZKP) algorithms to provide a privacy-friendly verification that two keys are associated with each other.
- *Key Management*: Provides key management services, including key generation, key derivation, and key agreement, similar to those offered by [JCAPI]. The difference is that a CSP Admin configures the algorithms, the purpose, and the access control rules for these key management operations, allowing Client Applications to invoke them only for dedicated tasks in specific scenarios. For example, key A is configured to be used exclusively to compute a shared secret; this shared secret is then used solely to derive a key B and the derived key B is restricted to be used with a specific cipher algorithm.

- **Password Management:** Facilitates fine-grained management of passphrases and PINs, including changing passwords, password verification with retry counter management, and password blocking and unblocking functionality.
- **Certificate Management:** Enables Client Applications to perform certificate management operations, such as policy-restricted replacement of certificates or initiating TA2 certificate rollovers managed by the CSP.
- **Counter and Limits:** Supports secure counters that can be manually incremented by Client Applications or automatically maintained by the CSP, such as usage counters that increment each time a specific key is used in a cryptographic operation. The CSP Admin may configure limits on these counters, which are enforced by the CSP, e.g., to prevent keys with exhausted usage counters from being used in further cryptographic operations. Client Applications may retrieve cryptographically signed counter values by using signature fields.
- **Timer and Time Management:** Estimates system time, for instance, by updating a reference time from Terminal Authentication (TA) certificates during secure channel establishment. This system time is then used to evaluate timers, such as validity periods configured for keys, certificates, or passwords. Client Applications may also set manual timers, for example, to verify if a person is over 18 years old.
- **Secure Auditing:** Logs asset usage events (e.g., signature creations), security incidents (e.g., password failures), and system events (e.g., time updates). The CSP ensures the integrity of these logs by incorporating a counter and/or timestamp, and signing it.
- **Access Control:** Supports the configuration of access control rules for keys, certificates, and passwords, restricting their access by Client Applications as required by specific use cases. Additionally, policy rules can define relationships between these resources that are enforced by the CSP (e.g., PIN-PUK relationship).

To implement these services, the CSP scheme uses CSP Resources (resources) as abstractions of cryptographic assets. A resource is a container for the cryptographic value, e.g., of a PIN or cryptographic key, and its associated attributes and properties, including access control and constraining policies. Each resource has a unique CSP Resource Identifier (`resourceId`) within the scope of the defined use case. Therefore, at least the entity acting as CSP Admin and the developer of the Client Application need to have a common agreement on the required resources, including their `resourceIds`, that define their use case. Rules on how to create such a CSP Configuration are described in the general CSP Guidance Rules for Client Application Developers [GP CSP Gdnc].

The Client Application initializes a CSP Service by passing a `resourceId` to the CSP API. The CSP then executes the service operation in an isolated environment using the specific cryptographic configuration associated with the provided `resourceId`. Consequently, Client Applications cannot access the actual values of keys, passwords, or other secrets, nor can they modify their cryptographic configurations. Additionally, Client Applications are restricted from using these resources for any purposes other than those for which they were configured.

Finally, using the CSP's cryptographic services unburdens Client Applications from implementing security-related functionality and reduces the need to consider platform-specific security guidance. This approach allows for the separation of security-related functionality and not-sensitive business logic of a Client Application, thereby enhancing overall system security.

To achieve all these objectives, the CSP shall meet the following essential requirements:

- The CSP shall support the *CSP Protocol* as specified in this document.
- The CSP shall support the *CSP API* as specified in [GP CSP API].
- The CSP shall support the *Architecture and Instantiation* as specified in this document.
- The CSP shall support all features, types, and algorithms marked as mandatory in this document.

- The CSP shall support the CSPEnforce command, which enables the detection of whether optional features and/or algorithms are supported by a specific platform.
- The CSP shall not require any vendor-specific guidance rules, i.e., it shall not request vendor-specific security-related instructions for CSP usage that could impact the security of a Client Application intended for certification without composite certification. This means the CSP is prohibited from imposing platform-specific security requirements on the Client Application, ensuring a uniform approach to security without necessitating vendor-specific implementations.

In addition to these requirements for CSP Implementations, Client Applications using a CSP must also meet certain requirements to gain certification benefits. For detailed information on these, please refer to [GP CSP Gdnc].

2.3 Modularity

This CSP specification supports modularity, allowing different CSP implementations to support varying features and algorithms (see Table 2-1 for an overview). Modularity applies not only to features but also at the algorithm level. This means the cryptographic algorithms and type definitions specified in Chapter 6 are optional and may be individually included or excluded from CSP implementations.

Table 2-1: Optional Modules, Algorithms, and Types

Modules	Cryptographic Types	Data Types	Operation Modes
<i>Cipher Module</i>	<i>Cipher Algorithms</i>	<i>Resource Types</i>	<i>Error Modes</i>
<i>Signature Module</i>	<i>Message Digest Algorithms</i>	<i>Fields</i>	<i>Field Modes</i>
<i>Transform Module</i>	<i>Signature Algorithms</i>	<i>Key Types</i>	<i>Policy Modes</i>
<i>Secure Channel Module</i>	<i>Protocol Types</i>	<i>Certificate Types</i>	<i>Counter Modes</i>
<i>Confidential Data Transfer Module</i>	<i>Key Derivation Algorithms</i>	<i>Password Types</i>	<i>Time Modes</i>
<i>Attestation Module</i>	<i>Key Agreement Schemes</i>	<i>Policy Types</i>	<i>Audit Modes</i>
<i>Key Module</i>	<i>Resource Attestation Types</i>	<i>Counter Types</i>	
<i>Certificate Module</i>	<i>Time Synchronization Methods</i>	<i>Counter Capacities</i>	
<i>Password Module</i>		<i>Timer Types</i>	
<i>Counter Module</i>		<i>Timeout Types</i>	
<i>Time Module</i>		<i>Audit Events</i>	
<i>Audit Module</i>		<i>Policy Types</i>	
<i>Offloading Module</i>			
<i>Field Module</i>			
<i>Policy Module</i>			
<i>Random Data Module</i>			

To support this modular approach in practice, the CSP shall implement the CSPEnforce command, allowing CSP Admins to query which features and algorithms are supported on a given platform.

The decision to implement optional features should be documented in a configuration document tailored to a specific use case. This configuration document should clearly indicate which modules, cryptographic types, data types, and operation modes (as defined in Table 2-1) shall be implemented, thereby removing any ambiguity regarding optional capabilities. If available, official GlobalPlatform configuration documents should be used as a basis to ensure consistent feature sets across different vendors.

For further information, see section 1, Introduction, and section 2.2, Requirements.

2.4 Exclusions and Limitations

2.4.1 Post-Quantum Cryptography (PQC) Considerations

At the time of creating this CSP specification, Post-Quantum Cryptography (PQC) algorithms have not been considered. These algorithms represent the next generation of cryptographic security and are designed to resist attacks enabled by quantum computers. They will be evaluated for inclusion in future updates of this document.

However, the sizes for public keys, challenges, and signatures within the CSP Protocol have already been selected with proper flexibility to accommodate future needs. These sizes are defined to ensure compatibility with the larger key sizes, challenges and signatures expected in post-quantum cryptographic algorithms.

2.4.2 Legacy Cryptographic Algorithms

The following cryptographic algorithms supported by the CSP are considered legacy ([GP Crypto Rec]):

- PAD_NOPAD (Exact-Length Required), when used with CIPHER_RSA or SIG_RSA.
- PAD_PKCS1 padding scheme.
- KEY_RSA_2048 bit for CIPHER_RSA and SIG_RSA.
- SIG_AES_MAC128 signature.
- SHA-1 is used for key derivation within the *PACE* protocol. However, SHA-1 is not available as a CSP service and is not used in any other cryptographic implementation within the CSP.

CSP Clients should not use these legacy algorithms in any new implementations.

For further information, see section 3.6.2, Hash Message Digest.

2.4.3 Interactions with Certificate Authorities

The CSP supports certificate operations but does not facilitate direct interactions with Certificate Authorities or manage blocklists. This includes capabilities such as certificate issuance, renewal, revocation and rejecting compromised certificates. For these functionalities, external systems or additional software will be required.

For further information, see section 3.4.6.1, Certificate Signing, section 3.4.6.2, Certificate Overlap, and section 3.4.6.3, Certificate Revocation.

3 FEATURES

3.1 Resource Management

The CSP shall securely manage CSP Resources, which represent cryptographic keys, digital certificates, passphrases, PINs, counters or timers, along with all necessary parameters required to process specific cryptographic services.

An administrative interface enables authorized CSP Admins to create, configure and securely delete these resources as needed for specific use cases, using the `CSPAdminCommand` of the CSP Protocol. The CSP shall store these resource configurations, along with dynamic resource parameters, such as the resource state, in persistent memory, unless otherwise specified for a specific parameter (e.g., the authenticated flag of a password is stored in volatile memory).

The CSP shall utilize these resource configurations to provide cryptographic services such as *Key Management*, *Certificate Management*, *Password Management*, *Cipher and Signatures*, *Secure Messaging*, *Confidential Data Transfer*, *Attestations*, and *Secure Auditing*.

Client Applications may utilize these services without needing to specify any cryptographic details. For example, to invoke a signing operation, a Client Application only needs to provide a CSP Resource Identifier and the data to be signed. This identifier refers to a signing key resource that already includes the key parameters and algorithm configuration necessary for executing the signing process.

Processing of these resources shall always occur within the tamper-proof and certified scope of the CSP Application.

For further information, see section 5.1, System Module, and section 5.2, Resource Module.

3.1.1 Create Resources

The CSP shall support the creation of CSP Resources using the `CSPCreateResource` command. This process involves allocating memory for the parameters of these resources.

The CSP may support the following resource types; the exact list shall be subject to *Modularity*.

- Keys: Cryptographic key values with key size and other parameters, as specified for `RESOURCE_KEY`.
- Certificates: Digital certificates containing public keys, as specified for `RESOURCE_CERTIFICATE`.
- Passwords: Passphrases or PINs with (re)try limit and other parameters, as specified for `RESOURCE_PASSWORD`.
- Counters: Integrity-protected, monotonically increasing counters used to enforce resource state changes, as specified for `RESOURCE_COUNTER`.
- Timers: Timeouts or validity dates evaluated by the CSP, as specified for `RESOURCE_TIMER`.

For further information, see section 5.2.1.1, Resource Types.

3.1.2 Destroy Resources

The CSP shall support the destruction of CSP Resources using the `CSPDestroyResource` command. This process involves securely erasing memory allocated for these resources.

3.1.3 Load Resource for Personalization

The CSP shall support secure loading of CSP Resource Values, such as symmetric, private or public key values, the content of certificates, and specific PINs and passphrases. The process covers the initial loading of these resource values into CSP Resources using the `CSPSetValue` command. During this process, sensitive data, such as symmetric keys, private keys, PINs, and passphrases shall be encrypted.

After a new value is assigned to a resource, the CSP shall transition the resource to `STATE_OPERATIONAL` and shall reset the `tryCounter`, `counterValue`, and `timerValue`, if applicable to the resource.

To ensure interoperability across different vendors, the CSP shall adhere to the data format for each of the *Key Types* and *Certificate Types* specified in this document, guaranteeing that CSP implementations from different vendors support the same data format for resource importation.

The `CSPSetValue` command for loading resource values is available only to CSP Admins. Client Applications may perform *Import and Export Public Keys*, *Import Certificates*, and *Updating Passwords* operations on resources, provided the appropriate `ACCESS_SETUP` right is configured for the resource.

For further information, see section 6.7.1.1, *Key Types*, and section 6.8.1.1, *Certificate Types*.

3.1.4 Configure Resources

The CSP shall support the configuration of CSP Resources using the `CSPConfigureResource` command. This includes configuring resource parameters such as `accessControl`, `usageType`, `algorithms`, `counters`, `timers`, and `resourceEvents` for each resource.

There is a direct relationship between resource configuration and service configuration. Each resource is designed for a singular purpose, specified by its `usageType`. For instance, a resource configured for `USAGE_ATTESTATION` is exclusively utilized as an attestation key within the CSP's attestation service and cannot be used as a signature key within the CSP's signature service. Consequently, configuring a resource is effectively synonymous with configuring a service provided by the CSP.

The `CSPConfigureResource` command is available only to CSP Admins. Client Applications cannot change this resource configuration.

Note: Parameters that define the resource itself, such as resource type and sizes, cannot be changed.

For further information, see section 5.2.2, *Resource Configuration*.

3.1.5 Resource Identifiers

The CSP's service operations shall determine their cryptographic and security-related configurations from CSP Resource Identifiers. These `resourceIds` refer to specific resource configurations set by the CSP Admin and managed by the CSP. Client Applications cannot configure cryptographic algorithm or key parameters, but they can invoke the CSP's service operation using these `resourceIds`.

A `resourceId` is a unique identifier of type signed short. Within a CSP Instance, each `resourceId` is unique across all resource types, meaning that different resources, regardless of their type, cannot share the same `resourceId`. The CSP Admin selects and assigns these `resourceIds` when creating resources using the `CSPResource` structure. If an existing `resourceId` is used when creating a new resource, the CSP Instance shall return error `0x2002`, and the resource shall not be created.

The `resourceId` chosen by the CSP Admin must match those referenced by the Client Application to ensure seamless integration and access.

3.1.6 Streaming Resources for Offloading

The CSP may support resource streaming through the *Offloading Module*; the choice shall be subject to *Modularity*. Offloading refers to the encrypted export and import of CSP Resources using the `offloading.manage` operation.

Client Applications may use offloading to store resources externally to save memory. The resources shall be encrypted and decrypted using an encryption key configured as a CSP Resource with `USAGE_OFFLOADING` and a specific cipher algorithm.

By default, Client Applications shall not have permission to initiate offloading operations. The CSP Admin may configure the `ACCESS_MOVE` right for each resource to grant permissions for import and export.

Resources exported through the offloading functionality remain active and usable within the CSP. If this is not intended, Client Applications must explicitly clear the resource to prevent it from being used further.

The CSP Admin may configure a validity date for exported resources. This date shall be included in the exported data and checked by the CSP during import. This mechanism may be used to protect against the import of outdated (replayed) resources.

Note: This specification does not define the resource offloading format, so it is vendor-specific. Therefore, only the original CSP Instance that exported the resources can re-import them. A future version of this specification may define a standard resource offloading format.

For further information, see section 6.13, Offloading Module.

3.1.7 Clear Resources

The CSP shall provide functionality to clear resources through the `resource.clear` operation and the `CSPClearResource` command. This functionality allows to remove a cryptographic value without the need to destroy the resource itself. Clearing a resource involves securely wiping the resource's value and resetting its state to `STATE_UNINITIALIZED`.

By default, Client Applications shall not have permission to initiate resource clearing. The CSP Admin may configure the `ACCESS_CLEAR` right for a resource to grant Client Applications permission to clear it.

The CSP Admin should grant the `ACCESS_CLEAR` right only to resources where key clearing is appropriate. The Client Application is then responsible for ensuring that the key value is reinitialized through *Key Management* or *Certificate Management* before the next use.

Note: This operation is applicable to all resource types, including certificates, passwords, counters, timers, and both transient and non-transient keys.

For further information, see section 5.3.1.2, Access Rights.

3.2 Access Control

The CSP shall ensure that cryptographic services can only be invoked by registered and authorized Client Applications with the appropriate access rights granted for the CSP Resource being used.

The CSP shall enforce the following access control mechanisms:

- Only CSP Clients registered to a CSP Instance through the `CSPRegisterClient` command are permitted to use its cryptographic services. *Client Authentication*, optional for on-card Clients and mandatory for off-card Clients, may further restrict this access.
- The CSP shall verify the resource's Access Control Rules (ACR) bitmask to ensure the invoking CSP Client has sufficient access for the requested operation and resource, as configured by the CSP Admin using the `CSPAccessControl` structure.
- The CSP shall allow a requested operation only if the *Policies* configured for the resource are met. CSP Admins may define these conditional access rules for different *Policy Types* using the `CSPPolicy` structure. Such policies constrain resource usage based on the states of other resources. Example: A policy can enforce that a cipher operation is only allowed when a secure channel is authenticated.
- The CSP shall ensure that the resource's configured intended usage is appropriate for the requested operation. These *Usage Types*, configured by the CSP Admin using the `CSPUsageType` structure, shall restrict the use of keys, certificates and passwords to specific cryptographic services such as cipher, signature, secure messaging or key management.
- The CSP shall restrict the use of keys and certificates to exactly one algorithm specified by the CSP Admin using the `CSPAlgorithms` structure.
- The CSP shall ensure that the `resourceState` is appropriate for the requested operation.
- The CSP shall evaluate counters and timers configured for the resource, restricting access if their state transitions to `STATE_EXHAUSTED` or `STATE_EXPIRED`. This evaluation applies only to counters and timers supported by the platform.

For further information, see section 5.3, Security Module.

3.2.1 Client Application Registration

Only Client Applications registered to a CSP Instance shall be permitted to use its pre-configured cryptographic services. The CSP Admin shall be able to register Application Identifiers (AID) of Client Applications using the `CSPRegisterClient` command, thereby authorizing these Client Applications to access a specific CSP Instance.

The CSP Admin may configure additional checks available in the `CSPClientApplication` structure, such as ensuring the authenticity of the Client Application through Data Authentication Pattern (DAP) verification, verifying data integrity by checking the Load File Data Block Hash (LFDBH), or enhancing authentication by verifying the AID of the Security Domain (SD) associated with the Client Application.

The CSP shall perform these checks when a Client Application attempts to *Retrieve the CSP Instance* using the CSP's Global Service. In this process, the CSP shall compare the AID of the calling Client Application and, if configured, the AID of its associated Security Domain with the AIDs registered by the CSP Admin, and shall perform any additional checks as configured.

Note: Client Application may be (re-)installed after registration. Deleted Client Applications are not automatically unregistered and must be manually unregistered through `CSPUnregisterClient`.

For further information, see section 4.1, Architecture and section 8.2, Retrieve CSP Instance.

3.2.2 Enforce Client Authentication

The CSP Admin may configure *Client Authentication* to restrict access to CSP services. If configured, the CSP shall deny access to its services for Client Applications and off-card Clients that are not authenticated according to this configuration.

The CSP Admin may configure different authentication keys per Client Application, as well as additional authentication keys to support multiple off-card Clients, using the `CSPRegisterClient` command. However, if this is not sufficient, a Client Application may use the *Secure Channel Module* to establish additional secure channels.

If *Client Authentication* is not configured by the CSP Admin, Client Applications can still use CSP services, as the checks performed to *Retrieve the CSP Instance* are typically sufficient for Client Applications running on the same platform as the CSP Application.

However, off-card Clients requesting CSP services via the `CSPClientCommand` of the CSP Protocol through the APDU Interface of the CSP Application shall always be denied access if *Client Authentication* is not configured by the CSP Admin.

For further information, see section 4.2, Role Model, and section 5.3.1.1, Client Authentication.

3.2.3 Resource Ownership

The CSP Admin shall be able to configure an owner to each resource and configure Access Control Rules (ACR) that define the actions the owner is permitted to perform on the resource.

The CSP Admin configures the owner using the `CSPAccessControl` structure. This owner is a Client Application, identified by its Application Identifier (AID).

For further information, see section 5.3.1.3, ACR Bit Positions.

3.2.4 Access Control Rules

The CSP shall verify whether an entity is authorized to invoke the service operations of a specific CSP Instance. If the Access Control Rule (ACR) configuration for the resources being used does not permit the invoking entity to access those resources, the CSP shall deny the service. The CSP shall evaluate the ACRs for every operation that involves key, certificate or password resources.

The CSP Admin may configure `accessControlRules` for a CSP Resource using the bit positions defined in the `CSPAccessControlRules` structure. This bitmask combines user and group within a single bitmask. As a result, it limits the ability to assign distinct rules for individual applications or groups. Instead, access control rules can only be configured for the following roles:

- OWNER: A Client Application, identified by the AID, configured as the owner of the resource, and/or
- ANY: All Client Applications registered to this CSP Instance, including the owner and others, and/or
- ADMIN: The CSP Admin, who, in addition to configuring the CSP Instance, may utilize CSP services, such as initializing key resources.
- CLIENT: An off-card Client that directly utilizes CSP services through the CSP's APDU interface.

For each role, the CSP Admin may configure the following access rights:

- ACCESS_SETUP: Modify resource values (e.g., change passwords, generate keys, import certificates).
- ACCESS_USE: Use key, certificate or password resources for cryptographic operations (e.g., cipher service).
- ACCESS_CLEAR: Erase the value of a resource and reset it to STATE_UNINITIALIZED (e.g., clear before re-deriving).

- ACCESS_MOVE: Import or export resources (e.g., offloading for backup purposes).

Note 1: The bitmask allows for combining different access rights by using their respective bit values. For example, ACCESS_SETUP and ACCESS_MOVE can be combined while excluding ACCESS_USE. Refer to Table 3-1 for an overview of possible combinations. The full definition of the bit positions is provided in Table 5-26 *ACR Bit Positions*.

Note 2: ACRs may be further restricted using *Policies*.

For further information, see section 5.3.1.2, Access Rights, and section 5.3.1.3, ACR Bit Positions.

Table 3-1: Access Control Rules Bit Positions

Role	USE	SETUP	CLEAR	MOVE
ANY	1	2	3	4
OWNER	5	6	7	8
ADMIN	9	10	11	12
CLIENT	13	14	15	16

3.2.5 Usage Concept

The CSP shall restrict the use of CSP Resources to exactly one specific purpose. The CSP Admin may configure this usageType using CSPAlgorithms structure.

Depending on the supported *Modules*, the CSP offers the following *Usage Types*:

- USAGE_CIPHER: The resource can only be used for cipher operations.
- USAGE_SIGNATURE: The resource can only be used for signature operations.
- USAGE_TRANSFORM: The resource can only be used as encryption key for re-encryption by the CSP.
- USAGE_SECCHANNEL: The resource can only be used for secure messaging authentication.
- USAGE_CONFIDENTIAL: The resource can only be used for confidential data transfer.
- USAGE_ATTESTATION: The resource can only be used as attestation key.
- USAGE_KEY: The resource can only be used as source for key derivation or key agreement operations.
- USAGE_PASSWORD: The resource can only be used for password verification.
- USAGE_AUDIT: The resource can only be used to sign audit log messages.
- USAGE_OFFLOADING: The resource can only be used as encryption key for resource import and export.

For further information, see section 5.2.1.2, Usage Types.

3.2.6 Policies

The CSP shall verify security dependencies between resources by evaluating policies. A policy is a conditional access rule that restricts a CSP service operation to resources that depend on each other.

The CSP Admin may configure policies for resources using the CSPPolicy structure. The CSP shall evaluate these policies in each operation that utilizes the resource to which the policy is configured. If a policy is not fulfilled, the CSP shall reject the operation, even if it would otherwise be allowed according to the *Access Control Rules* (ACR) configured for the resource. Conversely, if the ACR does not permit the operation, it is still rejected, even if the policy is fulfilled.

The CSP may support the following *Policy Types*; the exact list shall be subject to *Modularity*.

- Key Pair: Execute operations requiring public-private key pairs only if the provided public and private keys are cryptographically linked, as specified by POLICY_KEYPAIR.
- Decrypt if secure channel established: Allow decryption operations only if an associated secure channel is established, as specified for POLICY_SECCHANNEL_ESTABLISHED.
- Sign if secure channel established: Allow the creation of a signature only if an associated secure channel is established, as specified for POLICY_SECCHANNEL_ESTABLISHED.
- Signing PIN: Allow the creation of a signature only if an associated password is authenticated, as specified for POLICY_PASSWORD.
- Decryption PIN: Allow decryption operations only if an associated password is authenticated, as specified for POLICY_PASSWORD.
- Two-Person Verification: Combine multiple policies of type POLICY_PASSWORD, mandating that two or more passwords must be successfully verified before permitting the requested action.
- Unblock PUK: Allow unblocking a password and resetting the (re)try counter only if an associated password is authenticated, as specified for POLICY_UNBLOCK_PASSWORD.
- Pre-block CAN: Allow the final verification attempt of a password with tryCounter=1 only if an associated password is authenticated, as specified for POLICY_PRE_BLOCKED.
- Decrypt if TA access flag is present: Allow decryption operations only if a specific access flag has been provided by an off-card entity through Terminal Authentication 2 (TA2), as specified for POLICY_TA2_ACCESS_FLAG.
- Dedicated Attestation Key: Restrict the use of an attestation key to a specific associated resource that shall be attested, as specified for POLICY_ASSOCIATION.
- Dedicated Transformation: Restrict the use of a decryption key used for *Encryption Transformations* to a specific associated encryption key, as specified for POLICY_ASSOCIATION.
- Proof of Association: Allow the computation of a signature using the SIG_POA_FIAT_SHAMIR algorithm only if the two involved private keys are associated with each other, as specified for POLICY_ASSOCIATION.

Note: The use of policies is optional, meaning not every resource must be equipped with a policy. Whether and how policies are configured in detail to further restrict access to resources depends on the specific use cases being implemented.

Policy example scenarios:

POLICY_SECCHANNEL_ESTABLISHED: Client Applications may use the CSP's cipher service to encrypt data with a specific encryption key resource. The CSP Admin may configure a POLICY_SECCHANNEL_ESTABLISHED, constraining the use of this encryption key resource to an Authentication Terminal Root certificate used for SEC_TA_AT_ROOT. The CSP evaluates this policy when initializing the cipher service with the specified encryption key. If the secure channel associated with the TA certificate is not authenticated, the CSP will deny the initialization of the cipher service.

POLICY_UNBLOCK_PASSWORD: Client Applications may invoke the `pwd.resetAndUnblock` operation for *Unblocking Passwords*. In this operation, the Client Application provides the `resourceId` of the PIN to be unblocked and the `resourceId` of the PUK to be verified. If the CSP Admin has configured a policy of type POLICY_UNBLOCK_PASSWORD to the PIN resource, the CSP not only verifies that the provided PUK is authenticated but also that it is associated with the specified PIN to be unblocked.

For further information, see section 6.15.1.2, Policy Types, and section 6.15.1.3, Policy Combinations.

3.3 Key Management

The CSP may provide key management through the *Key Module*; the choice shall be subject to *Modularity*. Key management operations are used to initialize key values and include key generation, key derivation and key agreement, as well as the direct import of public key values. Key generation is a non-deterministic process that relies on the CSP's entropy source to produce new key values. Key derivation is a deterministic process that creates key values from existing data and key agreement involves computing shared secrets between the CSP and a remote party.

Client Applications shall not be able to select key parameters or algorithms. The CSP Admin shall configure these through the CSPKey and CSPAlgorithms structures, leveraging the CSP's *Resource Management* capabilities to define:

- The resource type, key type, size and/or curve (e.g., RESOURCE_KEY, KEY_SHARED_SECRET, 256 bit).
- The key derivation and agreement algorithms (e.g., KDF_AES_CMAC).

Client Applications may pass the resourceId of a key resource to the *Key Operations* provided by the CSP.

Only resources with USAGE_KEY in STATE_OPERATIONAL shall be accepted as source keys for the CSP's key management operations. Resources used as destination keys (e.g., to store the result of a key generation, key derivation or key agreement process) must be in STATE_UNINITIALIZED. The usage type of these destination keys may differ from USAGE_KEY, allowing them to be used in other CSP services, such as signing key for *Creating Log Messages* or *Config Attestation*.

A resource configured for USAGE_KEY with a specific key derivation algorithm, such as KDF_AES_CMAC, can only be used as a source key for this specific key derivation algorithm and not for any other cryptographic operation or algorithm. However, the CSP shall not enforce usage type restrictions on destination resources. This means that a second resource configured for USAGE_CIPHER can be used as the destination for the key derivation, since usage type restrictions apply only when a resource is used as a source in an operation.

Example: Consider an AES key that is derived through key derivation and later used for encryption. In this case, the key must be assigned USAGE_CIPHER to be used as a source for encryption. However, the same resource can first serve as a destination (output) for key derivation and later as a source (input) for encryption.

After each key management operation, the CSP shall transition the destination resource from STATE_UNINITIALIZED to STATE_OPERATIONAL. If counters or timers are configured for the destination resource, they shall be reset according to their respective configurations.

By default, Client Applications do not have permission to initialize key values and are therefore restricted from performing key management operations. The CSP Admin may configure the ACCESS_SETUP right for a destination resource to grant permissions to initialize its key value. To enable repeated key generation for the same resource (e.g., often required for key agreement), the ACCESS_CLEAR right must be granted. This allows Client Applications to *Clear Resources* before repeating key management operations.

For further information, see section 2.2, Requirements, and section 6.7, Key Module.

3.3.1 Key Generation

The KeyModule shall support the generation of cryptographic key values using a Secure Random Number Generator (SRNG) as specified in Table 6-62 *Key Generation Processes*.

The CSP Admin may initiate key generation using the CSPGenerateKey and CSPComputePublicKey commands. Client Applications may trigger key generation through the key.generate, key.computePublicKey, and key.generateKeyPair operations, provided that the destination resources are in STATE_UNINITIALIZED and the Client Application has the ACCESS_SETUP rights for these resources.

The KeyModule may support the following *Key Types*; the exact list shall be subject to *Modularity*.

- Keys for the Advanced Encryption Standard (AES) block cipher, as specified for KEY_AES.
- Key pairs based on Elliptic Curve Cryptography (ECC), as specified for KEY_ECC_PRIVATE and KEY_ECC_PUBLIC.
- Key pairs based on the Rivest-Shamir-Adleman (RSA), as specified for KEY_RSA_PRIVATE and KEY_RSA_PUBLIC.
- Keys for keyed hash functions (e.g., HMAC) or key derivation, as specified for KEY_HMAC.
- Keys used as secrets for key agreement and key derivation, as specified for KEY_DERIVED_SECRET, KEY_MASTER_SECRET, and KEY_SHARED_SECRET.

For repeated key generations for the same resource, the ACCESS_CLEAR right must be granted, allowing Client Applications to uninitialize the resource using the resource.clear operation.

Note: This document distinguishes between key generation and key derivation, where key generation is a random process that does not include Key Derivation Functions (KDFs).

For further information, see section 6.7.1.5, Key Generation Process, section 6.7.1.2, Key Sizes, and section 6.7.1.3, ECC Curves.

3.3.2 Key Derivation

The KeyModule shall support key derivation using a deterministic process that consistently produces the same derived key values when identical input data is provided.

The CSP Admin may initiate key derivation using the CSPDeriveKey command. Client Applications may trigger key derivation through the key.derive operation, provided that the destination resource is in STATE_UNINITIALIZED and the Client Application has the ACCESS_SETUP right for this resource.

The KeyModule may support the following KDFs; the exact list shall be subject to *Modularity*.

- AES-CMAC KDF: Two-step AES-CMAC key derivation to create an AES key from a secret, as specified for KDF_AES_CMAC. The derived keys are typically used for generating message authentication codes for signatures after Elliptic Curve Diffie-Hellman (ECDH) key agreement.
- ECC KDF: Key pair derivation for ECC systems to create ECC keys from a secret, as specified for KDF_ECC. Technologies like FIDO U2F leverage this method to derive ECC keys from secure passwords, enhancing token security with ECC cryptography's strength through high-entropy password inputs.
- HKDF: HMAC key derivation to create an HMAC key from a secret, as specified for KDF_HKDF. The derived keys are then used to create HMAC signatures, commonly in Encrypt-then-Sign scenarios.
- PBKDF: Password-Based Key-Derivation Function (PBKDF) to create a secret from a password, as specified for KDF_PBKDF2. The derived secret can then be utilized in subsequent key derivation operations, e.g., to generate ECC key pairs from user-provided secrets.

The input parameters for key derivation include an input source and additional data. The input source is a CSP Resource, which may be the result of key generation, key derivation, key agreement, an imported resource value or a password.

The additional data has dual functionality: ensuring uniqueness through a salt or enabling context binding.

For KDF_HKDF and KDF_PBKDF2, additional data is used as a salt, typically a random value, to ensure that the derived key is unique. This enhances security and prevents attacks such as precomputed attacks or rainbow table attacks.

In KDF_AES_CMAC and KDF_HKDF, additional data can be used for application-specific or context-specific key derivation. This allows, for example, for key management without storing each derived key individually. Instead, fixed application attributes can be used as input to derive multiple keys from a single key resource. This approach is particularly useful in environments with limited memory or storage capacity.

For repeated derivations into the same destination resource, the ACCESS_CLEAR right must be granted, allowing Client Applications to uninitialize the resource using the resource.clear operation.

Note: These KDF algorithms produce deterministic outputs and are not suitable for generating random key material. For generating random keys, use the CSP's *Key Generation* operations.

For further information, see section 6.7.1.6, Key Derivation Algorithms, section 6.7.1.7, Key Derivation Combinations, and section 6.7.1.8, Key Derivation Additional Data.

3.3.3 Key Agreement

The KeyModule shall support key agreement to compute a shared secret between the CSP and an external entity.

Client Applications may trigger key agreement through the key.computeSharedSecret operation, provided that the shared secret resource is in STATE_UNINITIALIZED and the Client Application has the ACCESS_SETUP right for the secret.

The KeyModule may support the following agreement schemes; the exact list shall be subject to *Modularity*.

- ECKA-DH: Elliptic Curve Key Agreement based on Diffie-Hellman also known as Elliptic Curve Diffie-Hellman (ECDH), as specified for KAS_ECKA_DH. A typical application using ECKA-DH is the Mobile Driving Licence (MDL) to create a shared secret for subsequent signature operations.
- ECKA-EG: Elliptic Curve Key Agreement based on the ElGamal scheme, as specified for KAS_ECKA_EG. Its dual functionality of key agreement and digital signature makes ECKA-EG particularly useful in architectures that require both encryption and digital signatures.

The outcome of every key agreement process is a shared secret, which is managed within the CSP as a key resource of type KEY_SHARED_SECRET. This key can be utilized in key derivation operations for subsequent procedures.

For repeated derivations into the same shared secret resource, the ACCESS_CLEAR right must be granted, allowing Client Applications to uninitialize the resource using the resource.clear operation.

Note: The key management operations, including key agreement, defined in this document apply only to the CSP's key services provided for Client Applications. The CSP may perform key agreement internally beyond these definitions, for example, when establishing a secure messaging channel.

For further information, see section 6.7.1.9, Key Agreement Schemes, and section 6.7.1.10, Key Agreement Combinations.

3.3.3.1 ECKA-DH

The KeyModule may support Elliptic Curve Key Agreement based on Diffie-Hellman (ECKA-DH) via the KAS_ECKA_DH algorithm; the choice shall be subject to *Modularity*. ECKA-DH, also known as Elliptic Curve Diffie-Hellman (ECDH), may be used by Client Applications to establish custom secure messaging solutions. The outcome is a resource of type KEY_SHARED_SECRET, which must be configured within the CSP for use in subsequent key derivation operations.

In ECKA-DH both parties, the initiator and the responder, use their own private keys in conjunction with the public key of the other party to compute the shared secret. The necessary resource definitions are as follows:

- **ECC Key Pair:** The keys of this key pair, once generated or imported, are used as private and public source keys for ECKA-DH.
- **Shared Secret:** A resource representing the computed shared secret, which can then be used as input for further key derivation operations.

For further information, see section 6.7.1.9, Key Agreement Schemes.

3.3.3.2 ECKA-EG

The KeyModule may support Elliptic Curve Key Agreement based on ElGamal (ECKA-EG) via the KAS_ECKA_EG algorithm; the choice shall be subject to *Modularity*. This algorithm may be used by Client Applications to establish custom secure messaging solutions. The result is a resource of type KEY_SHARED_SECRET, which must be configured within the CSP for use in subsequent key derivation operations.

The ElGamal algorithm requires the initiator and responder to use different input parameters for the key agreement. The CSP shall support ECKA-EG for both roles – the sender (for encryption) and the recipient (for decryption) – outlined as follows:

- **CSP is initiator:** To initiate the key agreement, the Client Application generates a temporary ECC key pair for each key agreement process. Each ephemeral key pair is used only for that specific transaction. The public key is sent to the responder. As initiator, the CSP uses its own static ECC public key in the calculation. Thus, the newly generated ephemeral ECC private key and the CSP's static ECC public key are used as input parameters for the key agreement.
- **CSP is responder:** If the key agreement is initiated by an external party, the remote party generates an ephemeral key pair for each key agreement process. Since the remote public key changes with each transaction, the Client Application has to import the remote public key into a CSP key resource. Finally, the CSP's static ECC private key and the ephemeral remote public key are used as input parameters for the key agreement.

Key resource definitions required for this process are as follows:

- **Static ECC Key Pair:** The static key pair for initializing ECKA-EG agreements.
- **Ephemeral ECC Key Pair:** This key pair is freshly generated for each session when the CSP is the initiator, thus enhancing security through forward secrecy.
- **Remote ECC Public Key:** A resource used by Client Applications to import the initiator's ephemeral public key.
- **Shared Secret:** A resource representing the computed shared secret, which can then be used as input for further key derivation operations.

For further information, see section 6.7.1.9, Key Agreement Schemes.

3.3.4 Import and Export Public Keys

The KeyModule shall support the import and export of public key values, allowing Client Applications to retrieve and set public key values in the form of unencrypted byte arrays.

The CSP Admin may import key values using CSPSetValue command. Client Applications may import or export public keys using the key.manage operation. Public keys can only be imported to resources in STATE_UNINITIALIZED with ACCESS_SETUP right.

To ensure interoperability, the CSP shall use the data format specified in Table 6-58 *Key Types*.

For repeated imports into the same resource, the ACCESS_CLEAR right must be granted, allowing Client Applications to uninitialize the resource using the resource.clear operation.

For further information, see section 3.1.3, Load Resources, and section 6.7.1.1, Key Types.

3.3.5 Transient Keys

The CSP Admin shall have the ability to mark a key as transient when creating new resources of type `RESOURCE_KEY` using the `CSPKey` structure. For these transient resources, the key value, along with all dynamic parameters, such as resource state, incremented counter values and computed validity dates, shall be stored in transient memory (`CLEAR_ON_RESET`).

Client Applications may reset all transient keys of a CSP Instance through the `resource.clearTransient` operation, provided the `ACCESS_USE` right is granted for the resource. This reset securely erases the key value, reverts the resource to `STATE_UNINITIALIZED` and resets all dynamic parameters to their initial values.

Note: Since the CSP Application cannot respond to a `CLEAR_ON_DESELECT` event, the Client Application must monitor this `javacard.framework.JCSystem` event ([JCAP]) and invoke `resource.clearTransient` if required by the use case.

3.4 Certificate Management

The CSP may provide certificate management through the *Certificate Module*; the choice shall be subject to *Modularity*. Certificate management operations are used to import certificates and to extract public keys or other certificate information.

Client Applications shall not be able to select the parameters or security function of a certificate. The CSP Admin shall configure these through the CSPCertificate and CSPAlgorithms structures, leveraging the CSP's *Resource Management* capabilities to define:

- The resource type and certificate type (e.g., RESOURCE_CERTIFICATE, CERT_X509).
- The security function of the certificate (e.g., SEC_TA_TERMINAL).

Client Applications may pass the resourceId of a certificate resource to the *Certificate Operations* provided by the CSP. Certificates may also be used in other CSP Services, such as USAGE_SECCHANNEL for secure channel authentication, provided they are in STATE_OPERATIONAL.

The CertificateModule may support the following *Certificate Types*; the exact list shall be subject to *Modularity*.

- Card Verifiable Certificates (CVC), as specified for CERT_CVC.
- X.509 certificates, as specified for CERT_X509.

By default, Client Applications do not have permission to import certificates. The CSP Admin may configure the ACCESS_SETUP right for a certificate resource to grant permissions for importing the certificate. To enable repeated imports into the same resource, the ACCESS_CLEAR right must be granted. This allows Client Applications to *Clear Resources* before importing new certificates.

For further information, see section 2.2, Requirements, and section 6.8, Certificate Module.

3.4.1 Import Certificates

The CertificateModule shall support the import of certificates into CSP Resources.

The CSP Admin may import certificates using CSPSetValue command and Client Applications may import certificates using the cert.manage operation with MANAGE_MODE_CERTIFICATE_IMPORT, provided that the destination resource is in STATE_UNINITIALIZED and has the ACCESS_SETUP right.

Each certificate import transitions the destination resource to STATE_OPERATIONAL.

To ensure interoperability, the CSP shall use the data format specified in Table 6-75 *Certificate Types*.

The CertificateModule may support the following *Policy Types* to restrict certificate import; the exact list shall be subject to *Modularity*.

- POLICY_SECCHANNEL_ESTABLISHED: Importing a certificate is only allowed if an associated secure channel is fully established.
- POLICY_PASSWORD: Importing a certificate is only allowed if an associated password is authenticated.

For repeated imports into the same resource, the ACCESS_CLEAR right must be granted, allowing Client Applications to uninitialize the resource using the resource.clear operation.

For further information, see section 3.4.6.3, Certificate Revocation, section 3.4.6.2, Certificate Overlap, and section 6.8.1.1, Certificate Types.

3.4.2 Export Certificates

The CertificateModule shall support the export and parsing of certificates, enabling Client Applications to manage and retrieve certificate content.

Client Applications may export certificates using the `cert.manage` operation with `MANAGE_MODE_CERTIFICATE_EXPORT` or may extract only the public key data using `cert.extractPublicKey`.

Only certificates in `STATE_OPERATIONAL` can be exported.

For further information, see section 6.8.1.1, Certificate Types.

3.4.3 Certificate Trust Chain Verification

For `PROTOCOL_EAC_ID`, `PROTOCOL_EAC_MRTD`, and `PROTOCOL_PACE_CAM`, the CSP shall verify the trust chain for Terminal Authentication (TA) certificates received during secure channel authentication. This verification is performed against a use-case-specific trust anchor certificate stored within the CSP Instance.

Only certificates in `STATE_OPERATIONAL` shall be accepted as TA root certificates for this trust chain verification, configured via `SEC_TA_AT_ROOT` and `SEC_TA_IS_ROOT`.

3.4.4 Certificate Rollover

The CSP may offer support for certificate rollover of Terminal Authentication (TA) root certificates, as specified for `SEC_TA_AT_LINKED` and `SEC_TA_IS_LINKED`; the choice shall be subject to *Modularity*. The CSP replaces TA root certificates with linked TA certificates during Extended Access Control (EAC) protocol execution through the `sc.processSecurity` operation using `PROTOCOL_EAC_ID`, `PROTOCOL_EAC_MRTD`, and `PROTOCOL_PACE_CAM`.

Client Applications may implement custom certificate rollover using the `cert.manage` operation to manually replace certificates, provided the CSP Admin grants both `ACCESS_CLEAR` and `ACCESS_SETUP` rights for the certificate.

For further information, see section 3.4.6.3, Certificate Revocation, section 3.7.2.2, EAC for eID, and section 3.7.2.3, EAC for MRTD.

3.4.5 Extract Certificate Data

The CertificateModule shall provide functionality to extract the public key from a certificate and store it in a public key resource using the `cert.extractPublicKey` operation for further use with other CSP services. The key resource is initialized with the extracted data, requiring that the public key resource be in `STATE_UNINITIALIZED` and have the `ACCESS_SETUP` right.

Additionally, the CertificateModule shall provide the following information extracted from certificates:

- Key size, via the `cert.getPublicKeySize` operation.
- ECC curve, via the `cert.getPublicKeyCurve` operation.
- Validity date, via the `cert.getValidityDate` operation.

3.4.6 Limitations of Certificate Handling

3.4.6.1 Certificate Signing

The CSP does not support certificate signing for the creation of new certificates. Instead, certificates must be created by an external entity, which operates outside the CSP's scope. Once created, the certificates can be imported into the CSP using the CSP's *Import Certificates* functionality.

For further information, see section 2.4.3, Interaction with Certificate Authorities.

3.4.6.2 Certificate Overlap

The certificate import operation provided by the CSP allows importing certificates only into resources that are in STATE_UNINITIALIZED. Therefore, before importing a new certificate, the resource containing the old certificate must be cleared and securely erased to return it to STATE_UNINITIALIZED.

To manage overlapping validity periods, where both the old and new certificates are valid simultaneously, the CSP Admin should create two separate certificate resources: one for the old certificate and another for the new certificate. Client Applications must manage both resources during the transition period until the old certificate can be completely replaced. Before removing the old certificate, Client Applications must ensure that it is no longer needed for verifying old signatures.

Note: The ACCESS_SETUP and ACCESS_CLEAR rights must be granted to both certificate resources, allowing the Client Application to clear and import certificates.

For further information, see section 2.4.3, Interaction with Certificate Authorities, and section 3.4.1, Import Certificates.

3.4.6.3 Certificate Revocation

This document does not cover certificate revocation or the verification of whether a certificate appears on a revocation list. These tasks are outside the scope of this specification and must be handled by vendor-specific CSP implementations, the Client Application, or the CSP Admin.

This means the CSP does not verify certificate revocation, such as checking any kind of Certificate Revocation List (CRL) or using an Online Certificate Status Protocol (OCSP). Instead, it is the responsibility of the CSP Admin to ensure that revoked certificates are replaced when necessary, possibly through an external application. Alternatively, the Client Application or a customized CSP implementation may perform revocation checks if required.

However, the CSP provides operations to *Import Certificates* and to perform *Certificate Rollover*, which can be utilized to replace compromised certificates.

For further information, see section 2.4.3, Interaction with Certificate Authorities, section 3.4.1, Import Certificates, and section 3.4.4, Certificate Rollover.

3.5 Password Management

The CSP may provide password management through the *Password Module*; the choice shall be subject to *Modularity*. The CSP treats passwords as byte arrays with a system-defined maximum size of 255 bytes. However, the actual password size is configurable through the parameters `minSize` and `maxSize`.

Passwords may be marked as `inTransport`, requiring them to be changed through the `transportUsageCounter` before they can be used for security-related operations. The CSP shall track password verification failures in persistent memory through the `tryCounter` and block further use of the password once the configured `tryLimit` is reached. Additionally, the CSP shall support Personal Unblocking Key (PUK) handling, allowing a password to be unblocked only after the associated PUK has been successfully verified.

Client Applications shall not be able to select password parameters or algorithms. The CSP Admin shall configure these through the `CSPPassword` and `CSPAlgorithms` structures, leveraging the CSP's *Resource Management* capabilities to define:

- The resource type and password parameters (e.g., `USAGE_PASSWORD`, `PWD_NUMERIC`, min/max size 6).
- The algorithm that supports password resources (e.g., `SEC_PACE_PIN` or `KDF_PBKDF2`).

Client Applications may pass the `resourceId` of a password resource to the *Password Operations* provided by the CSP.

Only resources with `USAGE_PASSWORD` in `STATE_OPERATIONAL` shall be accepted for Password Verification. Password management operations, such as updating passwords, are not restricted by a specific usage type and may differ from `USAGE_PASSWORD`, such as `USAGE_SECCHANNEL` for secure messaging with *PACE* and `USAGE_KEY` for password-based key derivation.

A password value can be initially set only for passwords in `STATE_UNINITIALIZED`, after which the password transitions to `STATE_OPERATIONAL`. Updating the value of a password that is in `STATE_OPERATIONAL` requires the password to be authenticated, e.g., by verifying the old password value or processing a *PACE* authentication.

By default, Client Applications do not have permission to set, change or unblock passwords. The CSP Admin may configure the `ACCESS_SETUP` right for a password resource to grant permissions for these password management activities.

Note: The CSP does not perform type checking or apply any kind of heuristics to assess password entropy.

For further information, see section 2.2, Requirements, and section 6.9, Password Module.

3.5.1 Password Verification

The `PasswordModule` shall support password verification by securely comparing a provided input with the password value stored within the CSP.

Client Applications may trigger password verification through the `pwd.check` operation, provided the password is in `STATE_OPERATIONAL` with `USAGE_PASSWORD`.

Based on the outcome of this comparison, the CSP shall execute the following state changes:

- Successful password verifications shall mark the password as authenticated, allowing subsequent password updates through `pwd.update`, and affecting the result of `POLICY_UNBLOCK_PASSWORD`, `POLICY_PASSWORD`, and `POLICY_PRE_BLOCKED` policies constrained by this password.
- Failed password verifications shall be tracked using the `tryCounter` and result in the password being set to `STATE_BLOCKED` if the remaining retry counter reaches zero. A `POLICY_PRE_BLOCKED` may be used to prevent the password from being immediately blocked by requiring another password (e.g., a CAN) to be authenticated before performing the last possible password verification.

The CSP shall also mark passwords as authenticated when they are successfully verified through a *PACE* authentication using the `sc.processSecurity` operation and shall manage the `tryCounter` and blocking behavior of passwords during *PACE* authentication.

Note: Failed password attempts may be logged through *Secure Auditing*.

For further information, see section 6.9.4, Password Lifecycle.

3.5.2 Retry Counter with Maximum Try Limit

The CSP shall support enforcing a maximum (re)try limit for incorrect password attempts. This `tryLimit` may be configured by the CSP Admin. Once configured, the CSP shall maintain a `tryCounter`, decrementing it with each incorrect attempt, such as failed password verifications using the `pwd.check` operation and failed *PACE* authentications within the `sc.processSecurity` operation. The CSP shall prevent timing attacks and race conditions, e.g., by decrementing the retry counter at the start of password verification and incrementing it only upon successful verification.

When the `tryCounter` reaches zero, indicating that the maximum number of allowed attempts has been exceeded, the CSP shall set the password to `STATE_BLOCKED`. A blocked password shall be denied for any further use, including password verifications, *PACE* authentications, *PACE-CAM* authentications, and password-based key derivation.

When the `tryCounter` equals one and a `POLICY_PRE_BLOCKED` is configured for the password, a constraining resource (e.g., a `CAN`) must be authenticated before the password can be verified again through `pwd.check` or `sc.processSecurity`. The `tryCounter` shall be reset to its initially configured `tryLimit` when the `pwd.resetAndUnblock` operation is invoked, as well as after each successful password verification via the `pwd.check` operation or *PACE* authentications. It shall also be reset when a new password value is set, such as when set by the CSP Admin using the `CSPSetValue` command or by the Client Application using the `pwd.update` operation.

Note: The `tryCounter` is dedicated to password resources and is not available for keys or certificates. It is not related to other *Counter Types* provided by the `CounterModule`.

For further information, see section 3.2, Access Control, and section 3.5.3, Unblocking Password.

3.5.3 Unblocking Passwords

The `PasswordModule` shall provide functionality for unblocking passwords, transitioning them from `STATE_BLOCKED` to `STATE_OPERATIONAL` and resetting the `tryCounter` to its initially configured `tryLimit`.

Client Applications may unblock a password using the `pwd.resetAndUnblock` operation, providing the `ACCESS_SETUP` right is granted for the password. If the Client Application passes a new password value to this operation, the old password will be replaced with this new passphrase or PIN provided. If the Client Application passes a second `resourceId` referring to a Personal Unblocking Key (PUK) resource to the operation, the CSP shall verify this PUK and deny unblocking if the PUK is not authenticated.

The CSP Admin may enforce the use of a PUK by configuring a `POLICY_UNBLOCK_PASSWORD`. If the `CounterModule` is available, the PUK may also be restricted by a `usageCounter` with a maximum `counterLimit`; for example, to allow only 10 unblocking attempts.

In addition to the `pwd.resetAndUnblock` operation, the CSP Admin may reset a password using the `CSPClearResource` command, which sets the password to `STATE_UNINITIALIZED`, followed by the `CSPSetValue` command to set a new password value and transition it to `STATE_OPERATIONAL`.

Client Applications may also uninitialized a password using the `resource.clear` operation, provided the `ACCESS_CLEAR` right is granted for the password. To set a new password value via the `pwd.update` operation, Client Applications must have the `ACCESS_SETUP` right for the password.

For further information, see section 3.2.6, Policies, and section 3.5.2, Maximum Try Limit.

3.5.4 Passwords in Transport

The CSP Admin may set passwords to `inTransport` to indicate that the password has an initial value that must be changed before further use.

If `COUNT_TRANSPORT_USAGE` is available, the CSP Admin may configure a `transportUsageCounter` to limit the usage of passwords marked as `inTransport`. For example, setting this transport usage limit to one ensures that the password must be changed after the first login, effectively enabling a One-Time Password (OTP) mechanism through CSP functionality.

For more information, see section 3.9.1.6, Transport Counter.

3.5.5 Updating Passwords

The `PasswordModule` shall support the initial setting of a password or changing an existing password, but shall restrict changes to authenticated passwords only.

The CSP Admin may import an initial passphrase or PIN using `CSPSetValue` command for passwords in `STATE_UNINITIALIZED`.

Client Applications may set an initial passphrase or PIN using the `pwd.update` operation, provided the password is in `STATE_UNINITIALIZED` and has the `ACCESS_SETUP` right.

Setting the initial password will transition the password to `STATE_OPERATIONAL`.

Client Applications may change an existing password value using the `pwd.update`, provided the password is in state `STATE_OPERATIONAL`, has the `ACCESS_SETUP` right and is validated as authenticated through a prior successful invocation of the `pwd.check` operation or *PACE* authentication.

If available, the CSP shall reset `tryCounter`, `counterValue`, and `timerValue` for the password during each password set or change.

If configurations for a `minSize` or `maxSize` of the password are available, the CSP shall verify the new password size against the minimum and maximum limits set by the CSP Admin and shall deny the request if the size falls outside these limits.

For further information, see section 6.9.4, Password Lifecycle.

3.6 Cipher and Signatures

The CSP may provide cipher and signature functionalities; the choice shall be subject to *Modularity*, including:

- Encryption and decryption through the *Cipher Module*.
- Transferring encryption from one key and/or algorithm to another through the *Transform Module*.
- Signature creation and verification through the *Signature Module*.

Client Applications shall not be able to select cipher and signature algorithms. The CSP Admin shall configure these through the CSPCipherAlgorithms and CSPSignatureAlgorithms structures, leveraging the CSP's *Resource Management* capabilities to define:

- The resource type, key type, size, and/or curve (e.g., RESOURCE_KEY, KEY_RSA_PRIVATE, 3072 bit).
- The padding algorithm and cipher algorithm (e.g., PAD_PKCS1_OAEP_SHA256, CIPHER_RSA) or
- The padding algorithm, signature algorithm and message digest algorithm (e.g., PAD_PKCS1_OAEP_SHA256, SIG_RSA, ALG_SHA_512)

Client Applications may pass the resourceIds referring to these configurations to cipher and signature services, with the following restrictions:

- Only resources with USAGE_CIPHER in STATE_OPERATIONAL shall be accepted for *Cipher Operations*.
- Only resources with USAGE_TRANSFORM in STATE_OPERATIONAL shall be accepted as decryption resources for *Transform Operations*.
- Only resources with USAGE_SIGNATURE in STATE_OPERATIONAL shall be accepted for *Signature Operations*.

For further information, see section 2.2, Requirements, section 6.1, Cipher Module, section 6.2, Signature Module, and section 6.3, Transform Module.

3.6.1 Padding Schemes

The CipherModule may support the following padding schemes; the exact list shall be subject to *Modularity*.

- Optimal Asymmetric Encryption Padding (OAEP) scheme, as specified for PAD_PKCS1_OAEP_SHA256 and related values. OAEP is typically used in standard scenarios.
- No Padding, applicable in specific use-cases, as specified for PAD_NOPAD.
- ISO9797 ALG3 M2, a padding method for block ciphers, as specified for PAD_ISO9797_1_M2_ALG3. This algorithm is available for backward compatibility.
- Padding according to PKCS#7, as specified for PAD_PKCS7. Typically used with block ciphers like AES in CBC mode. This algorithm is available for backward compatibility.
- PKCS#1 v1.5 scheme, as specified for PAD_PKCS1. This algorithm is considered legacy and is supported for backward compatibility.

Note: The same padding algorithms as supported by the CipherModule are also used by the TransformModule, the SignatureModule, the ConfidentialModule, the AttestationModule, and the AuditModule.

For further information, see section 6.1.1.2, Padding Algorithms.

3.6.2 Hash and Message Digest

The SignatureModule may support the following hash algorithms; the exact list shall be subject to *Modularity*.

- SHA2 Family: This includes secure hash algorithms SHA-256, SHA-384, and SHA-512, as specified for ALG_SHA_256 and following. These algorithms are used for computing a hash digest.
- SHA3 Family: Comprising secure hash algorithms SHA3-256, SHA3-384, and SHA3-512, these are based on the Keccak algorithm, as specified for ALG_SHA3_256 and following. They provide an alternative approach and can serve as a secure option in case vulnerabilities are discovered in the SHA-2 family.

The CSP shall not offer SHA-1 as an explicit interface toward Client Applications due to its lack of collision resistance. However, the use of SHA-1 is permitted internally for key derivation within PROTOCOL_PACE and PROTOCOL_PACE_CAM (both defined in Annex A.2.3.2 of [TR-03110-3]), since collision resistance is not an issue in this key derivation.

Note: The same hash algorithms as supported by the SignatureModule are also used by the AuditModule and the AttestationModule.

For further information, see section 2.4.2, Legacy Cryptographic Algorithms, section 6.2.1.2, Message Digest Algorithms, and section 6.2.1.4, Signature Combinations.

3.6.3 Cipher

The CipherModule may support the following cipher algorithms; the exact list shall be subject to *Modularity*.

- AES: Advanced Encryption Standard with key sizes of 128 and 256 bits, as specified for CIPHER_AES_CBC and following.
- RSA: Rivest Shamir Adleman (RSA) encryption algorithm, as specified for CIPHER_RSA.

Note: The same cipher algorithms as supported by the CipherModule are also used by the TransformModule and the ConfidentialModule.

For further information, see section 6.1.1.3, Cipher Algorithms, and section 6.1.1.4, Cipher Combinations.

3.6.4 Cipher Block Modes

The CipherModule may support the following block cipher modes; the exact list shall be subject to *Modularity*.

- CBC: Cipher Block Chaining Mode, as specified for CIPHER_AES_CBC.
- CCM: Counter Counter/CBC Mode, as specified for CIPHER_AES_CCM.
- CFB, CFBx: Cipher Feedback Mode, as specified for CIPHER_AES_CFB.
- CTR: A simplification of OFB, Counter mode updates the input block as a counter, as specified for CIPHER_AES_CTR.
- GCM: Galois/Counter Mode, as specified for CIPHER_AES_GCM.
- XTS: XEX Tweakable Block Cipher with Ciphertext Stealing (XTS) Mode, as specified for CIPHER_AES_XTS.

Most cipher block modes require additional input data, such as a unique initialization vector (IV), as reusing an IV can compromise security. The Client Application shall ensure that IVs and other additional input data are handled properly, as specified in Table 6-5, *Cipher Initialization Data*.

Note: The same cipher block modes as supported by the CipherModule are also used by the TransformModule and the ConfidentialModule.

For further information, see section 6.1.1.3, Cipher Algorithms.

3.6.5 Signature Suites

The SignatureModule may support the following algorithms; the exact list shall be subject to *Modularity*.

- AES: Creation and verification of signatures using AES with block sizes of 128 bits with 16-byte MAC (GMAC) or 16-byte cipher-based MAC (CMAC), as specified for SIG_AES_CMAC128 and SIG_AES_MAC128.
- HMAC: Creation and verification of signatures using HMAC, as specified for SIG_HMAC.
- RSA: Creation and verification of signatures using RSA cipher with pads according to the PKCS#1-PSS scheme, as specified for SIG_RSA.
- ECDSA: Creation and verification of signatures using Elliptic Curve Digital Signature Algorithm (ECDSA) in X9.62 format and plain format, as specified for SIG_ECDSA and SIG_ECDSA_PLAIN.
- EC-Schnorr: Creation and verification of signatures using Elliptic Curve Based Schnorr Digital Signature Algorithm (ECSDSA), as specified for SIG_EC_SCHNORR.
- PoA Fiat-Shamir: Creation and verification of Proof of Association (PoA) signatures based on a Schnorr non-interactive zero-knowledge proof, as specified for SIG_POA_FIAT_SHAMIR.

Note: The same signature algorithms as supported by the SignatureModule are also used by the AuditModule and the AttestationModule.

For further information, see section 6.2.1.3, Signature Algorithms, and section 6.2.1.4, Signature Combinations.

3.6.6 Proof of Association

The CSP may support generating a cryptographic Proof of Association (PoA) between two public-private key pairs; the choice shall be subject to *Modularity*. In this PoA, the CSP derives an ephemeral key from the two associated private keys and uses it to generate a signature over both public keys and related metadata. This proof demonstrates that both private keys are managed by the same CSP Instance, without exposing them to external entities. This privacy-friendly approach allows, for example, Attestation Providers in the EUDI Wallet context to verify that attestations are issued to the correct user without exposing any user-specific information.

Client Applications may initiate PoA computations using the `sig.init` operation with both public-private key pairs configured for SIG_POA_FIAT_SHAMIR to create or verify PoA-based signatures. A complete PoA computation, where the CSP returns both the signature and the concatenated PoA data, can be triggered by Client Applications using `att.init` with ATTESTATION_DATA, as defined in the *Attestation Module*. Regardless of which API operation is used, the Client Application must the following resourceIds:

- First Private Key: The resourceId of the private part of the first key pair to be associated.
- First Public Key: The resourceId of the public part of the first key pair to be associated.
- Second Private Key: The resourceId of the private part of the second key pair to be associated.
- Second Public Key: The resourceId of the public part of the second key pair to be associated.

The first and second private key are used to derive a new key which signs the PoA data. This derived key is used internally for generating the PoA signature only and not managed as CSP Resource.

For further information, see section 2.2, Requirements, and section 6.6, Attestation Module.

Editor's Note: This PoA algorithm is illustrated in Figure 3-1 and described temporarily below for discussion. The proposal refers to [Fiat-Shamir] and [RFC 8235], but the [Verheul-PoA] introduces modifications and there is a patent: <https://patents.google.com/patent/NL1044483B1/>

Figure 3-1: Proof of Association Key Attestation

Algorithm 1 Proof of Association (PoA) generation

Input: optional verifier challenge C (byte array), two associated public keys $P_1 = p_1 \cdot G$, $P_2 = p_2 \cdot G$ with respective private keys p_1, p_2 .

Output: PoA = $\{P_1, P_2, C, (r, s)\}$.

```

1: If  $P_1 = P_2$  return error //  $P_1, P_2$  need to be different
2: Compute association key  $z = p_2 \cdot p_1^{-1} \bmod q$ . // note  $P_2 = z \cdot P_1$ 
3: Convert public keys  $P_1$  and  $P_2$  to byte arrays  $\bar{P}_1, \bar{P}_2$  respectively
4: Select random  $k \in \{1, \dots, q-1\}$ .
5: Compute  $P'_1 = k \cdot P_1 = (x, y)$  and convert to byte array  $\bar{P}'_1$ . // commitment
6: Compute byte array  $H(\bar{P}'_1 || \bar{P}_1 || \bar{P}_2 || C)$  and convert it to an integer  $r$ .
7: If  $r \bmod q = 0$  then go to Line 4.
8: Compute  $s = k + r \cdot z \bmod q$ .
9: If  $s = 0$  then go to Line 4.
10: Return PoA =  $\{P_1, P_2, C, (r, s)\}$ .

```

The following algorithm specifies the verification of a PoA.

Algorithm 2 Proof of Association (PoA) verification

Input: WTE, PoA = $\{P_1, P_2, C, (r, s)\}$

Output: Acceptance or rejection of the PoA.

```

1: Verify  $P_1 \neq P_2$  on failure Return Error //  $P_1, P_2$  need to be different
2: Verify the input, including that
    $r \in \{1, 2^{8 \cdot |q|} - 1\}$  and  $s \in \{1, q-1\}$ , on failure Return False.
3: Convert public keys  $P_1$  and  $P_2$  to byte arrays  $\bar{P}_1, \bar{P}_2$  respectively
4: Compute  $Q = s \cdot P_1 - r \cdot P_2$  if  $Q = \mathcal{O}$  Return False.
5: Convert  $Q$  to byte array  $\bar{Q}$ . // i.e. of size  $2 \cdot |p|$ 
6: Compute byte array  $H(\bar{Q} || \bar{P}_1 || \bar{P}_2 || C)$  and convert it to an integer  $v$ .
7: If  $v = r$  accept the PoA otherwise reject it.

```

Step-by-step explanation of this PoA algorithm:

- Assume an elliptic curve group G of order q with generator G , represented additively. Both public keys P_1 and P_2 belong to the same elliptic curve group. The base point G and parameters are shared for both keys.
- Line 1: Checks if the two public keys P_1 and P_2 are not the same. If they are equal, the algorithm returns an error because the two keys must be different to create a valid association.
- Line 2: The association key z is calculated by dividing the private key p_2 by the private key p_1 and taking the result modulo q . This operation binds the two keys together to ensure the association is established.
- Line 3: The public keys P_1 and P_2 are converted into byte arrays P_1^B and P_2^B , since cryptographic operations are typically performed using byte arrays rather than abstract key objects.
- Line 4: A random value k is selected from the set $\{1, 2, \dots, q-1\}$. This value will be used in the next step for computation.
- Line 5: A new point P_1' on the elliptic curve is computed by multiplying the random value k with the public key P_1 . This point is then converted into a byte array $P_1'^B$ for further cryptographic operations.
- Line 6: A hash value is created by concatenating the byte arrays $P_1'^B$, P_1^B , P_2^B , and the optional challenge C . This hash value is then converted into an integer r . This ensures that all inputs are processed and combined properly.
- Line 7: Checks if $r \bmod q = 0$. If this condition is true, it means r is invalid, and the algorithm returns to step 4 to select a new random value k .
- Line 8: The value s is calculated by adding k to the product of r and z , and the result is taken modulo q . This is part of the signature for proof of association.

- Line 9: Checks if $s = 0$. If so, the algorithm goes back to step 4 to choose a new random k , because $s = 0$ is invalid.
- Line 10: The Proof of Association (PoA) is returned, which includes the two public keys $P1$ and $P2$, the optional challenge C , and the tuple r and s . These values serve as proof that the two keys $P1$ and $P2$ are correctly associated.

3.6.7 Signatures with Counters & Timestamps

The CSP may support *Fields*; the exact list shall be subject to *Modularity*. Fields refer to specific system or resource parameters, such as a counter value, system time, or a resource state, which are added by the CSP to attestation data or log messages before they are signed. The SignatureModule may support the following fields:

- Version information, e.g., configured through the signature fields `FIELD_CSP_CONFIG_VERSION`, `FIELD_CSP_PROTOCOL_VERSION`, and `FIELD_CSP_ELF_VERSION`.
- The resource identifier, resource state or a public key value, through the fields `FIELD_RESOURCE_STATE` and `FIELD_PUBLIC_KEY`.
- Counters, such as manual or built-in counters, configured through the signature fields `FIELD_USAGE_COUNTER` and `FIELD_MANUAL_COUNTER`, allowing third parties to analyze gaps and discontinuities in a series of signed data messages (dependent on *Counter and Limits* functionality).
- Time-based information, configured through the signature fields `FIELD_SYSTEM_TIME`, `FIELD_TIME_SINCE_BOOT`, and `FIELD_REFERENCE_TIME`, ensuring the assessment and traceability of signed data messages (dependent on *Timer and Time Management* functionality).

The CSP-Admin may configure the *Fields* to be included through the `CSPAttestationAlgorithms` structure, applicable to *Config Attestation*, *Data Attestation*, *Proof of Possession Key Attestation*, *Generate Key Pair Attestation*, and *Creating Log Messages*. To ensure data integrity and validity, the Client Application shall not have the ability to manipulate these fields.

The CSP shall append these fields to attestation data or log messages when the Client Application invokes the `att.computeAttestation` or `audit.dequeueEvent` operations. The fields configured by the CSP-Admin are added to the data before signing, ensuring their inclusion in the generated signature.

Note: These fields are only supported by the `AttestationModule` and the `AuditModule`. The `SignatureModule` is limited to creating signatures and does not return any modified input data.

For further information, see section, 6.14.1.1, Fields.

3.6.8 Encryption Transformations

The `TransformModule` shall provide functionality to transfer the encryption from one key and algorithm to another in a single atomic step, thereby preventing Client Applications from accessing the unencrypted data.

The CSP Admin may configure two resources using the `CSPKey` and `CSPAlgorithms` structures, e.g.,

- Decryption resource (e.g., `RESOURCE_KEY`, `KEY_SHARED_SECRET`, 128 bit, `CIPHER_AES_CBC`).
- Encryption resource (e.g., `RESOURCE_KEY`, `KEY_SHARED_SECRET`, 256 bit, `CIPHER_AES_GCM`).

Client Applications may pass both `resourceIds` to the `transform.init` operation.

The CSP shall only accept decryption keys with `USAGE_TRANSFORM` or `USAGE_CONFIDENTIAL`, restricting Client Applications from using the standard *Cipher Operations* to decrypt the data to plaintext format.

For further information, see section 2.2, Requirements, and section 6.3, Transform Module.

3.7 Secure Messaging

The CSP may provide secure messaging functionalities through the *Secure Channel Module*; the choice shall be subject to *Modularity*. The CSP shall manage the complete protocol execution, including verifying the protocol sequence and generating the Application Protocol Data Unit (APDU) responses, while the Client Application acts only as a proxy, forwarding incoming and outgoing APDU commands.

Client Applications can select the protocol to use, such as `PROTOCOL_PACE`. However, Client Applications shall not be able to configure keys and their purpose. The CSP Admin shall configure these through the `CSPSecureChannelAlgorithms` and `CSPSecureChannelSettings` structures, leveraging the CSP's *Resource Management* capabilities to define:

- The resource type, key type, size and/or curve (e.g., `KEY_ECC_PRIVATE`, `CURVE_SEC_P256_R1`).
- The security usage of the resource (e.g., `SEC_PACE_PIN`, `SEC_CA2`).
- A secure channel timeout (if the `TimeModule` is available).

Client Applications may pass the `resourceIds` referring to these configurations to *Secure Channel Operations*. Only resources with `USAGE_SECCHANNEL` in `STATE_OPERATIONAL` shall be accepted for these operations.

For further information, see section 2.2, Requirements, and section 6.4, Secure Channel Module.

3.7.1 Secure Communication Flow

A secure channel, provided by the CSP, is designed to establish a secure communication link between the CSP Instance and an Off-card Entity (OCE), enabling an authorized Client Application to transmit and receive data through this channel, as illustrated in Figure 3-2.

To establish the secure channel, the Client Application forwards the APDU commands intended to initiate secure messaging from the OCE to the CSP using the `sc.processSecurity` operation. The CSP verifies each incoming command and generates an appropriate APDU response, which the Client Application forwards back to the OCE. This process is repeated until the secure channel is fully established.

The CSP offers two operations for encrypting data before transport and decrypting received data:

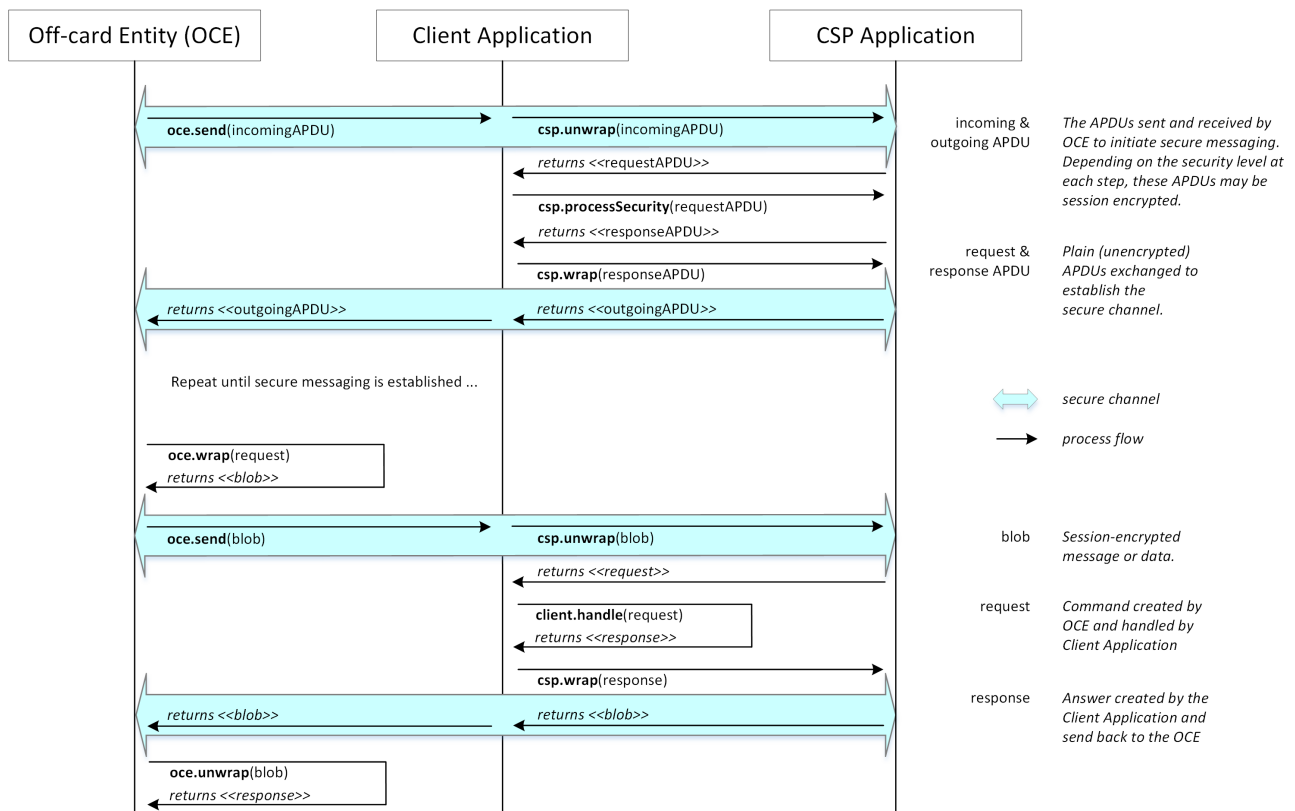
- `sc.wrap` operation: Adds session encryption to outgoing data.
- `sc.unwrap` operation: Removes session encryption from incoming data.

For each call to `sc.processSecurity`, the Client Application invokes `sc.unwrap` beforehand and `sc.wrap` afterward, as `sc.processSecurity` operates only on plain APDU commands. The `sc.wrap` and `sc.unwrap` operations add and remove session encryption based on the current `securityState` of the secure channel. If these methods are invoked without a secure channel being established, the input data is simply forwarded without modifications.

After the secure channel is established, Client Applications may utilize the `sc.wrap` and `sc.unwrap` operations to securely process incoming data and create responses for OCEs through the secure channel.

Note: These *Secure Channel Operations* may utilize keys, such as ephemeral session keys, which must not be configured as CSP Resources.

Figure 3-2: Secure Messaging Flow Chart



3.7.2 Built-In Secure Channel Protocols

The SecureChannelModule may support the following protocols; the exact list shall be subject to *Modularity*.

- **PACE:** Password Authenticated Connection Establishment (PACE) is a secure channel based on password authentication, as specified for `PROTOCOL_PACE`. PACE is typically used in combination with EAC to secure eID and passport applications. In the CSP context, a dedicated PACE protocol secures the connection between an off-card user application, such as a mobile app, and the Client Application.
- **EAC for eID:** Extended Access Control v2 (EAC v2) with Terminal Authentication (TA) and Chip Authentication (CA). Designed for eID scenarios, the CSP integrates PACE for PIN, PUK and CAN with TA2, CA2 and optionally CA3, and handles the complete protocol flow, as specified for `PROTOCOL_EAC_ID`.
- **EAC for MRTD:** Extended Access Control v1 (EAC v1). Designed for Machine Readable Travel Documents (MRTD) scenarios, the CSP integrates PACE for CAN and MRZ with CA1 and TA1, and handles the complete protocol flow, as specified for `PROTOCOL_EAC_MRTD`.
- **PACE-CAM:** PACE with Chip Authentication Mapping (CAM) and optionally TA1, as specified for `PROTOCOL_PACE_CAM`. This protocol is an efficient alternative to `PROTOCOL_EAC_MRTD`.
- **SCP03:** GlobalPlatform Secure Channel Protocol 03 (SCP03), specified for `PROTOCOL_SCP03`, is a more general-purpose protocol used beyond eID scenarios. It supports secure data exchange in applications such as mobile payments and secure identity verification.
- **SCP04:** GlobalPlatform Secure Channel Protocol 04 (SCP04), specified for `PROTOCOL_SCP04`, is a configurable version of SCP03.

Beyond those listed protocols, additional secure channel protocols can be constructed using the CSP's cipher and key management services, as outlined in Annex A, *Custom Secure Messaging*.

For further information, see section 6.4.1.1, Protocol Types.

3.7.2.1 PACE

Password Authenticated Connection Establishment (PACE) is a protocol that uses password-based authentication to mutually authenticate an off-card entity (i.e., the terminal) and the CSP. It relies on a shared secret, such as a PIN or password, to establish a secure channel for protecting subsequent communications. PACE enhances traditional password authentication by incorporating a password-authenticated Diffie-Hellman key agreement, allowing the CSP and the terminal to derive an ephemeral shared secret without transmitting the password itself.

The CSP may offer support for PACE in the following variants; the choice shall be subject to *Modularity*:

- Standalone secure channel protocol, as specified for PROTOCOL_PACE.
- Integrated in *EAC for eID*, as specified for PROTOCOL_EAC_ID.
- Integrated in *EAC for MRTD*, as specified for PROTOCOL_EAC_MRTD.
- Integrated in *PACE-CAM*, as specified for PROTOCOL_PACE_CAM.

The CSP shall use CSP Resources of type RESOURCE_PASSWORD to manage the PIN, PUK, CAN, MRZ, or custom PACE passwords. The CSP shall set these PACE resources to authenticated upon successful PACE authentications and shall reset them when the corresponding secure channel is reset. CSP Admins may set the password values of the PACE resources through the CSPSetValue command. Client Applications may change the password values using the *Updating Passwords* functionality, provided the respective password is authenticated and the ACCESS_SETUP right is granted.

The CSP Admin shall be able to configure a *Retry Counter with Maximum Try Limit* for the PACE resources, and the CSP shall handle a resource-individually tryCounter during PACE authentication. If the maximum tryLimit is exceeded, the CSP shall set the PACE resource to STATE_BLOCKED.

The CSP Admin may configure a POLICY_PRE_BLOCKED for PACE resources. If configured, the CSP shall enforce the constraining resource (e.g., the CAN) to be authenticated when the tryCounter of the policy owner (e.g., the PACE PIN) is already 1, before allowing the next password verification. This prevents the password from being immediately blocked.

Client Applications may use the *Unblocking Passwords* operation of the PasswordModule to unblock and reset a blocked PACE resource.

Note: The CSP inherently provides functionality for updating all kinds of PACE passwords, including PUK, CAN, and MRZ, as well as maintaining retry counters for each through the PasswordModule. However, this functionality is only available for Client Applications if CSP Admins grant the necessary access rights, which should not be activated for typical eID and MRTD use cases.

For further information, see section 3.5, Password Management.

3.7.2.2 EAC for eID

Extended Access Control (EAC) is a security protocol used to authenticate terminals and protect access to sensitive data on eIDs and MRTDs through Terminal Authentication (TA) and Chip Authentication (CA). TA is designed for the unilateral authentication of an off-card entity (i.e., the terminal) to the CSP. The terminal presents its digital certificate to the CSP, which verifies the certificate chain to ensure the terminal is authentic and authorized. CA is based on the Diffie-Hellman (DH) key agreement protocol, designed to provide unilateral authentication using an ephemeral-static key agreement mechanism. The 'static' aspect refers to the use of a pre-established CA key pair stored within the CSP, which is used to authenticate the CSP to the terminal.

The CSP may offer EAC integrated with PACE, tailored for eID scenarios as specified for PROTOCOL_EAC_ID. For this protocol, the CSP shall handle at least the following processes:

1. PACE: The CSP shall establish a secure channel between the terminal and the CSP (chip) using a password resource configured for SEC_PACE_PIN.
2. TA2: The CSP shall verify the terminal's certificate chain to ensure it is authorized to access CSP services using a trust anchor certificate resource configured for SEC_TA_AT_ROOT.
3. CA2: The CSP shall provide proof of its authenticity to the terminal, ensuring that it is genuine, and replace the PACE session keys with CA-specific session keys computed from a key resource configured for SEC_CA2.

The CSP may handle the following additional security processes during protocol execution; the exact list shall be subject to *Modularity*:

- SEC_PACE_PUK: Evaluate MSE:SET AT to use the Personal Unblocking Key (PUK) for PACE.
- SEC_PACE_CAN: Evaluate MSE:SET AT to use the Card Access Number (CAN) for PACE.
- SEC_CA2_PRIVILEGED: Support a chip-specific CA2 key for Privileged Terminals.
- SEC_CA3: Perform Chip Authentication Version 3 (CA3).
- SEC_CA3_PSA: Support a chip-specific CA3 key for Pseudonymous Secure Authentication (PSA).
- SEC_TA_DV: Return the terminal certificate received from the Document Verifier (DV).
- SEC_TA_TERMINAL: Return the terminal certificate received from individual terminals.
- SEC_TA_AT_LINKED: Perform *Certificate Rollover* of the Authentication Terminal (AT) root certificate.
- TIME_SYNC_FROM_TA: Update the CSP's referenceTime using the validity of received TA certificates.
- POLICY_TA2_ACCESS_FLAG: Evaluate access flags present in received TA certificates.
- POLICY_PRE_BLOCKED: Enforce that a CAN must be authenticated to restrict further *PACE* authentications when the tryCounter is already 1.

CSP Admins may import TA trust anchor certificates using the CSPSetValue command and may replace them using the CSPClearResource command followed by CSPSetValue. If the PasswordModule is supported, root certificates may also be replaced by Client Applications using the cert.manage operation, provided the ACCESS_CLEAR and ACCESS_SETUP rights are granted for the certificate.

The CSP shall ignore unknown extensions in TA certificates.

Note: CA1 and TA1 are not supported for this protocol type.

For further information, see sections 3.2.6, Policies, section 3.4.4, Certificate Rollover, and section 3.10.2.1 TA as Time Source.

3.7.2.3 EAC for MRTD

Machine Readable Travel Documents (MRTD), such as passports, rely on an older version of the Extended Access Control (EAC v1), which uses Chip Authentication (CA1) followed by Terminal Authentication (TA1).

The CSP may offer EAC integrated with PACE, tailored for MRTD as specified for PROTOCOL_EAC_MRTD. For this protocol, the CSP shall handle at least the following processes:

1. PACE: The CSP shall evaluate MSE:SET AT to establish a secure channel between the terminal and the CSP (chip) using password resources configured for SEC_PACE_CAN and SEC_PACE_MRZ.
2. CA1: The CSP shall provide proof of its authenticity to the terminal, ensuring that it is genuine, and replace the PACE session keys with CA-specific session keys computed from a key resource configured for SEC_CA1.

3. TA1: The CSP shall verify the terminal's certificate chain to ensure it is authorized to access CSP services using a trust anchor certificate resource configured for SEC_TA_IS_ROOT.

The CSP may handle the following additional security processes during protocol execution; the exact list shall be subject to *Modularity*:

- SEC_TA_DV: Return the terminal certificate received from the Document Verifier (DV).
- SEC_TA_TERMINAL: Return the terminal certificate received from individual terminals.
- SEC_TA_IS_LINKED: Perform *Certificate Rollover* of the Inspection System (IS) root certificate.
- TIME_SYNC_FROM_TA: Update the CSP's referenceTime using the validity of received TA certificates.
- POLICY_PRE_BLOCKED: Enforce that a CAN must be authenticated to restrict further *PACE* authentications when the tryCounter is already 1.

Same as for *EAC for eID*, the CSPSetValue, CSPClearResource commands, as well as the cert.manage operation, may be used to set or replace the TA trust anchor certificate.

The CSP shall ignore unknown extensions in TA certificates.

For further information, see section 3.10.2.1, TA Certificates as Time Source.

3.7.2.4 PACE-CAM

PACE with Chip Authentication Mapping (PACE-CAM) is an optimized approach that integrates Chip Authentication (CA) into PACE to streamline the authentication process. It eliminates the need for a separate CA step by reusing the ephemeral keys from PACE directly in the Chip Authentication process.

The CSP may offer PACE-CAM with an optional Terminal Authentication (TA) step, as specified for PROTOCOL_PACE_CAM. For this protocol, the CSP shall handle at least the following processes:

1. PACE: The CSP shall evaluate MSE:SET AT to establish a secure channel between the terminal and the CSP (chip) using password resources configured for SEC_PACE_CAN and SEC_PACE_MRZ.
2. CAM: The CSP shall map the ephemeral keys from the PACE to the CA process, authenticate itself to the terminal and derive a new set of session keys from a key resource configured for SEC_CA1.
3. TA1: The CSP shall verify the terminal's certificate chain to ensure it is authorized to access CSP services, using a trust anchor certificate resource configured for SEC_TA_IS_ROOT.

The CSP may handle the following additional security processes during protocol execution; the exact list shall be subject to *Modularity*:

- SEC_TA_DV: Return the terminal certificate received from the Document Verifier (DV).
- SEC_TA_TERMINAL: Return the terminal certificate received from individual terminals.
- TIME_SYNC_FROM_TA: Update the CSP's referenceTime using the validity of received TA certificates.
- POLICY_PRE_BLOCKED: Enforce that a CAN must be authenticated to restrict further *PACE* authentications when the tryCounter is already 1.

Same as for *EAC for eID*, the CSPSetValue, CSPClearResource commands, as well as the cert.manage operation, may be used to set or replace the TA trust anchor certificate.

The CSP shall ignore unknown extensions in TA certificates.

For further information, see section 6.4.1.4.4, PACE CAM States.

3.7.2.5 SCP03

GlobalPlatform Secure Channel Protocol '03' (SCP03), as specified for `PROTOCOL_SCP03`, is designed for a wide range of use cases, offering protection against eavesdropping and replay attacks while ensuring data integrity and confidentiality. It uses symmetric key cryptography to provide mutual authentication between an Off-card Entity (OCE) and the CSP through the derivation of shared session keys:

1. Session Encryption Key (S-ENC Key)
2. Session Message Authentication Code Key (S-MAC Key)

Note: The SCP03 Data Encryption Key (Key-DEK) is not part of the CSP functionality. If required, its key value can be imported into a CSP Resource and used with the *Cipher Module*, *Transform Module*, or *Confidential Data Transfer Module*.

For further information, see section 6.4.1.4.5, SCP03 & SCP04 States.

3.7.2.5.1 Key-ENC

The Session Encryption Key (S-ENC Key) is derived from a Static Secure Channel Encryption Key (Key-ENC), as specified for `SEC_KENC`, which is used to encrypt outgoing data and decrypt incoming messages. The OCE can choose from the following security levels:

- R-Encryption: Activating this option encrypts all outgoing responses.
- C-Decryption: Activating this option decrypts all incoming messages.

3.7.2.5.2 Key-MAC

The Session MAC Key (S-MAC Key) is derived from a Static Secure Channel Message Authentication Code Key (Key-MAC), as specified for `SEC_KMAC`, and is used to generate a Command MAC (C-MAC) for incoming messages and a Response MAC (R-MAC) for outgoing data.

These MACs ensure message integrity and authenticity, confirming that the messages have not been tampered with and are from legitimate sources. The OCE can choose from the following security levels:

- C-MAC: Activating this option verifies the MAC of all incoming messages.
- R-MAC: Activating this option ensures that all outgoing data includes a MAC.

3.7.2.6 SCP04

GlobalPlatform Secure Channel Protocol '04' (SCP04), as specified for `PROTOCOL_SCP04`, is a configurable version of *SCP03* that allows flexible configuration of its building blocks. These include options for data derivation, MAC calculation, rekeying, cipher, sensitive data encryption and random number generation.

For further information, see section 6.4.1.4.5, SCP03 & SCP04 States.

3.7.3 Building Blocks for Custom Secure Channel Protocols

The CSP exclusively supports built-in protocols that already specify commands for the establishment and management of secure channels. Protocols lacking these definitions are not available as built-in options. To address scenarios where Client Applications require secure channel protocols not directly available as a CSP protocol type, Client Applications can construct custom secure channels by leveraging the CSP's services for key agreement, combined with key derivation and subsequent cipher operations:

- *Key Operations*: A building block providing key agreement and key derivation services.
- *Cipher Operations*: A building block providing encryption and decryption services.

- *Signature Operations*: A building block providing signature creation and verification services.

Examples of such custom secure channels, including Mobile Driving Licence (MDL) and Elliptic Curve Integrated Encryption Scheme (ECIES), are available in Annex A, *Custom Secure Messaging*.

Table 3-2 provides an overview of the algorithms supported by the CSP. Detailed definitions are provided in their respective sections. Each algorithm shall adhere to the allowed combinations specified in:

- Table 6-4 for compatible *Cipher, Padding, and Key Size Combinations*
- Table 6-16 for compatible *Signature, Padding, Hash, Key Size, and Curve Combinations*
- Table 6-64 for compatible *Key Derivation, Resource Type, Hash, Size, and Curve Combinations*
- Table 6-67 for compatible *Key Agreement, Key Type, Curve, and Size Combinations*

Table 3-2: Algorithm Overview

Padding	Hash	Cipher	Signature	Key Derivation	Key Agreement
PAD_NOPAD	ALG_SHA_256	CIPHER_AES_CBC	SIG_AES_CMAC128	KDF_AES_CMAC	KAS_ECKA_DH
PAD_PKCS7	ALG_SHA_384	CIPHER_AES_CFB	SIG_AES_MAC128	KDF_ECC	KAS_ECKA_EG
PAD_ISO9797_1_M2_ALG3	ALG_SHA_512	CIPHER_AES_CTR	SIG_HMAC	KDF_HKDF	
PAD_PKCS1_PSS	ALG_SHA3_256	CIPHER_AES_GCM	SIG_RSA	KDF_PBKDF2	
PAD_PKCS1_OAEP_SHA256	ALG_SHA3_384	CIPHER_AES_CCM	SIG_ECDSA		
PAD_PKCS1_OAEP_SHA384	ALG_SHA3_512	CIPHER_AES_XTS	SIG_ECDSA_PLAIN		
PAD_PKCS1_OAEP_SHA512		CIPHER_RSA	SIG_EC_SCHNORR		
PAD_PKCS1_OAEP_SHA3_256					
PAD_PKCS1_OAEP_SHA3_384					
PAD_PKCS1_OAEP_SHA3_512					

3.7.4 Confidential Data Transfer

The CSP may offer support to transfer confidential data from session encryption in a secure messaging channel to storage-layer encryption for long-term storage, and vice versa, through the *Confidential Data Transfer Module*; the choice shall be subject to *Modularity*. This functionality is designed to prevent Client Applications from handling confidential data in plain text.

The Confidential Data Transfer feature extends the *Secure Channel Module*, providing the same features as the secure channel service while adding operations to transfer incoming data to a storage layer using a storage key explicitly configured for confidential data handling.

Client Applications shall not be able to configure keys or algorithms for confidential data transfer. The CSP Admin shall configure these through the `CSPCipherAlgorithms` and `CSPSecureChannelAlgorithms` structures, leveraging the CSP's *Resource Management* capabilities to define:

- The resource type, key type, size and/or curve (e.g., `KEY_AES`, 256 bit).
- The security usage type of the resource for the secure messaging channel (e.g., `SEC_PACE_PIN`).
- The cipher algorithms for the storage encryption (e.g., `CIPHER_AES_XTS`).

Client Applications may pass the `resourceId` referring to these configurations to *Confidential Data Transfer Operations*.

Only resources with `USAGE_SECCHANNEL` providing the secure messaging configuration and resources with `USAGE_CONFIDENTIAL` providing the storage-layer encryption configuration, all in `STATE_OPERATIONAL`, shall be accepted for these operations.

Note 1: The difference between the CSP's confidential data transfer and cipher operations is that, in confidential data transfer, the CSP securely switches encryption from one algorithm or key to another in a single step, without exposing the plaintext data to the Client Application. In contrast, when using a combination of the secure channel service and cipher service, the Client Application would have to handle the plaintext data, even if only briefly.

Note 2: Resources with `USAGE_CONFIDENTIAL` cannot be used in standard CSP cipher operations, as standard CSP cipher operations accept only resources with `USAGE_CIPHER`.

For further information, see section 2.2, Requirements, and section 6.5, Confidential Data Transfer Module.

3.7.4.1 Motivation for Confidential Data Transfer

The secure channel services provided by the CSP establish a secure messaging channel between the CSP Instance and an Off-card Entity (OCE). Authorized Client Applications can use the `sc.wrap` and `sc.unwrap` operations to encrypt outgoing messages and decrypt incoming ones, allowing them to access data transmitted through this channel in unencrypted form. This functionality enables Client Applications to implement their specific use cases.

However, allowing Client Applications to handle confidential data, such as passwords, secrets or sensitive personal information, in an unencrypted form may introduce security risks.

To mitigate these risks, the CSP offers a confidential data transfer service that offers direct encryption transformation for session decryption and storage encryption in a single atomic operation through `sc.confidentialWrap` and for storage decryption and session encryption through `sc.confidentialUnwrap`. This approach ensures that confidential data remains encrypted throughout its journey, without being exposed in plaintext to the Client Applications.

3.7.4.2 Receiving Confidential Data

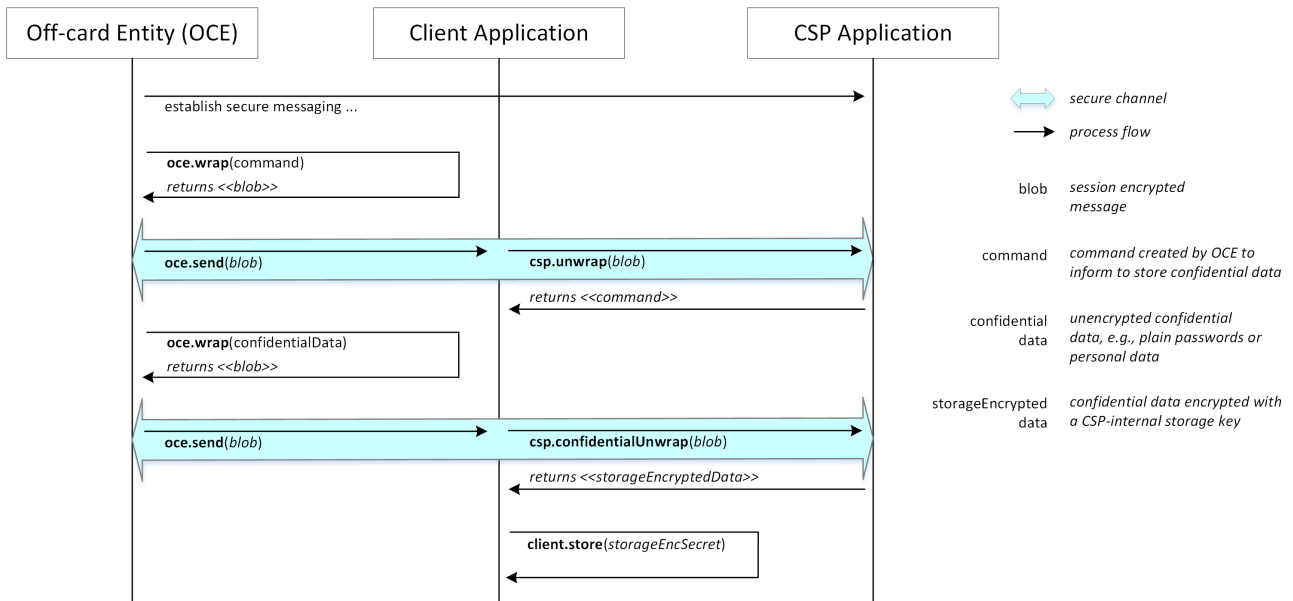
To use the Confidential Data Transfer service, the Client Application initializes the service with all resources required for secure messaging, along with a storage key, a resource with `USAGE_CONFIDENTIAL` and a configured cipher algorithm, used by the CSP to re-encrypt confidential data.

When the Client Application invokes the `sc.confidentialUnwrap` method on session-encrypted received data, the CSP removes the session encryption and directly re-encrypts the data using the configured storage key, as illustrated in Figure 3-3.

The Client Application, responsible for storing the encrypted data, does not need to access the data in plaintext.

Note: While the CSP facilitates the encryption and decryption processes, the CSP does not handle the persistent storage of encrypted data blocks. Managing the storage location and persistence of this data is the responsibility of the Client Application.

Figure 3-3: Confidential Data Personalization Flow Chart



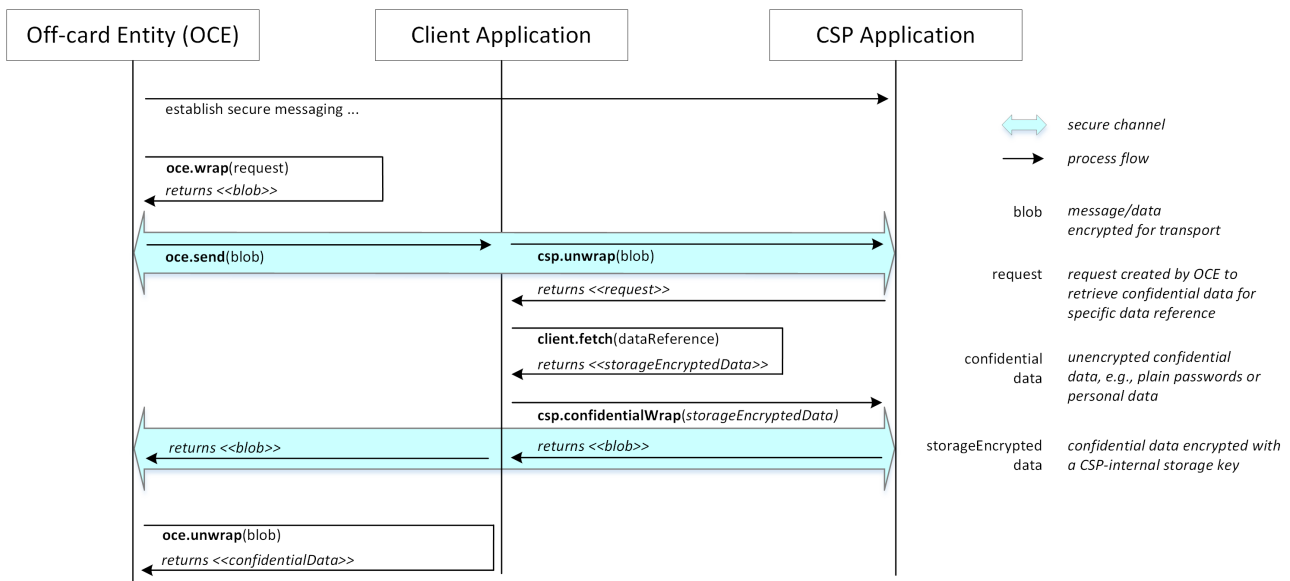
3.7.4.3 Sending Confidential Data

For OCEs requiring access to read confidential data, the CSP offers the `sc.confidentialWrap` method, which directly transfers data from storage-layer encryption to session encryption, facilitating easy access for OCE reader applications.

The `sc.confidentialWrap` operation re-encrypts confidential data that was previously encrypted with a storage key configured for `USAGE_CONFIDENTIAL`, transitioning the encryption from this storage-layer to the session encryption of the currently active secure channel. This enables the Client Application to securely transmit the data back to the same or other remote applications, ensuring that the Client Application itself does not need to handle the data in an unencrypted form, as illustrated in Figure 3-4.

The storage key used for decrypting confidential data may be restricted by policy rules. For example, access to decrypt confidential data, such as human fingerprints, may be constrained by a `POLICY_TA2_ACCESS_FLAG`, requiring a specific access flag encoded in the TA2 certificate, received during Terminal Authentication, as specified for `SEC_TA_TERMINAL` and `SEC_TA_DV`.

Figure 3-4: Confidential Data Accessing Flow Chart



3.7.4.4 Combination Transform Service with Confidential Data Transfer

While the Confidential Data Transfer service enables Client Applications to handle confidential data without accessing it in plaintext, a Client Application could still invoke the `sc.unwrap` operation to decrypt confidential data, thereby gaining access to unencrypted information. To prevent this, a combination of the CSP's *Transform Module* and *Confidential Data Transfer Module* can ensure that Client Applications are technically unable to decrypt confidential data.

In scenarios where highly sensitive information, such as human fingerprints or health records, needs to be stored on a user's device without being directly accessible, dual-layer encryption may protect confidential data as follows:

- **Transport-Layer Encryption:** A trusted OCE encrypts the confidential data using a transport key shared with the CSP but unknown to the Client Application. This transport key must be imported by the CSP Admin.
- **Session Encryption:** The encrypted data is then wrapped with session encryption for transmission to the Client Application.

Upon receiving the data, the Client Application processes it as follows:

- **Remove session encryption:** The Client Application removes session encryption using `sc.unwrap`. Due to the dual-layer encryption, the data remains encrypted with the transport key.
- **Re-encrypt with storage key:** The Client Application initiates the transform service, using the transport key (with `USAGE_TRANSFORM`) for decryption and a storage key (with `USAGE_CONFIDENTIAL`) for encryption to re-encrypt the data from transport-layer to storage-layer. The storage key may be randomly generated by the CSP or imported by the CSP Admin.
- **Secure storage:** The Client Application stores the re-encrypted data on the user's device.

To send confidential data to other trusted OCEs, which do not have access to the storage and transport keys, the Client Application can use the `sc.initConfidentialWrap` operation. This operation re-encrypts the data from storage-layer to session encryption using the storage key and the active session keys. The authenticated OCE can then access the confidential data by simply removing the session encryption, as illustrated in Figure 3-4.

3.8 Attestations

The CSP shall support computing *System Attestations*. The CSP may support computing attestations for data and keys, through the *Attestation Module*; the choice shall be subject to *Modularity*. Attestation is a security mechanism designed to securely verify the identity and authenticity of components, such as the SE platform, a public key or a counter value.

Client Applications shall not be able to select the attestation parameters and algorithms. The CSP Admin shall configure these through the CSPAlgorithms structure, leveraging the CSP's *Resource Management* capabilities to define:

- The resource type, key type, size and/or curve of the attestation key (e.g., RESOURCE_KEY, KEY_ECC_PRIVATE, CURVE_BRAINPOOL_P512_R1).
- The attestation signing algorithm (e.g., SIG_ECDSA).
- Fields to be added to the attestation data (e.g., FIELD_SYSTEM_TIME, FIELD_PUBLIC_KEY).

The difference between the CSP's attestation and signing operations is that, for attestation, the CSP computes the data to be attested, and the Client Application may only add data that gets appended to the attestation. In contrast, for signing operations, the Client Application has full control over the data to be signed.

Note: Resources with USAGE_ATTESTATION used as attestation keys cannot be used in standard signature operations provided by the CSP.

For further information, see section 2.2, Requirements, and section 6.6, Attestation Module.

3.8.1 Platform Attestation

The CSP shall support verification of the authenticity and identity of the CSP-enabled Secure Element.

Client Applications may initiate the platform attestation using the att.computeAttestation operation with ATTESTATION_PLATFORM and a unique nonce to prevent replay attacks.

In this operation, the CSP shall utilize the GlobalPlatform Controlling Authority Security Domain (CASD) technology, as specified in [GP Amd A], to sign the attestation data defined in ASN 6-18 CSPPlatformAttestation. This format includes:

- Name of the SE platform, as specified for platformName.
- Version of the CSP Protocol, the CSP API, and the CSP ELF, as specified for cspProtocolVersion, cspApiVersion, and cspELFVersion.
- Optional DLOA certification information about the CSP and SE chip according to Platform_DLOA and Application_DLOA from [GP DLOA] section 4.4.
- CASD signing information according to SecurityEnvironmentTemplate from [GP Amd A] section 5.3.1.

This attestation type requires a CASD key for signing. The SE Admin may configure the CASD key as specified in [GP Amd A] section 3.3. In the absence of a CASD key, the attestation computation shall fail with an ERROR_ILLEGAL_CONFIG [0x3006] exception.

For further information, see section 2.2, Requirements, and section 5.1.1.3, System Attestations.

3.8.2 Config Attestation

The CSP shall support verification of the authenticity and identity of the utilized CSP Instance and its use-case-specific configuration.

Client Applications may initiate the config attestation using the `att.computeAttestation` operation with `ATTESTATION_CONFIG` and a unique nonce to prevent replay attacks.

In this operation, the CSP shall use a CSP Instance-specific `configAttestationKey` to sign the attestation data defined in ASN 6-19 `CSPConfigAttestation`. This format includes:

- The AID of the CSP Instance.
- Name of the SE platform, as specified for `platformName`.
- Version of the CSP Protocol, the CSP API, and the CSP ELF, as specified for `cspProtocolVersion`, `cspApiVersion`, and `cspELFVersion`.
- An identifier (e.g., a name) of this CSP configuration, as specified for `configName`.
- Version of this CSP configuration, as specified for `configVersion`.

The `configAttestationKey` shall be configured by the CSP Admin through the `CSPSettings` structure. As a standard CSP Resource, it may be imported, generated or derived using the *Key Management* operations provided by the CSP. If the `configAttestationKey` is not configured, the attestation computation shall fail with an `ERROR_ILLEGAL_CONFIG [0x3008]` exception.

The CSP shall support the `configAttestationKey` parameters listed in Table 3-3, and the *Signature, Padding, Hash, Key Size, and Curve* Combinations specified in Table 6-16 if supported by the platform.

Only resources with `USAGE_ATTESTATION` in `STATE_OPERATIONAL` shall be accepted for config attestation.

The config attestation shall utilize the CSP's *Signatures with Counters & Timestamps* functionality, allowing the CSP Admin to configure *Fields* to be included into the attestation result via the `CSPAttestationAlgorithms` structure, which is configured to the `configAttestationKey`.

For further information, see section 2.2, Requirements, and section 5.1.1.3, System Attestations.

Table 3-3: Attestation Key Parameters

Parameter	Value	Description
Key Type	<code>KEY_ECC_PRIVATE</code>	Elliptic Curve Cryptography (ECC) private key.
ECC Domain Parameters	<code>CURVE_SEC_P256_R1</code>	NIST 256-bit curve.
Signature Algorithm	<code>SIG_ECDSA</code>	Elliptic Curve Digital Signature Algorithm (ECDSA) for signing.
Padding Algorithm	<code>PAD_NULL</code>	No padding is applied to the data.
Message Digest Algorithm	<code>ALG_SHA_256</code>	Message digest algorithm SHA-256.

3.8.3 Data Attestation

The CSP may support creating data attestations by signing external input data; the choice shall be subject to *Modularity*. Unlike a standard signature created with the `SignatureModule`, the CSP combines resource values and/or pre-configured fields with the input data to produce signed attestation data.

The CSP Admin shall be able to configure *Fields* to be included in the attestation result via the `CSPAttestationAlgorithms` structure, which is configured to the attestation signing key.

Client Applications may initiate data attestations using the `att.computeAttestation` operation with `ATTESTATION_DATA`, the external input data and the following `resourceIds`:

- **Attestation Key:** The `resourceId` of the key used to sign the attestation data.
- **Public Resource:** Optional, the `resourceId` of a public key, a counter or a timer value to be included in the attestation result.

The attestation key shall be used to sign the data defined in ASN 6-20 CSPDataAttestation.

The CSP shall support attestation keys with parameters listed in Table 3-3, and *Signature, Padding, Hash, Key Size, and Curve* Combinations specified in Table 6-16 if supported by the platform.

Only resources with USAGE_ATTESTATION in STATE_OPERATIONAL shall be accepted as attestation key.

For example, to ensure the integrity of tax bills or invoices, each bill includes sequentially numbered and timestamped bill-specific information, all digitally signed. Such an attestation provides proof of integrity (bills cannot be inserted afterward), completeness (no bills have been removed) and non-repudiation (individual bills cannot be modified undetected). In the context of tax documentation, these proofs are essential for detecting potential tax or fiscal evasion.

For further information, see section 2.2, Requirements, section 3.9, Counter and Limits, section 3.10, Timers, and section 6.6, Attestation Module.

3.8.4 Proof of Possession Key Attestation

The CSP may support the creation of a cryptographic proof that the private counterpart of a public key is managed by the same CSP Instance; the choice shall be subject to *Modularity*. The attestation result includes a Proof of Possession (PoP) via an inner signature generated using the private key being attested, enabling third parties to verify that the corresponding private key is indeed managed by the CSP. An outer signature ensures the integrity of the attestation itself.

Client Applications may initiate PoP key attestation using the att.computeAttestation operation with ATTESTATION_KEY_POP, a unique nonce to prevent replay attacks and the following resourceIds:

- Private Key: The resourceId of the private key used to create the PoP signature for the public key being attested. It refers to the attestation signing algorithm configuration for the inner signature.
- Public Key: The resourceId of the public key that shall be attested.
- Attestation Key: The resourceId of the private attestation key to create the overall signature. It refers to the attestation signing algorithm configuration for the outer signature.
- Public Attestation Key: Optional, the resourceId of the public part of the attestation key, if it shall be included in the attestation result.

The private key being attested shall be used to sign its public part to create the inner signature for Proof of Possession as defined in ASN 6-21 CSPKeyPoPAttestation. The attestation resource shall be used to create the outer signature defined in this structure.

For both the resource being attested and the attestation key, the CSP shall support the parameters listed in Table 3-3. If the platform supports additional *Signature, Padding, Hash, Key Size, and Curve* Combinations from Table 6-16, the CSP shall enable configuration of distinct key parameters and algorithms for the inner and outer signatures, such as KAS_ECKA_DH-MAC based on SIG_AES_CMAC128 for PoP computing and SIG_ECDSA for the overall attestation signature.

The CSP shall only accept attestation keys configured for USAGE_ATTESTATION and private resources for the inner signature configured for USAGE_SIGNATURE. All resources utilized shall be in STATE_OPERATIONAL.

The PoP key attestation shall reuse the CSP's *Signatures with Counters & Timestamps* functionality, allowing the CSP Admin to configure *Fields* to be included in the attestation result via the CSPAttestationAlgorithms structure, which is configured to the attestation key.

For further information, see section 2.2, Requirements, and section 6.6, Attestation Module.

3.8.5 Generate Key Pair Attestation

The CSP may support generating a new key pair and returning the computed public key, along with proof that it corresponds to the newly generated private key, both managed by the same CSP Instance; the choice shall be subject to *Modularity*.

Client Applications may initiate the key generation attestation using the `att.computeAttestation` operation with `ATTESTATION_KEY_GENERATION`, along with a unique nonce to prevent replay attacks and the following `resourceIds`:

- **Private Key:** The `resourceId` of the private key that shall be newly generated with random data. This resource is used to sign an inner PoP signature. It must be in `STATE_UNINITIALIZED` with `ACCESS_SETUP` right granted.
- **Public Key:** The `resourceId` of the public key that shall be computed from the newly generated private key. Only resources in `STATE_UNINITIALIZED` with `ACCESS_SETUP` right granted are accepted.
- **Attestation Key:** The `resourceId` of the private attestation key to create the overall signature. It refers to the attestation signing algorithm for the outer signature. It must be in `STATE_OPERATIONAL`.
- **Public Attestation Key:** Optional, the `resourceId` of the public part of the attestation key, if it shall be included in the attestation result. The resource must be in `STATE_OPERATIONAL`.

This operation creates the same result as *Proof of Possession Key Attestation* defined in ASN 6-21 `CSPKeyPoPAttestation`. The only difference is that the public-private key pair to be attested will be freshly generated.

For both the private key being generated and the attestation key, the CSP shall support the parameters listed in Table 3-3. If the platform supports additional *Signature, Padding, Hash, Key Size, and Curve* Combinations from Table 6-16, the CSP shall enable configuration of distinct key parameters and algorithms for the inner and outer signatures.

The CSP shall only accept attestation keys configured for `USAGE_ATTESTATION` and private resources for the inner signature configured for `USAGE_SIGNATURE`.

The key generation attestation shall reuse the CSP's *Signatures with Counters & Timestamps* functionality, allowing the CSP Admin to configure *Fields* to be included in the attestation result via the `CSPAttestationAlgorithms` structure, which is configured to the attestation key.

For further information, see section 2.2, Requirements, and section 6.6, Attestation Module.

3.9 Counter and Limits

The CSP may support counters with optional maximum limits through the CounterModule; the choice shall be subject to *Modularity*. Counters may be managed internally within the CSP, for example, automatically incremented with each signature creation, or, for counters set to manual, increments can be actively triggered by Client Applications. Upon reaching the maximum limit, a counter triggers a state change for its associated CSP Resource, transitioning the resource to STATE_EXHAUSTED, making it unavailable for further use.

Client Applications shall not be able to configure counters or its incrementation mechanisms. The CSP Admin may configure counters through the CSPCounters, CSPCounter and CSPCounterSettings structures, leveraging the CSP's *Resource Management* capabilities to define:

- A resource of type RESOURCE_COUNTER for manual counting by Client Applications
- Activation of built-in counters for key, password or certificate resources (e.g., COUNT_USAGE)
- The counter capacity appropriate for the required increments (e.g., COUNTER_TINY, COUNTER_LARGE)
- An optional maximum limit before the resource is exhausted (e.g., 10 or 100,000)

Counter values may be used to create *Signatures with Counters & Timestamps* or returned directly through the *Data Attestation*.

The CSP Admin may reset exhausted resources using the CSPClearResource command. By default, Client Applications do not have permission to reset exhausted resources. The CSP Admin may configure ACCESS_CLEAR and ACCESS_SETUP rights to grant permissions for clearing and reinitializing exhausted resources.

For further information, see section 2.2, Requirements, section 3.6.7, Signatures with Counters, and section 6.10, Counter Module.

3.9.1 Overview Counters

This document specifies different kinds of counters, supporting the following scenarios:

- A *Usage Counter* activated for the audit signing key resource can serve as a log counter to create an audit log trail containing sequentially numbered log messages through FIELD_USAGE_COUNTER.
- A *Usage Counter per Block* with counterLimit activated for cipher key resources can be used to enforce CSP guidance on the maximum number of times a key may be reused before it is blocked, e.g., to enforce AES key usage limits for cipher operations listed in Table 6-3. This counter tracks each cipher block operation in persistent memory, including those from incomplete cipher operations.
- A *Success Counter* activated for PUK resources limits the number of times a PUK can be used to unblock a PIN.
- A *Failure Counter* can track security incidents, such as unauthorized password attempts or signature verification failures.
- A *Transport Counter* with counterLimit set to one can enforce a One-Time Password (OTP) by requiring a PIN or password marked as inTransport to be changed after its first authentication use.
- A *Timeout Counter* with a counterLimit set to 10 can serve as a substitute for time-based expiration of authenticated passwords when time is unavailable on the platform. The CSP decrements this counter with each operation that requires the authenticated password, such as signing operations restricted by POLICY_PASSWORD, and resets the counter after each successful password verification.
- The *Manual Counter* supports application-specific counting requirements not covered by built-in counters. The Client Application can trigger counter increments, which are securely performed within the CSP, and, if granted ACCESS_SETUP rights, can also reset it.

Note: The *Retry Counter with Maximum Try Limit*, which sets a password to STATE_BLOCKED after too many failed verifications, is managed within the CSP's *Password Module* and is not implemented as a counter type.

For further information, see section 5.2.4, Resource Lifecycle, and section 6.10.1.2, Counter Types.

3.9.1.1 Manual Counter

The CSP may support the counter type COUNT_MANUAL, enabling CSP Admins to create a CSP Resource of type RESOURCE_COUNTER; the choice shall be subject to *Modularity*.

Client Applications may increment this counter using the counter.increment operation. If the counter reaches its configured counterLimit, the CSP shall set the counter to STATE_EXHAUSTED. Attempts to increment an exhausted counter will fail with ERROR_NOT_ALLOWED [0x50A0].

The Client Application may retrieve the authentic, integrity-protected counter value using att.computeAttestation with ATTESTATION_DATA.

CSP Admins may configure the fields FIELD_MANUAL_COUNTER and FIELD_MANUAL_COUNTER_LIMIT using the CSPAttestationAlgorithms structure to automatically include the counter value in attestation results or audit messages.

3.9.1.2 Usage Counter

The CSP may support the counter type COUNT_USAGE, enabling CSP Admins to activate a usageCounter for key and password resources and to set a counterLimit to restrict the number of times the resource can be used in cryptographic operations provided by the CSP.

The CSP shall increment this counter for each invocation of an operation that utilizes the resource for cryptographic or security-related services, including:

- Successful or failed password verification.
- Final cipher decrypt and encrypt operations.
- Final signature creation operations.
- Final secure channel authentication.
- Final attestation computation.

If a maximum counterLimit is configured, the CSP shall, with each increment, check whether the counter exceeds the limit and shall set the resource owning the counter to STATE_EXHAUSTED, making it inoperative for any further cryptographic operations. If the limit is set to zero, the counter will continue to be incremented.

The usage counter can be used to implement key or password rotation or to detect unusual activities such as repeated login attempts. It may also serve as a log counter when included in attestation results or audit messages through FIELD_USAGE_COUNTER.

The CSP shall reset the usage counter each time a new value is assigned to the resource, as specified for COUNT_USAGE. This includes:

- Key generation and import.
- Keys derived through key derivation.
- Shared secrets created via key agreement.
- Password updates.

Note: The usage counter is not available for certificate resources.

For further information, see section 3.9.1.3, Block Counter, section 3.9.1.4, Success Counter and section 3.9.1.5, Failure Counter.

3.9.1.3 Usage Counter per Block

The CSP may support the counter type `COUNT_USAGE_PER_BLOCK`, enabling CSP Admins to activate a `blockUsageCounter` for key resources and to set a `counterLimit` to restrict the number of times the resource can be used in cryptographic operations provided by the CSP; the choice shall be subject to *Modularity*.

The CSP shall increment this counter for each cryptographic block operation, including:

- Each cipher block processed in `update` and `doFinal` operations.
- Each signature creation block processed in `update` and `sign` operations.
- Each block processed for secure channel authentication.

The CSP shall evaluate the `counterLimit` and reset this counter in the same manner as specified for the *Usage Counter*.

For further information, see section 3.9.1, Overview Counters, and section 3.9.1.2, Usage Counter.

3.9.1.4 Success Counter

The CSP may support the counter type `COUNT_USAGE_SUCCESS_ONLY`, enabling CSP Admins to activate a `successCounter` for key and password resources and to set a `counterLimit` to restrict the number of times the resource can be used; the choice shall be subject to *Modularity*.

The CSP shall increment, evaluate and reset this counter in the same manner as specified for the *Usage Counter*, with one difference: The CSP shall increment and evaluate this counter only for successfully completed operations. Operations resulting in negative return values, such as password verification failures, or exceptions during method invocations, are not counted.

For further information, see section 3.9.1, Overview Counters, and section 3.9.1.2, Usage Counter.

3.9.1.5 Failure Counter

The CSP may support the counter type `COUNT_USAGE_FAILURE_ONLY`, enabling CSP Admins to activate a `failureCounter` for key and password resources and to set a `counterLimit` to restrict the number of times the resource can be used; the choice shall be subject to *Modularity*.

The CSP shall increment, evaluate and reset this counter in the same manner as specified for the *Usage Counter*, with one difference: The CSP shall increment and evaluate this counter only for failed operations. Only operations resulting in negative return values, such as password verification failures, or exceptions during method invocations, are counted.

For further information, see section 3.9.1, Overview Counters, and section 3.9.1.2, Usage Counter.

3.9.1.6 Transport Counter

The CSP may support the counter type `COUNT_TRANSPORT_USAGE`, enabling CSP Admins to activate a `transportUsageCounter` for password resources and to set a `counterLimit` to restrict the number of times the password can be used before requiring change; the choice shall be subject to *Modularity*.

The CSP shall increment this counter upon each successful password verification and shall set the password to `STATE_EXHAUSTED` when the `counterLimit` is exceeded.

The counter is active only when the password's `inTransport` flag is set. The CSP shall deactivate the counter upon the first password change when resetting `inTransport`.

This transport counter can be used to implement One Time Passwords (OTP) by requiring a password change after the first use.

For further information, see section 3.9.1, Overview Counters, and section 3.5.4, Passwords in Transport.

3.9.1.7 Timeout Counter

The CSP may support the counter type `COUNT_AUTH_USAGE`, enabling CSP Admins to activate an `authUsageCounter` for password resources and set a `counterLimit` to restrict the number of uses before re-authentication is required; the choice shall be subject to *Modularity*. This timeout counter functions similarly to an *Authentication Timeout* but is intended for cases where a time-based mechanism is unavailable.

The CSP shall reset this counter after each successful password verification through the `pwd.check` operation and shall increment it each time an operation requires an authenticated password, including those enforced by `POLICY_PASSWORD`, `POLICY_UNBLOCK_PASSWORD`, and `POLICY_PRE_BLOCKED`. When the counter reaches its configured limit, the CSP shall reset the authenticated flag of the password.

Note: Timeout counters only reset the authenticated flag; they do not transition the password to `STATE_EXHAUSTED`.

For further information, see section 3.9.1, Overview Counters.

3.9.2 Maximum Counter Increment Capacities

All counter types, except timeout-related counters, shall store each increment in persistent memory. This is essential to ensure the integrity and continuity of the counters, even after a system restart or unexpected shutdown.

However, each write operation affects the memory's lifespan, as frequent writing can cause certain memory areas to wear out faster than others. Wear leveling is a technique that distributes write operations across memory to extend its overall lifespan.

The CSP Admin shall configure the maximum number of expected changes for each counter, allowing the CSP to select an appropriate wear leveling technique. The CSP may support the following capacities:

- 1-byte range supporting up to 10,000 changes, as specified for `COUNTER_TINY`.
- 2-byte range supporting up to 10,000 changes, as specified for `COUNTER_SMALL`.
- 4-byte range supporting up to 100,000 changes, as specified for `COUNTER_MEDIUM`.
- 4-byte range supporting up to 5,000,000 changes, as specified for `COUNTER_LARGE`.

For further information, see section 6.10.1.3, Counter Capacities.

3.10 Timer and Time Management

The CSP may support timers and time management through the *Time Module*; the choice shall be subject to *Modularity*. Depending on platform capabilities, time support may be either `timeSinceBoot`, representing the elapsed time in seconds since system boot, or `systemTime`, a Unix timestamp estimated from a `referenceTime` provided by an external source. This time information is used by the CSP to enforce timers, such as validity dates for certificates and keys, and to transition the resource to `STATE_EXPIRED` once the validity date is reached.

Client Applications shall not be able to configure timers or time management settings. The CSP Admin may configure timers through the `CSPTimers`, `CSPManualTimer`, and `CSPTimeSettings` structures, leveraging the CSP's *Resource Management* capabilities to define:

- A resource of type `RESOURCE_TIMER` for manual time-based restrictions.
- Activation of built-in timers for key, password or certificate resources (e.g., `TIMER_VALIDITY_DATE`).
- The time synchronization mechanism (e.g., `TIME_SYNC_FROM_TA`, `TIME_SYNC_FROM_CLIENT`).
- Adding timestamps to attestations and log messages (e.g., `FIELD_SYSTEM_TIME`, `FIELD_TIME_SINCE_BOOT`).

The CSP Admin may reset expired resources using the `CSPClearResource` command. By default, Client Applications do not have permission to reset expired resources. The CSP Admin may configure `ACCESS_CLEAR` and `ACCESS_SETUP` rights to grant permissions for clearing and reinitializing expired resources.

Note: The CSP does not support time zones and may not provide a secure real-time clock.

For further information, see section 2.2, Requirements, and section 6.11, Time Module.

3.10.1 Overview Timers

This document specifies different kinds of timers, supporting the following scenarios:

- A *Validity Period* for keys and passwords can be used to enforce mandatory password changes or key changes by setting a validity period after which the resource expires and requires updating.
- The *Validity Date for Keys and Passwords* ensures a resource expires on a specific date. Unlike a validity period - where the CSP recalculates expiration dates upon each update – the validity date is a fixed timestamp, beyond which the resource cannot be used.
- A *Validity Date of Certificates* is a fixed timestamp extracted from the certificate content, used by the CSP to limit the certificate's use in cryptographic operations.
- An *Authentication Timeout* resets the authenticated flag of a password upon expiration.
- A *Security Timeout* resets an established secure channel when it expires.
- The *Manual Timer* can be used to implement custom time-based restrictions, such as enforcing an expiration date for eID documents or verifying age thresholds.

For further information, see section 5.2.4, Resource Lifecycle, and section 6.11.1.3, Timer Types.

3.10.1.1 Manual Timer

The CSP may support the timer type `TIMER_MANUAL_DATE` or `TIMER_MANUAL_PERIOD`, enabling CSP Admins to create a CSP Resource of type `RESOURCE_TIMER`; the choice shall be subject to *Modularity*.

Client Applications may start a timer of type `TIMER_MANUAL_PERIOD` using the `time.reset` operation. If the timer expires, the CSP shall set the timer to `STATE_EXPIRED`.

CSP Admins may configure the fields `FIELD_RESOURCE_STATE` and `FIELD_MANUAL_TIMER` using the `CSPAttestationAlgorithms` structure to automatically include timer information in attestation results or audit messages.

3.10.1.2 Validity Period

The CSP may support the timer type `TIMER_VALIDITY_PERIOD`, enabling CSP Admins to configure a `validityPeriod` in seconds for key and password resources; the choice shall be subject to *Modularity*.

The CSP shall compute a validity date from this period each time a new resource value is set, such as through key generation, key derivation, key import or password updates.

Each time the resource is used for cryptographic operations (e.g., cipher or signature operations, secure channel authentications, password verifications), the CSP shall check if the computed validity has expired. If expired, the CSP shall transition the resource to `STATE_EXPIRED`, making it unusable for further use in cryptographic operations.

An expired resource can only be reset by clearing it through `resource.clear` or `CSPClearResource` and reinitializing it with a new value.

Note: The validity period is not available for certificate resources.

For further information, see section 3.10.1, Overview Timer.

3.10.1.3 Validity Date for Keys and Passwords

The CSP may support the timer type `TIMER_VALIDITY_DATE`, enabling CSP Admins to configure a specific `validityDate` as Unix timestamp in seconds for key and password resources; the choice shall be subject to *Modularity*.

The CSP shall evaluate the validity date the same manner as specified for the *Validity Period*, with the difference that the CSP Admin sets a fixed timestamp as validity date, which is not recomputed by the CSP.

Note: Configuring a specific validity date is not available for certificate resources.

For further information, see section 3.10.1, Overview Timer.

3.10.1.4 Validity Date of Certificates

The CSP may support the timer type `TIMER_VALIDITY_CERTIFICATE`, enabling CSP Admins to activate `validityCertificate` for certificate resources; the choice shall be subject to *Modularity*.

When enabled, the CSP extracts the validity date from the 'Certificate Validity Period' field within the certificate content and evaluates it in the same way as a validity date computed from a *Validity Period*.

For further information, see section 3.10.1, Overview Timer.

3.10.1.5 Authentication Timeout

The CSP may support the timer type `TIMER_AUTH_TIMEOUT`, enabling CSP Admins to configure an `authTimeout` in seconds for password resources; the choice shall be subject to *Modularity*.

The CSP shall activate this timer each time a password is successfully verified and set to authenticated.

The CSP shall check this timer with each method invocation that requires an authenticated password, such as `pwd.update`, `pwd.isAuthenticated`, and `POLICY_PASSWORD`. When the timer expires, the CSP shall reset the password's authenticated flag, forcing a new password verification.

Additionally, the CSP Admin may configure the `timeoutType` that distinguishes between two types:

- `TIMEOUT_HARD`: Relies on a fixed timeout that starts after successful authentication. Once this limit is reached, re-authentication is required, regardless of resource usage.
- `TIMEOUT_SOFT`: The timeout is reset with each use of the authenticated resource, allowing continuous usage as long as the resource is actively utilized within these intervals.

Note: Like the *Timeout Counter*, the authentication timeout only resets the authenticated flag; it does not change the password to `STATE_EXPIRED`.

For further information, see section 3.10.1, Overview Timer.

3.10.1.6 Security Timeout

The CSP may support the timer type `TIMER_SECURITY_TIMEOUT`, enabling CSP Admins to configure a `securityTimeout` in seconds for secure channels; the choice shall be subject to *Modularity*.

The CSP shall start this timer when the secure channel enters the `SECURITY_ESTABLISHED` state.

The CSP shall check the timer on each method invocation that relies on the authenticated secure channel, including operations such as wrapping (`sc.initWrap`), unwrapping (`sc.initUnwrap`), or the `POLICY_SECCHANNEL_ESTABLISHED` policy. If the timer expires, the CSP shall reset the security of the secure channel, forcing a new secure channel authentication.

Note: The security timeout only resets the secure channel; it does not change any resource to `STATE_EXPIRED`.

For further information, see section 3.10.1, Overview Timer.

3.10.2 Time Synchronization

Given the limited capacity of secure elements, the CSP must rely on an external time source to estimate a `systemTime`. This requires the target platform to support basic timer functionality to retrieve the elapsed `timeSinceBoot`. The CSP shall combine this elapsed time with a `referenceTime` from an external source. The system time is then calculated by adding the elapsed time to the reference time and subtracting the elapsed time recorded when the reference time was set.

The CSP may support the following *Time Synchronization Methods*; the exact list shall be subject to *Modularity*.

- CSP Admin may set the `referenceTime` using the `CSPSetTime` command.
- Client Applications may set the `referenceTime` through the `time.setReferenceTime` operation, if `TIME_SYNC_FROM_CLIENT` is enabled by the CSP Admin.
- The CSP may update the `referenceTime` from the creation date of Terminal Authentication (TA) certificates used in EAC secure messaging protocols, as specified for `TIME_SYNC_FROM_TA`.

The CSP may support additional security measures to enhance the trustworthiness of the `referenceTime`, the exact list shall be subject to *Modularity*. These security measures can be enabled or disabled by the CSP Admin and may include:

- Force the CSP to verify a signature for the new `referenceTime`, as specified for `TIME_SYNC_VERIFY_SIG`.
- Use a challenge generated by the CSP, as specified for `TIME_SYNC_VERIFY_SIG_WITH_CHALLENGE`.
- Accept only timestamps newer than the last set time, as specified for `TIME_SYNC_ENFORCE_NEWER`, making it impossible to go backward in time.
- Store the `referenceTime` in persistent memory, as specified for `TIME_SYNC_PERSIST`.

The choice how to configure time synchronization is a trade-off between trust in the time source and the accuracy of the estimated system time. Using TA certificates can ensure the authenticity of the time source but may result in a system time that significantly differs from the actual real-world time. To ensure an accurate system time, the reference time may need to be updated regularly, potentially every few minutes or hours, which may require relying on Client Applications as external timekeepers. Trust in this approach is enhanced by using signed timestamps or accepting only timestamps that are newer than the previously set ones. However, not all use cases might need or benefit from these configuration adjustments.

Note: The CSP may track time synchronization activities through `EVENT_CSP_TIME_SET`, which is logged each time the `referenceTime` changes. This event records three timestamps: the previous reference time, the new reference time, and the timestamp when the reference time has changed.

For further information, see section 6.11.2, Time Sync Mechanisms.

3.10.2.1 TA Certificates as Time Source

The CSP may support updating its `referenceTime` from Terminal Authentication (TA) certificates; the choice shall be subject to *Modularity*. According to [TR-03110-3] section 2.1, TA certificates have a short validity period. When the CSP receives these TA certificates from reader applications during the authentication process via `sc.processSecurity`, the CSP shall verify the creation date of these certificates. If this date is later than the current `referenceTime` or if no `referenceTime` has been set, the CSP shall update the `referenceTime` accordingly, as specified for `TIME_SYNC_FROM_TA`.

Note: This approach is used in Extended Access Control (EAC), such as *EAC for eID* or *EAC for MRTD*, where TA certificates, typically valid for a few days or at most 30 days, are used to authenticate terminals.

For further information, see section 3.7.2.2, EAC for eID, section 3.7.2.3, EAC for MRTD, and section 3.7.2.4, PACE CAM.

3.10.2.2 External Timekeeper

The CSP shall support setting the `referenceTime` by the CSP Admin through the `CSPSetTime` command and may allow Client Applications to update the `referenceTime`, as specified for `TIME_SYNC_FROM_CLIENT`; the choice shall be subject to *Modularity*.

Details such as the frequency of time updates and which security measures to enable depend on the specific use case and may vary as follows:

- The Client Application may retrieve signed timestamps at regular intervals (e.g., every 30–60 minutes) from a trusted remote source using a challenge-response mechanism. In this process, the CSP generates a challenge and provides it to the Client Application, which forwards it to the trusted source. The trusted source signs the challenge along with the current timestamp and returns the signed data to the Client Application. The CSP then verifies the signature using the challenge and, upon successful verification, updates the `referenceTime`.
- Alternatively, the Client Application might request the current timestamp from an off-card application, like a toolkit on a mobile phone, and then update the CSP with the new `referenceTime` either at every system power-on or upon selecting the Client Application.
- Furthermore, the CSP Admin may set the `referenceTime` remotely through the `CSPSetTime` command before finalizing the CSP configuration.

Note: It is recommended to develop a specialized Client Application solely dedicated to acting as a timekeeper, not performing any cryptographic operations, to ensure accurate time synchronization.

3.10.2.3 Time Verification Key

The CSP may support the verification of signed timestamps to update the `referenceTime`, as specified for `TIME_SYNC_VERIFY_SIG`; the choice shall be subject to *Modularity*. The signature verification may enhance trust in new reference times provided by Client Applications through `time.setReferenceTime` or by CSP Admins through `CSPSetTime`.

The CSP shall use a time verification key managed per CSP Instance to verify the authenticity of the provided timestamp. However, this check alone cannot prevent the use of an older, previously recorded signed timestamp. To mitigate this, the `TIME_SYNC_PERSIST` and `TIME_SYNC_ENFORCE_NEWER` can help ensure that the reference time cannot move backward in time.

The `timeVerificationKey` shall be configured by the CSP Admin through the `CSPTimeSettings` structure. As a standard CSP Resource, it may be imported, generated or derived using the *Key Management* operations provided by the CSP. If `TIME_SYNC_VERIFY_SIG` is enabled but the `timeVerificationKey` is not configured, any attempt to set a new reference time shall fail with an `ERROR_ILLEGAL_CONFIG [0x30B0]` exception.

The CSP shall support the `timeVerificationKey` parameters listed in Table 3-4, and the *Signature, Padding, Hash, Key Size, and Curve* Combinations specified in Table 6-16 if supported by the platform.

Only resources with `USAGE_SIGNATURE` in `STATE_OPERATIONAL` shall be accepted for time verification.

Note: Even if `TIME_SYNC_VERIFY_SIG` is enabled, reference times updated from TA certificates are not verified using the `timeVerificationKey`.

Table 3-4: Time Verification Key Parameters

Parameter	Value	Description
Key Type	<code>KEY_ECC_PUBLIC</code>	Elliptic Curve Cryptography (ECC) public key.
ECC Domain Parameters	<code>CURVE_SEC_P256_R1</code>	NIST 256-bit curve.
Signature Algorithm	<code>SIG_ECDSA</code>	Elliptic Curve Digital Signature Algorithm (ECDSA) is used to verify the new reference time.
Padding Algorithm	<code>PAD_NULL</code>	No padding is applied to the data.
Message Digest Algorithm	<code>ALG_SHA_256</code>	Message digest algorithm SHA-256.

3.10.3 Time Not Available

The CSP may support the following modes to handle scenarios when time is not supported or not synchronized:

- In `TIME_MODE_OFF`, the CSP ignores and does not evaluate time-related configurations, such as validity dates and timeouts, and returns the appropriate number of `0xFF` depending on the time format.
- In `TIME_MODE_IGNORE`, the CSP evaluates timers and returns time information if available. If time is not available, the CSP ignores time-related configurations and returns the appropriate number of `0xFF` depending on the time format.
- In `TIME_MODE_STRICT`, the CSP fails with `ERROR_NOT_SUPPORTED (0x80B1)` when a timer or time information is requested that is not supported by the platform, and with `ERROR_INVALID_INIT (0x40B0)` when the `systemTime` is required but not synchronized; e.g., if no new `referenceTime` was set since the last system startup.

For further information, see section 6.11.1.1, Time Modes.

3.10.3.1 Time Not Synchronized

Even if the target platform is equipped with a timer that can retrieve the elapsed `timeSinceBoot`, the `systemTime` may not be available if the CSP lacks a `referenceTime`.

In this case, when `TIME_MODE_IGNORE` is active, features that rely on time shall behave as follows:

- The *Manual Timer* shall be ignored and will never expire.
- Any configured *Validity Period*, *Validity Date for Keys and Passwords*, or *Validity Date of Certificates* shall be treated as if it does not exist. However, the CSP shall continue to operate normally without showing any errors.
- The *Authentication Timeout* and the *Security Timeout* shall be managed solely through the `timeSinceBoot` feature of the target platform and shall function normally, even when `systemTime` is unavailable.
- The field `FIELD_SYSTEM_TIME` shall be populated with four bytes of zeros followed by the `timeSinceBoot`. The `FIELD_TIME_SINCE_BOOT` shall operate as usual.

In `TIME_MODE_STRICT`, an unsynchronized `systemTime` shall be handled as follows:

- The *Manual Timer* shall throw `ERROR_INVALID_INIT [0x40B0]` on use.
- Any configured *Validity Period*, *Validity Date for Keys and Passwords*, or *Validity Date of Certificates* shall throw `ERROR_INVALID_INIT [0x40B0]` when the related resources are used for cryptographic operations.
- The *Authentication Timeout* and the *Security Timeout* shall function normally.
- The `FIELD_SYSTEM_TIME` shall trigger `ERROR_INVALID_INIT [0x40B0]` when used in signing operations, while `FIELD_TIME_SINCE_BOOT` shall function normally.

`TIME_MODE_OFF` behaves the same as when time support is unavailable.

3.10.3.2 No Time Support

The CSP may not support time handling if the target platform lacks any form of time management capabilities.

In `TIME_MODE_OFF` and `TIME_MODE_IGNORE`, the CSP shall behave as follows:

- All timers shall be treated as if they do not exist, without throwing any errors.
- The signature field `FIELD_SYSTEM_TIME` shall be populated with `0xFFFFFFFF`, and `FIELD_TIME_SINCE_BOOT` with `0xFFFF`.

In `TIME_MODE_STRICT`, any timers or signature fields that require time shall throw an `0x80B1` when used, with detailed error codes indicating the unsupported timer type, `systemTime` or `timeSinceBoot`.

3.11 Secure Auditing

The CSP may support secure auditing via the *Audit Module*; the choice shall be subject to *Modularity*. This feature creates integrity-protected audit logs for selected CSP operations, such as modifications to the reference time or failed *Password Verification*. Logs may be sequentially numbered and/or timestamped and shall be signed to ensure integrity.

By default, audit logging is deactivated. To enable audit logging, the CSP Admin shall activate it through *CSPAuditSettings* and configure *Audit Events* via *CSPSystemEvent* and/or *CSPResourceEvent*, leveraging the CSP's *Resource Management* capabilities to define:

- Resource-related events to log (e.g., *EVENT_SIGNATURE_VERIFIED*, *EVENT_PASSWORD_UPDATED*).
- General system events to log (e.g., *EVENT_CSP_CONFIG_UPDATED*, *EVENT_CSP_ERROR*)
- Audit signing key parameters (e.g., *KEY_ECC_PRIVATE*, *CURVE_SEC_P256_R1*, *SIG_ECDSA*)
- Counters on audit signing key or other resources (e.g., *COUNT_USAGE*)
- Fields to be included in the log message (e.g., *FIELD_SYSTEM_TIME*, *FIELD_USAGE_COUNTER*)
- Audit queue handling mode (e.g., *AUDIT_MODE_STRICT*, *AUDIT_MODE_OVERWRITE*)

The *AuditModule* shall collect events in an internal queue. Client Applications must actively fetch log messages using *Audit Operations*. The log messages are generated by the CSP based on the stored events. Depending on the CSP implementation, events may already be formatted and signed, or may only contain the required data, with log messages created and signed upon retrieval. In the latter case, the CSP shall ensure that all time-related and counter-related data in the log message matches what would have been used if the message had been generated at the time of the event.

Note: The CSP's audit feature is primarily designed to create secure audit logs for security purposes, not for error or application monitoring, though it may support such uses.

For further information, see section 2.2, Requirements, and section 6.12, Audit Module.

3.11.1 Creating Log Messages

The CSP shall sign each log message using an audit key managed per CSP Instance. This signature ensures that each log originates from this CSP Instance, ensuring the integrity and authenticity of the log data.

The *auditSigningKey* shall be configured by the CSP Admin through the *CSPAuditSettings* structure. As a standard CSP Resource, it may be imported, generated or derived using the *Key Management* operations provided by the CSP. If the *auditSigningKey* is not configured, fetching log messages shall fail with an *ERROR_ILLEGAL_CONFIG* [0x30C0] exception.

The CSP shall support the *auditSigningKey* parameters listed in Table 3-5, and the *Signature, Padding, Hash, Key Size, and Curve* Combinations specified in Table 6-16 if supported by the platform.

Only resources with *USAGE_AUDIT* in *STATE_OPERATIONAL* shall be accepted for signing audit logs.

The audit signing process shall utilize the CSP's *Signatures with Counters & Timestamps* functionality, allowing the CSP Admin to configure *Fields* to be included in each log message via the *CSPAttestationAlgorithms* structure, which is configured to the *auditSigningKey*. For example:

- *FIELD_USAGE_COUNTER*: Utilizes the *usageCounter* of the *auditSigningKey* as a log counter, which is included in each log message (dependent on *Usage Counter* functionality). As a unique identifier, this counter helps to detect unauthorized modifications or missing entries.
- *FIELD_SYSTEM_TIME*: Adds a current timestamp to each log message (dependent on *Time Synchronization* functionality). This timestamp, along with the usage counter as sequential identifier, enables the correlation of a log message with a specific real-world event.

Client Applications may provide additional input data to be included to the log message using the `audit.dequeueEvent` operation before it is signed by the CSP.

Note: The CSP does not serve as a secure real-time clock and only provides an indication of the time when the event occurred.

For further information, see section 6.14.1.1, Fields.

Table 3-5: Audit Signing Key Parameters

Parameter	Value	Description
Key Type	KEY_ECC_PRIVATE	Elliptic Curve Cryptography (ECC) private key.
ECC Domain Parameters	CURVE_SEC_P256_R1	NIST 256-bit curve.
Signature Algorithm	SIG_ECDSA	Elliptic Curve Digital Signature Algorithm (ECDSA) for signing.
Padding Algorithm	PAD_NULL	No padding is applied to the data.
Message Digest Algorithm	ALG_SHA_256	Message digest algorithm SHA-256.

3.11.2 Processing Log Messages

The CSP is responsible for creating the log messages, incorporating security features through *Fields* such as counter and timestamps, and signing them. However, the Client Application remains responsible for regularly fetching these log messages from the CSP Instance and handling their further processing, typically by transferring them to dedicated storage to ensure their availability for future audits or references.

Until the log messages are fetched, the CSP shall persistently store them as new log entries in an internal event queue. Client Applications may be notified of new events through the `listener.auditEventOccurred` callback, informing them to fetch the accumulated log entries from the CSP.

The Client Application should fetch events using the `audit.dequeueEvent` operation. During this process, the CSP shall create log messages by combining event-related data, application-specific input data and *Fields* configured by the CSP Admin. The CSP shall sign these log messages with the `auditSigningKey`. Once the log message is successfully generated, the event shall be removed from the audit event queue.

Note: Although the CSP creates numbered, timestamped and signed log messages, it does not process them further. The long-term storing and evaluation of log messages is specific to each use case and beyond the scope of this specification document.

3.11.3 Audit Event Queue

The CSP shall temporarily store events, including the event-related data (e.g., the affected `resourceId`) and *Fields* that must remain unchanged for the event (e.g., the timestamp), in persistent memory until fetched by Client Applications.

The maximum capacity of the audit event queue shall be subject to *Modularity*. Technically, an event queue capacity of one constitutes a correct implementation. This approach may suffice for straightforward auditing implementations, wherein, data being logged may be taken directly from the resources themselves rather than being copied to a separate event queue. The recommended audit queue size is 10. Client Applications may retrieve the remaining capacity using the `audit.getRemainingCapacity` operation.

Considering the limited storage capacity of SEs, the CSP Admin may manage this limitation by selecting from available *Audit Modes* using the `CSPAuditMode` structure:

- In `AUDIT_MODE_OVERWRITE`, older log messages are discarded to make space for new events when the audit event queue is full.

- In `AUDIT_MODE_STRICT`, the CSP halts when the audit queue is full, returning `ERROR_INVALID_INIT [0x40C0]` for any CSP API call that tries to add a new entry until the accumulated logs are fetched by the Client Application.

Note: Fields included in log messages are also stored in persistent memory within the audit event queue. However, `FIELD_USAGE_COUNTER`, when used as a log counter, shall not be persisted.

For further information, see section 6.12.1.1, Audit Modes, and section 6.12.3.4, Audit Listener.

3.11.4 Selecting Audit Events

The CSP Admin can select which *Audit Events* shall be logged by the CSP. The CSP categorizes them as system events and resource-specific events.

System events, configured through `CSPSystemEvent`, relate to the overall operation and state of the CSP and are not tied to a specific resource. They include:

- CSP start-up, such as following a power cycle, as specified for `EVENT_CSP_START`.
- Updates to the system clock, as specified for `EVENT_CSP_TIME_SET`.
- General error occurrences and exceptions, as specified for `EVENT_CSP_ERROR`.
- Updates or changes to the CSP configuration and system software, as specified for `EVENT_CSP_UPDATE_STARTED` and following.

Resource events, configured through `CSPResourceEvent`, cover cryptographic operations related tied to specific CSP Resources. They include:

- Resource modifications, as specified for `EVENT_OFFLOAD_IMPORTED`, `EVENT_RESOURCE_CLEARED`, `EVENT_KEY_GENERATED`, `EVENT_KEY_DERIVED`, `EVENT_PASSWORD_UPDATED`, etc.
- Cryptographic operations performed on resources, as specified for `EVENT_SECCHANNEL_ESTABLISHED`, `EVENT_SIGNATURE_CREATED`, `EVENT_CIPHER_DECRYPTED`, etc.
- Failed operations involving resources, as specified for `EVENT_PASSWORD_CHECK_FAILED`, `EVENT_PASSWORD_UPDATE_FAILED`, `EVENT_SECCHANNEL_AUTHENTICATION_FAILED`, `EVENT_SIGNATURE_VERIFICATION_FAILED`, etc.
- Changes in resource states, as specified for `EVENT_PASSWORD_BLOCKED`, `EVENT_TIMER_EXPIRED`, and `EVENT_COUNTER_EXHAUSTED`.

For further information, see section 6.12.1.2, Audit Events.

3.11.5 Limitations of Audit

3.11.5.1 Impact on Memory from Logging

The CSP's event logging involves writing data to the persistent memory of the Secure Element (SE). Since each log entry stored in the audit event queue requires a new write operation, frequent logging may increase the risk of memory degradation over time. This is an important consideration for systems that generate a large number log events. Developers should take this into account for systems that heavily rely on CSP logging.

For further information, see section 6.12.1.1, Audit Modes.

3.11.5.2 Audit is deactivated by Default

By default, no events are logged by the CSP. The CSP Admin must explicitly configure which events should be logged by selecting specific events of interest. Additionally, the CSP Admin is responsible for choosing *Fields*, such as a counter or timestamp, to be included in each log entry.

3.11.5.3 Limited Event Buffering Capacity

The CSP can only store a limited number of audit events at the same time, for example, fewer than 10; the exact number may be retrieved through the `audit.getRemainingCapacity` operation. If audit events are configured for logging by the CSP Admin but are not fetched by a Client Application, either older events will be lost or the CSP will halt with an exception depending on the supported *Audit Modes*.

4 ARCHITECTURE AND INSTANTIATION

4.1 Architecture

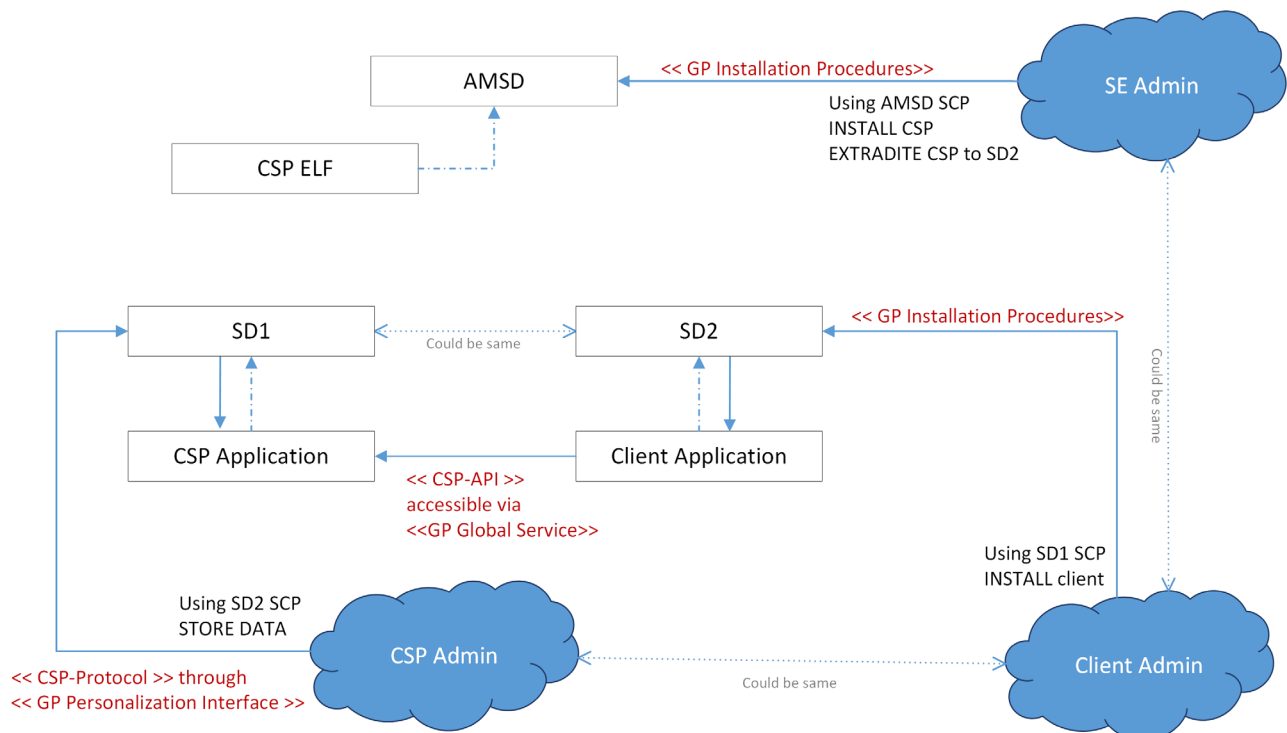
The Cryptographic Service Provider (CSP) specified in this document is based on Secure Element (SE) technology and utilizes GlobalPlatform Security Domains (SDs) to ensure isolation of the CSP's *Resource Management*. Figure 4-1 illustrates the architecture of the CSP Platform.

The CSP Platform contains a certified CSP Executable Load File (CSP ELF) that is associated with the Authorized Management Security Domain (AMSD) of the SE. CSP Applications dedicated to specific use cases are created as instances of the CSP Executable Module (CSP EM) within the CSP ELF, adhering to the GlobalPlatform Installation Procedures specified in [GP Card Spec]. This instantiation relies on fixed Application Identifiers (AIDs) for both the EM and ELF, as defined in Table 4-1, along with standardized installation parameters defined in Table 4-4.

The CSP Application offers administrative operations through the CSPAdminCommand, which utilizes the GlobalPlatform Personalization interface ([GP Card Spec]). This command, part of the *CSP Protocol*, enables CSP Admins to configure a CSP Application for a specific use case such as eID, payment or ticketing.

To support multiple use cases on the same SE, each use case requires a dedicated CSP Application, instantiated in its own SD. GlobalPlatform Secure Channel Protocols (SCPs) with SD-specific access keys enable multiple entities to use the same CSP Platform while ensuring the confidentiality of each other's data. This arrangement isolates CSP configurations and cryptographic secrets for different use cases.

Figure 4-1: Architecture of a CSP Platform



The CSP Admin configures cryptographic keys and algorithms for a CSP Instance. This instance is an object of `org.globalplatform.csp.api.CSP` ([GP CSP API]), which is part of the *CSP API*, and provides predefined cryptographic services with use-case-specific operations based on the CSP Configuration set up by the CSP Admin. Client Applications can retrieve the CSP Instance through the Global Service mechanism ([GP Card Spec]), using the *Global Service ID for the CSP* defined in Table 4-2.

Only Client Applications whose AIDs have been registered by the CSP Admin are granted access to the CSP Instance. The CSP Admin may configure additional checks, such as:

- Allowing only Client Applications that were verified during installation using the Data Authentication Pattern (DAP) ([GP Card Spec]), which is a mechanism to verify that the ELF is authentic.
- Checking that the AID of the SD of the calling Client Application matches the configured SD AID.
- Verifying that the Load File Data Block Hash (LFDBH) of the Client Application's ELF ([GP Card Spec]) matches the configured hash.

These checks are performed by the CSP when a Client Application requests to retrieve the CSP Instance from the CSP Global Service, ensuring that only authorized Client Applications can access the CSP API with the cryptographic configurations personalized for their use case.

The CSP Admin may register multiple Client Applications to one or more CSP Applications. Although not typical, a single Client Application may be registered to operate with multiple CSP Applications instantiated on the same platform.

Note: While the CSP Platform specified in this document is based on GlobalPlatform technologies, it is not limited to implementation as a Java Card applet. Other approaches are possible, such as Java Card OS extensions or native OS extensions.

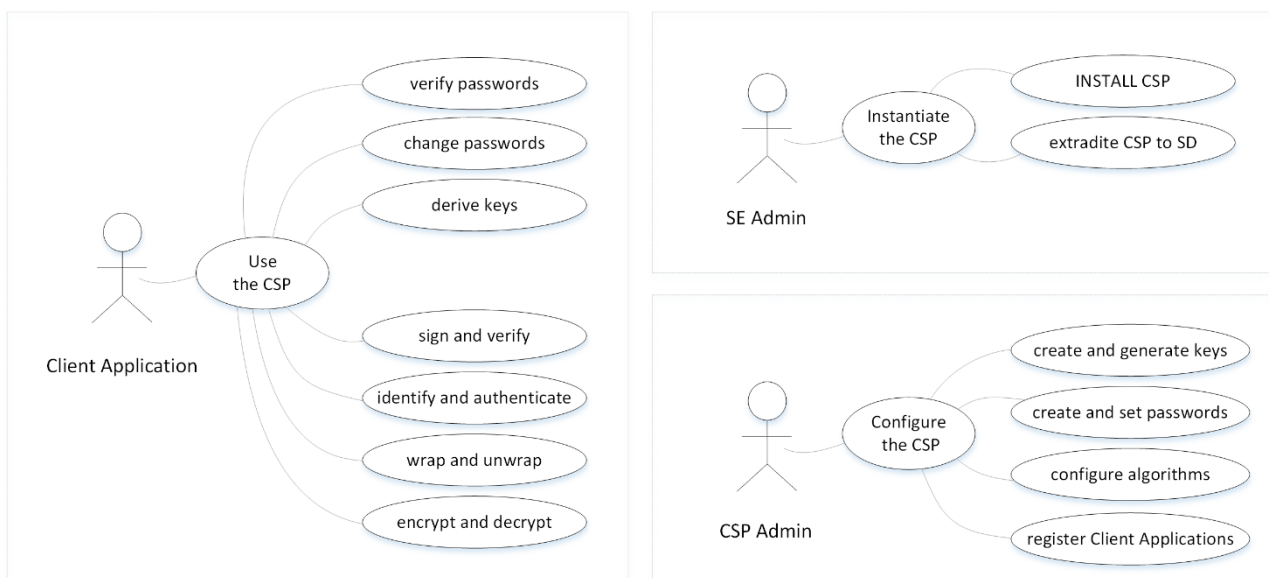
For further information, see section 1.5.4, Interfaces, section 3.2.1 Client Application Registration, and section 4.4 CSP Instantiation.

4.2 Role Model

This specification defines the entities interacting with the CSP, as illustrated in Figure 4-2:

- SE Admin: An off-card entity responsible for instantiating the CSP Application on the Secure Element.
- CSP Admin: An off-card entity that configures the CSP via the CSPAdminCommand of the *CSP Protocol*.
- Client Application: An on-card entity that utilizes predefined CSP services through the *CSP API*.

Figure 4-2: Roles



The CSP concept described in [TR-03181] defines additional entities related to the CSP:

- Admin Application: An on-card entity with access rights to configure the CSP.
- Off-card Client: An off-card entity that directly utilizes CSP services.

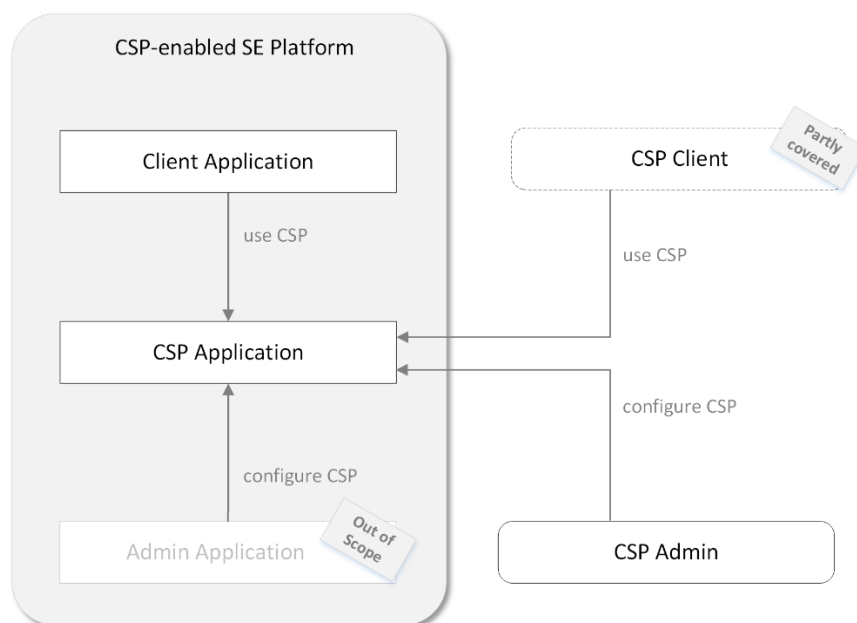
However, the Admin Application and off-card Clients are not fully covered by this document, as illustrated in Figure 4-3. The CSP Protocol provides a CSPClientCommand with a subset of CSP service operations that can be utilized by off-card Clients through the CSP's APDU Interface, but this command is optional and does not cover the entire CSP functionality. Furthermore, the CSP API does not define administrative methods or an access control mechanism for on-card Admin Applications.

Note 1: Vendor-specific CSP implementations may support Admin Applications by introducing a new administrative operation, e.g., CSPRegisterAdminApplication as part of the CSPAdminCommand, and converting configuration-related operations (e.g., CSPSetup, CSPCreateResource, CSPConfigureResource) from ASN.1 to Java Card. For example, they may add an `org.globalplatform.csp.api.AdminService` to the CSP API and implement access control so that only registered Admin Applications can retrieve this AdminService from the CSP Global Service. While this approach may be considered in a future version of this document, it is currently out of scope.

Note 2: A future version of this specification may extend the CSPClientCommand by converting all CSP API operations to ASN.1 structures of the CSP Protocol.

For further information, see section 7, CSP Protocol, and section 8, CSP API.

Figure 4-3: Roles and Applications Interacting with the CSP



4.3 Deployment

The SE Admin, authorized to access the Authorized Management Security Domain (AMSD), is responsible for instantiating the CSP Application from the CSP ELF. If the CSP ELF is not present, the SE Admin may load it, provided the platform meets the technical and certification requirements of the CSP implementation.

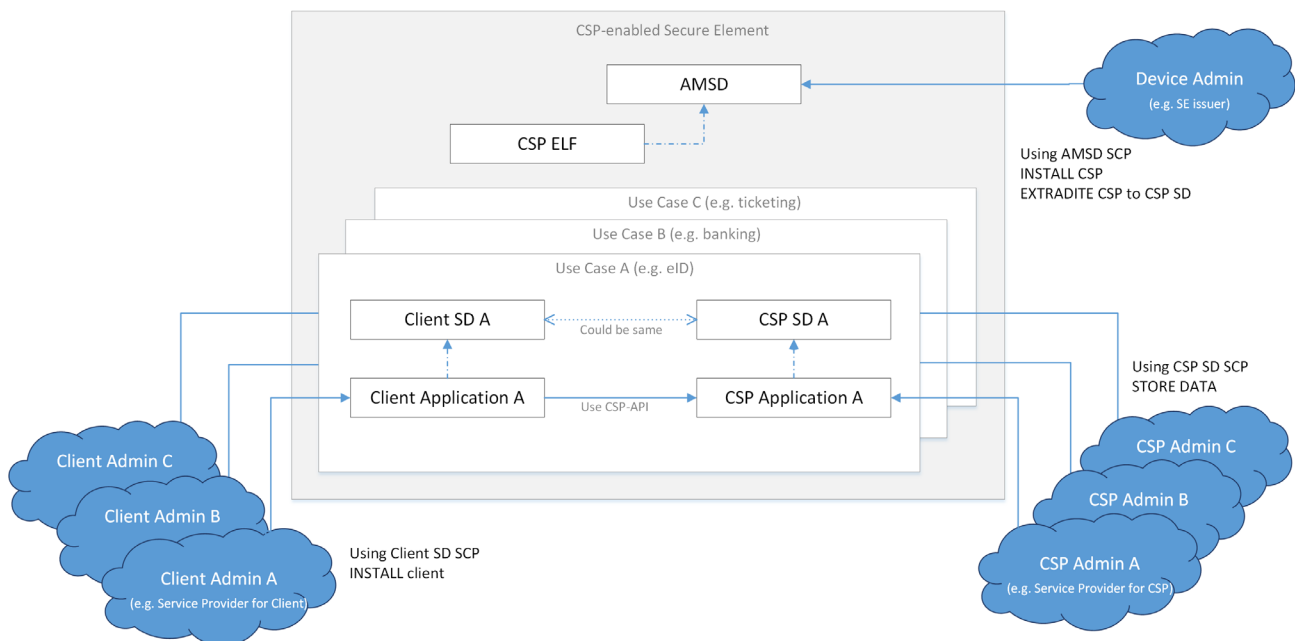
A SE Admin may instantiate multiple CSP Applications on same Secure Element (SE). The available memory on the SE should be the sole limiting factor concerning the maximum number of CSP Applications supported.

Depending on the deployment model set by the SE owner, the Client Application can be installed either by a Client Admin acting through a SD with Authorized Management privileges or by the SE Admin. Both the Client and CSP Application designed for the same use case can be associated with either the same Security Domain (SD) or separate ones, as shown in Figure 4-4.

Each CSP Application serves exactly one use case such as eID, payment or ticketing. Multiple CSP Applications are used to separate cryptographic configurations from each other, ensuring support for different use cases on the same SE without any interference.

A CSP Admin, with access to the SD of the CSP, configures the CSP by sending Store Data APDUs to the CSP Application through the GlobalPlatform Personalization interface. This configuration involves creating or importing cryptographic keys, setting up algorithms and general CSP setup, as well as registering Client Applications.

Figure 4-4: CSP Deployment Diagram



4.4 Instantiation

The CSP shall follow the GlobalPlatform Installation Procedures specified in [GP Card Spec] using fixed AIDs for its binary packages and the fixed installation parameters specified in this section. This approach ensures a consistent instantiation of the CSP Application, regardless of the CSP implementation vendor.

The CSP Application can be pre-installed on the Secure Element or installed post-issuance.

For further information, see section 4.1, Architecture.

4.4.1 AIDs of the CSP

The CSP shall use the Application Identifiers (AIDs) for the Executable Load File (ELF), the Executable Module (EM) and the Java Card packages as specified in Table 4-1.

The AID of the CSP Application instance may be chosen by the SE Admin.

Table 4-1: AIDs of the CSP

Component	AID	Description
CSP ELF AID	A0 00 00 01 51 43 53 50	Fixed AID for the CSP Application's Executable Load File (ELF).
CSP EM AID	A0 00 00 01 51 43 53 50 00	Fixed AID for the CSP Application's Executable Module (EM).
CSP Package AID org.globalplatform.csp	A0 00 00 01 51 09	Fixed AID for the Java package org.globalplatform.csp.
CSP Package AID org.globalplatform.csp.api	A0 00 00 01 51 0A	Fixed AID for the Java package org.globalplatform.csp.api.
CSP Applications	any	AIDs for specific CSP Application instances; chosen by the SE Admin.

4.4.2 Global Service ID for the CSP

The CSP shall implement a Global Service according to [GP Card Spec] using the service name listed in Table 4-2 to enable registered Client Applications accessing cryptographic operations of the CSP API. Client Applications shall use the AID of the CSP Application instance to retrieve the Global Service of the CSP. Thus, the CSP does not need to register its Global Service to the OPEN.

For further information, see section 8.2, Retrieve CSP Instance.

Table 4-2: Global Service ID of the CSP

Service	Service Name	Description
CSP	0x8B00	Service name of the Global Service provided by the CSP, see [GP Card Spec] section 8.1.3.

4.4.3 Privileges Required by the CSP

A CSP shall exclusively use the GlobalPlatform privileges specified in Table 4-3. It is expressly prohibited for the CSP to require additional privileges; the listed privileges shall be sufficient for the proper operation of the CSP.

Table 4-3: GlobalPlatform Privileges Required by the CSP

Privilege	Privilege Number	Description
Global Service	15	The CSP requires the Global Service privilege to provide a Global Service through <code>org.globalplatform.GPSystem.getService(..)</code> .

4.4.4 CSP Install Parameters

A CSP shall exclusively use the GlobalPlatform install parameters specified in Table 4-4. It is expressly prohibited for the CSP to require additional install parameters; the listed install parameters shall be entirely sufficient for the proper operation of the CSP.

Sample 4-1: CSP Install Parameters

```
-- CSP Install Parameters
C9 00 EF 04 CB 02 8B 00
```

Table 4-4: CSP Install Parameters

Position (Start Byte)	Value (Hex)	Description
0	'C9'	Tag for Application Specific Install Parameters.
1	'00'	Length of the Application Specific Parameters.
2	'EF'	Tag for System Specific Parameters.
3	'04'	Length of the System Specific Parameters.
4	'CB'	Tag for Global Service Parameters.
5	'02'	Length of the Global Service Parameters.
6	'8B 00'	Global Service Identifier of the CSP.

4.4.5 CSP INSTALL Command

The CSP Application shall be instantiated using the CSP INSTALL command specified in Table 4-5. This command, compliant with [GP Card Spec], instantiates a new CSP Application from an existing CSP Executable Load File (ELF) present on the Secure Element.

Sample 4-2: CSP INSTALL Command

```
"INSTALL [for install and make selectable] command to create a CSP instance"
80 E6 0C 00 <LC>
08 A0 00 00 01 51 43 53 50
09 A0 00 00 01 51 43 53 50 00
<L1> <CSPAID>
03 00 01 00
08 C9 00 EF 04 CB 02 8B 00
```

```
00 00;
```

A CSP Application instantiated with this command shall be fully operable without the need for additional installation parameters. An appropriate instance-AID of the instantiated CSP Application must be chosen by the CSP Admin. To avoid AID conflicts, it is highly recommended that each CSP Admin uses instance-AIDs starting with its respective Registered Application Provider Identifier (RID).

Note: CSP INSTALL [for load] is out of scope for this document and might be vendor specific.

Table 4-5: CSP INSTALL [for install and make selectable] Command

Position (Start Byte)	Value (Hex)	Description
1	'80 E6 0C 00 <LC>'	Class Byte (CLA) and Instruction Byte (INS) mark this as a GlobalPlatform INSTALL command ([GP Card Spec]). The parameters P1 and P2 indicate that the command is for installing an Executable Module and immediately making it selectable. The command is followed by a placeholder for the length of the subsequent data field, with $LC = L1 + L2 + 40$.
6	'08 A0 00 00 01 51 43 53 50'	Length and AID of the CSP's Executable Load File to be installed.
15	'09 A0 00 00 01 51 43 53 50 00'	Length and AID of the CSP's Executable Module to be installed.
25	<L1> <CSPAID>	A placeholder for the length and AID identifying the CSP Application instance (any AID chosen by the CSP Admin).
(26+L1)	'03 00 01 00'	Length and privileges required by the CSP.
(30+L1)	'08 C9 00 EF 04 CB 02 8B 00'	Length and install parameters required by the CSP.
(39+L1)	<L2> <TOKEN>	A placeholder for the length and token for Delegated Management (DM). Is '00 00' if DM is not used.

4.5 Lifecycle

The CSP Application shall follow the GlobalPlatform application lifecycle states, and its Security Domain shall adhere to the Security Domain lifecycle states as specified in [GP Card Spec]. For operational use:

- The CSP Application must be in the SELECTABLE state.

In this state, the CSP Application becomes configurable, allowing the creation of cryptographic resources such as keys and passwords, and the setup of algorithms and other configurations. However, a newly installed CSP Application remains inaccessible to Client Applications until the AIDs of the Client Applications are registered within the CSP Instance and the CSP Instance is activated, as specified for Figure 5-1 *System Lifecycle*.

When deleted, the CSP shall ensure the complete and secure erasure of the CSP Instance's configuration, including all cryptographic keys, passwords and other secrets.

5 CORE MODULES

5.1 System Module

The CSP shall offer two distinct operation modes to support *Resource Management*, defined in this section: a configuration mode that allows CSP Admins to set up cryptographic resources and algorithms, and an operational mode in which Client Applications can utilize these resources.

For further information, see section 3.1, Resource Management.

5.1.1 System Definitions

5.1.1.1 Operation Modes

The CSP shall support the operation modes listed in Table 5-1. The selected `cspMode` defines if the CSP operates either in configuration or in operational mode.

The CSP Admin may switch the `CSPMode` using the `CSPActivate` and `CSPDeactivate` commands. The current active mode is part of the `CSPSettings` structure.

Table 5-1: CSP Operation Modes

Mode	Value (Hex)	Value (Int)	Description
CSP_MODE_CONFIGURATION	0x00	0	<p>In configuration mode, the CSP Admin can utilize the <code>CSPAdminCommand</code> to configure the CSP Instance.</p> <p>In this mode, the CSP shall deny any attempts from Client Applications to utilize cryptographic services of the CSP with <code>ERROR_ILLEGAL_CONFIG [0x3001]</code>.</p> <p>Note: This is the default mode when a CSP Application is initially instantiated.</p>
CSP_MODE_OPERATIONAL	0x01	1	<p>In operational mode, Client Applications can use cryptographic services provided by the CSP.</p> <p>In this mode, the CSP shall deny any attempts to change the CSP Configuration via the <code>CSPAdminCommand</code> with <code>ERROR_ILLEGAL_CONFIG [0x3005]</code>.</p>

5.1.1.2 Modules

The CSP may support the modules listed in Table 5-2; the exact list shall be subject to *Modularity*.

The CSP Admin may detect if a module is supported using the `CSPEnforce` command.

For more information, see section 8, CSP API.

Table 5-2: CSP Modules

Module	Value (Hex)	Value (Int)	Description
CipherModule	0x01	1	<p>Provides encryption and decryption functionalities as part of <i>Cipher and Signatures</i>, specified in the <i>Cipher Module</i>.</p> <ul style="list-style-type: none"> CSP API: <code>org.globalplatform.csp.api.CipherService</code>

Module	Value (Hex)	Value (Int)	Description
SignatureModule	0x02	2	Provides signature generation and verification functionalities as part of <i>Cipher and Signatures</i> , specified in the <i>Signature Module</i> . <ul style="list-style-type: none"> CSP API: <code>org.globalplatform.csp.api.SignatureService</code>
TransformModule	0x03	3	Provides <i>Encryption Transformations</i> by transferring the encryption from one key and algorithm to another in a single atomic step, thereby preventing Client Applications from accessing the plain data, as specified in the <i>Transform Module</i> . <ul style="list-style-type: none"> CSP API: <code>org.globalplatform.csp.api.TransformService</code>
SecureChannelModule	0x04	4	Provides <i>Secure Messaging</i> authentication and communication, as specified in the <i>Secure Channel Module</i> . <ul style="list-style-type: none"> CSP API: <code>org.globalplatform.csp.api.SecureChannelService</code>
ConfidentialModule	0x05	5	Provides <i>Confidential Data Transfer</i> as an extension of the <i>SecureChannelModule</i> , facilitating the secure transfer of confidential data by converting session-layer encryption to storage-layer encryption, and vice-versa, in a single atomic step, as specified in the <i>Confidential Data Transfer Module</i> . <ul style="list-style-type: none"> CSP API: <code>org.globalplatform.csp.api.ConfidentialDataTransferService</code> Dependencies: <i>SecureChannelModule</i>
AttestationModule	0x06	6	Provides computing <i>Attestations</i> to verify the identity and authenticity of Public Keys, the CSP Instance or the SE platform, as specified in the <i>Attestation Module</i> . <ul style="list-style-type: none"> CSP API: <code>org.globalplatform.csp.api.AttestationService</code> CSP Protocol: <i>CSPSystemAttestation</i>
KeyModule	0x07	7	Provides <i>Key Management</i> operations, such as key generation, key derivation and key agreement, as specified in the <i>Key Module</i> . <ul style="list-style-type: none"> CSP API: <code>org.globalplatform.csp.api.KeyService</code> CSP Protocol: <i>CSPGenerateKey</i>, <i>CSPComputePublicKey</i>, <i>CSPDeriveKey</i>
CertificateModule	0x08	8	Provides <i>Certificate Management</i> , as specified in the <i>Certificate Module</i> . <ul style="list-style-type: none"> CSP API: <code>org.globalplatform.csp.api.CertificateService</code> CSP Protocol: partially via <i>CSPSetValue</i>
PasswordModule	0x09	9	Provides <i>Password Management</i> , including retry counter and a blocking mechanism, as specified in the <i>Password Module</i> . <ul style="list-style-type: none"> CSP API: <code>org.globalplatform.csp.api.PasswordService</code> CSP Protocol: partially via <i>CSPSetValue</i>
CounterModule	0x0A	10	Extends all other modules by providing <i>Counter and Limits</i> to increment and manage large counters, such as built-in usage counters, as specified in the <i>Counter Module</i> . <ul style="list-style-type: none"> CSP API: <code>org.globalplatform.csp.api.CounterService</code>
TimeModule	0x0B	11	Extends all other modules by providing <i>Timer and Time Management</i> to enforce time-based limits, such as validity date and authentication timeouts, as specified in the <i>Time Module</i> . <ul style="list-style-type: none"> CSP API: <code>org.globalplatform.csp.api.TimeService</code> CSP Protocol: <i>CSPSetTime</i>

Module	Value (Hex)	Value (Int)	Description
AuditModule	0x0C	12	Extends all other modules by providing <i>Secure Auditing</i> to create integrity-protected audit log messages for events occurring within a CSP Instance, as specified in the <i>Audit Module</i> . • CSP API: <code>org.globalplatform.csp.api.AuditService</code>
OffloadingModule	0x0D	13	Provides offloading functionality to import and export Resources, as specified in the <i>Offloading Module</i> . • CSP API: <code>org.globalplatform.csp.api.OffloadingService</code>
FieldModule	0x0E	14	The <i>Field Module</i> extends the <i>Attestation Module</i> and the <i>Audit Module</i> to support configuring signature fields to be added to attestation data and/or log messages, e.g., to create <i>Signatures with Counters & Timestamps</i> .
PolicyModule	0x0F	15	The <i>Policy Module</i> provides an additional constraint-based access control mechanism that extends the <i>Access Module</i> and supports the evaluation of <i>Policies</i> .
RandomDataModule	0x10	16	The <i>Random Data Module</i> provides support for generating random data.

5.1.1.3 System Attestations

The CSP shall support the system attestation types listed in Table 5-3.

The Client Application may select the attestationType when invoking the `att.computeAttestation`.

The CSP Admin or the off-card Client may invoke system attestations using the `CSPSystemAttestation` command.

For more information, see section 3.8, Attestations, and section 6.6.1.1, Resource Attestation Types.

Table 5-3: System Attestations Types

System Attestation	Value (Hex)	Value (Int)	Description
ATTESTATION_PLATFORM	0x01	1	The <i>Platform Attestation</i> verifies the identity and authenticity of the CSP-enabled Secure Element by signing platform information using <code>org.globalplatform.AuthoritySignature</code> [GP API] from the GlobalPlatform Card Application Security Domain (CASD) mechanism ([GP Amd A]). Technical Details: • Format: <code>CSPPlatformAttestation</code>
ATTESTATION_CONFIG	0x02	2	The <i>Config Attestation</i> verifies the identity and authenticity of a CSP Instance and its use-case-specific configuration. The <code>configAttestationKey</code> is imported by the CSP Admin and used to sign information of a specific CSP Instance. Technical Details: • Format: <code>CSPConfigAttestation</code>

5.1.1.4 Core System Events

The AuditModule may log system-related *Audit Events* listed in Table 5-4 for *Secure Auditing*; the exact list shall be subject to *Modularity*. Event-specific data listed in the table shall be stored according to the configured auditMode. These events are specific to a CSP Instance and are not tied to any particular resource.

The CSP Admin may configure these systemEvents to be logged using the CSPSystemEvent structure.

The CSP Admin may detect if an event type is supported using the CSPAuditSupport structure.

For further information, see section 3.11.4, Selecting Audit Events, and section 6.12.1.2, Audit Events.

Table 5-4: Core System Events for Auditing

Event Type	Value (Hex)	Value (Int)	Description	Event Category
EVENT_CSP_START	0x0001	1	CSP Application starts. Logged after each start of the CSP Application, e.g. after system reboot or when the CSP Application starts after re-installation. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Event Data: - 	System Event
EVENT_CSP_UPDATE_STARTED	0x0002	2	Start of a CSP software update. Logged when a CSP software update begins. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Event Data: CSPEventDataUpdateCSP 	System Event
EVENT_CSP_UPDATE_FINISHED	0x0003	3	Completion of a CSP software update. Logged when a CSP software update has finished. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Event Data: CSPEventDataUpdateCSP 	System Event
EVENT_CSP_CONFIG_UPDATED	0x0004	4	Successful update of the CSP configuration. Logged when the CSP Instance configuration is modified and activated. It is triggered after the CSPActivate command is invoked, but only if there were prior modifications to the CSPConfiguration structure. This includes activating the initial configuration. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Event Data: CSPEventDataUpdateConfig 	System Event
EVENT_CSP_ERROR	0x0005	5	Occurrence of an unspecified error. Logs every exception thrown by the CSP. The log message contains the reason from the <code>javacard.framework.CardException ([JCAP])</code> . Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Event Data: CSPEventDataGeneralError 	System Event

5.1.1.5 Error Modes

The CSP may support the error modes listed in Table 6-138; the exact list shall be subject to *Modularity*. The selected error mode defines how the CSP shall handle exceptions.

The CSP Admin may select the errorMode using the CSPErrorMode structure. The currently configured error mode is part of the CSPSettings structure.

The CSP Admin may detect if an error mode is supported using the CSPEnforce command.

Table 5-5: CSP Error Modes

Error Mode	Value (Hex)	Value (Int)	Description
ERROR_MODE_BASIC	0x00	0	The CSP shall only use basic <i>Error Types</i> specified in Table 5-6. The CSP shall not use the detailed CSP error codes specified for ERROR_MODE_DETAILED. Note: This is the standard mode when the platform does not support detailed error codes.
ERROR_MODE_DETAILED	0x01	1	The CSP shall use the error codes specified in: <ul style="list-style-type: none"> • Table 5-7 Core Error Codes • Table 6-7 Cipher Error Codes • Table 6-18 Signature Error Codes • Table 6-24 Transform Error Codes • Table 6-39 Secure Channel Error Codes • Table 6-46 Confidential Data Transfer Error Codes • Table 6-52 Attestation Error Codes • Table 6-69 Key Error Codes • Table 6-78 Certificate Error Codes • Table 6-86 Password Error Codes • Table 6-98 Counter Error Codes • Table 6-113 Time Error • Table 6-121 Audit Error Codes • Table 6-130 Offloading Error Codes

5.1.1.6 Error Types

The CSP shall use the `org.globalplatform.csp.CSPException` class ([GP CSP API]) with the error types listed in Table 5-6 for error handling in both the CSP Admin Protocol and the CSP API.

Table 5-6: CSP Error Types

Error Type	Reason	Description
ERROR_ILLEGAL_BUFFER	0x1000	Illegal buffer: Thrown when a provided buffer is null, out of bounds, or inaccessible in the caller's context.
ERROR_ILLEGAL_VALUE	0x2000	Illegal input parameter: Thrown when one or more input parameters are out of the allowed bounds.
ERROR_ILLEGAL_CONFIG	0x3000	Illegal configuration: Thrown when a configuration is uninitialized or cryptographically misconfigured.
ERROR_INVALID_INIT	0x4000	Not initialized: Thrown when a CSP service is not properly initialized.
ERROR_NOT_ALLOWED	0x5000	Not allowed: Thrown when an operation is not allowed due to the current state or configuration.
ERROR_ILLEGAL_USE	0x6000	Illegal use: Thrown when data cannot be processed for cryptographic operations, such as non-block-aligned input.
ERROR_NOT_SUPPORTED	0x8000	Feature or algorithm not supported: Thrown when an algorithm or feature is not supported by the platform.

5.1.1.7 Core Error Codes

If `ERROR_MODE_DETAILED` is supported and activated, the CSP shall use the error codes listed in Table 5-7 as reasons for exceptions.

For further information, see section 5.1.1.6, Error Types.

Table 5-7: Core Error Codes

Reason	Description
0x1001	Output buffer null: The output buffer is required but is <code>null</code> .
0x1002	Input buffer null: The input buffer is required but is <code>null</code> .
0x1003	Null buffer with length: Buffer is <code>null</code> , but length is not 0.
0x1004	Illegal buffer index: Buffer offset or length out of bounds.
0x1005	Illegal buffer context: Buffer not accessible in the caller's context (e.g., not a global array).
0x1006	Missing integrity protection: A provided buffer is not supporting <i>Sensitive Arrays</i> .
0x1007	Integrity check failed: <i>Sensitive Arrays</i> verification detected inconsistencies.
0x2001	Resource does not exist: The resource identifier provided does not refer to any existing resource.
0x2002	Resource already exists: The resource identifier provided is already existing.
0x2003	Not a CSP Instance: The provided service context is not an instance of <code>org.globalplatform.csp.api.CSP</code> .
0x3001	CSP not activated: The CSP is in <code>CSP_MODE_CONFIGURATION</code> and cannot be used by Client Applications.
0x3002	Missing resource: The resource that corresponds to an CSP-Instance-specific configuration (e.g., the config attestation key) does not exist.
0x3003	Resource not initialized: Resource is in <code>STATE_UNINITIALIZED</code> and cannot be used for cryptographic or security-related operations.
0x3004	Resource already initialized: Resource is not in <code>STATE_UNINITIALIZED</code> and cannot be re-initialized or removed.
0x3005	CSP already activated: The CSP is in <code>CSP_MODE_OPERATIONAL</code> and cannot be configured by the CSP Admin.
0x3006	Missing platform attestation key: There is no CASD attestation key configured. The CASD key is required by the <i>Platform Attestation</i> .
0x3007	Inconsistent platform attestation config: The CASD attestation has inconsistent cryptographic configurations. The CASD key is required by the <i>Platform Attestation</i> .
0x3008	Missing config attestation key: There is no CSP Instance-specific config attestation key configured.
0x3009	Inconsistent policy config: Owner resource, constraining resource, or additional data are incompatible with the policy type.
0x5001	Not allowed: The AID of the Client Application is unknown
0x5002	Not allowed: The AID of the invoking Application does not match the AID of the GPRegistryEntry.
0x5003	Not allowed: Illegal Security Domain AID of the Client Application.
0x5004	Not allowed: Illegal LFDBH of the Client Application.
0x5005	Not allowed: Missing DAP verification for the Client Application.

Reason	Description
0x5006	Not allowed: A <i>Client Authentication</i> is required to use the operation.
0x5007	Not allowed: At least one resource used is missing the ACCESS_USE right for this operation.
0x5008	Not allowed: At least one resource used is missing the ACCESS_SETUP right for this operation.
0x5009	Not allowed: At least one resource used is missing the ACCESS_CLEAR right for this operation.
0x500A	Not allowed: At least one resource used is missing the ACCESS_MOVE right for this operation.
0x500B	Policy violation: A policy failed.
0x6001	Illegal resource import: The data provided is not compatible to the resource it shall be imported to.
0x8001	Unsupported: The protocol version is not supported.
0x8002	Unsupported: The resource type is not supported.
0x8003	Unsupported: The policy mode is not supported.
0x8004	Unsupported: The policy type is not supported and POLICY_MODE_STRICT is set.

5.1.2 System Configuration

The CSP shall support the configuration parameters defined in Table 5-8.

The CSP Admin may retrieve the CSPConfiguration using the CSPGetConfiguration command.

Table 5-8: CSP Configuration Parameters

Parameter	Description	Parameter Type
cspMode	Indicates whether this CSP Instance is currently activated and usable by Client Applications, as specified for the <i>CSP Operation Modes</i> . Technical Details: <ul style="list-style-type: none"> • Data type: CSPMode • Administration: CSPActivate, CSPDeactivate • Initial value: CSP_MODE_CONFIGURATION (0x00) 	CSP Instance Parameter
configName	A custom name or identifier of the use-case-specific CSP Configuration. It is specified by the CSP Admin and used in the <i>Config Attestation</i> . Technical Details: <ul style="list-style-type: none"> • Data type: CSPConfigName • Administration: CSPSettings • Initial value: filled with zeros • Event: CSPEventDataUpdateConfig 	CSP Instance Parameter
configVersion	A custom version number of the use-case-specific CSP Configuration. It is specified by the CSP Admin and used in the <i>Config Attestation</i> . Note: To ensure accurate version control and tracking, it is recommended to increment this version number each time the CSP Configuration is changed and activated. Technical Details: <ul style="list-style-type: none"> • Data type: CSPConfigVersion • Initial value: 0x0000 • Administration: CSPActivate, CSPSettings • Fields: FIELD_CSP_CONFIG_VERSION • Events: CSPEventDataUpdateConfig 	CSP Instance Parameter

Parameter	Description	Parameter Type
configAttestationKey	<p>Identifier of the config attestation key. It is imported by the CSP Admin and is used to create the <i>Config Attestation</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPResourceId • Attestation: ATTESTATION_CONFIG • Algorithm: CSPAttestationAlgorithms • Administration: CSPSettings • Initial value: null 	CSP Instance Parameter
errorMode	<p>The currently active error mode, selected from the available <i>Error Modes</i>. It defines how the CSP shall handle errors.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPErrorMode • Initial value: 0 (ERROR_MODE_BASIC) • Administration: CSPSettings 	CSP Instance Parameter
clientApplications	<p>The list of Client Applications registered to this CSP Instance that are allowed to utilize the cryptographic services provided by the CSP.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: SET OF CSPAID • Initial value: empty list • Administration: CSPRegisterClient 	CSP Instance Parameter
resources	<p>This includes registered Client Applications, initialized resources, cryptographic algorithms, access control configurations and general system settings.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: SET OF CSPResource • Initial value: empty list • Administration: CSPCreateResource, CSPConfigureResource 	CSP Instance Parameter
cspProtocolVersion	<p>Version of the <i>CSP Protocol</i>. It is included in each command of the CSP Protocol and is used to ensure backward compatibility by checking <i>API Level Compliance</i> with each command sent.</p> <p>The current version of the <i>CSP Protocol</i> is used in the <i>Platform Attestation</i> and the <i>Config Attestation</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPProtocolVersion • Retrieved: CSPPlatform • Events: EVENT_CSP_UPDATE_STARTED, EVENT_CSP_UPDATE_FINISHED 	Method Parameter

5.1.3 System Operations

The CSP shall implement the system operations listed in Table 5-9 and shall implement module-specific operations required by the *Modules* it supports. It may also implement additional convenience operations not listed in this specification.

Table 5-9: CSP Operations

Operation	Description
CSPEnforce	<p>Detect if CSP-specific modules, types or algorithms are supported by this CSP.</p> <ul style="list-style-type: none"> • CSP Protocol: CSPEnforce

Operation	Description
CSPRegisterClient	Register the AID of a Client Application to this CSP Instance. <ul style="list-style-type: none"> CSP Protocol: CSPRegisterClient
CSPUnregisterClient	Unregister a Client Application from the CSP Instance. <ul style="list-style-type: none"> CSP Protocol: CSPUnregisterClient
CSPCreateResource	Create a new use-case-specific resource, such as keys, for this CSP Instance. <ul style="list-style-type: none"> CSP Protocol: CSPCreateResource
CSPDestroyResource	Securely wipe a resource. <ul style="list-style-type: none"> CSP Protocol: CSPDestroyResource
CSPConfigureResource	Change the cryptographic or security-related parameters of a resource. <ul style="list-style-type: none"> CSP Protocol: CSPConfigureResource
CSPSetup	Set up the general configuration for the CSP Instance. <ul style="list-style-type: none"> CSP Protocol: CSPSetup
CSPActivate	Change the CSP Instance to CSP_MODE_OPERATIONAL. <ul style="list-style-type: none"> CSP Protocol: CSPActivate
CSPDeactivate	Change the CSP Instance to CSP_MODE_CONFIGURATION. <ul style="list-style-type: none"> CSP Protocol: CSPDeactivate
CSPSystemAttestation	Compute <i>System Attestations</i> . <ul style="list-style-type: none"> CSP Protocol: CSPSystemAttestation
csp.getConfigVersion	Retrieve the version of the CSP Configuration. This version is set by the CSP Admin using the CSPSettings structure within the CSPSetup command. <ul style="list-style-type: none"> CSP API: org.globalplatform.csp.api.SystemService.getConfigVersion(..)
csp.getConfigName	Retrieve the name of the CSP Configuration. This name is set by the CSP Admin using the CSPSettings structure within the CSPSetup command. <ul style="list-style-type: none"> CSP API: org.globalplatform.csp.api.SystemService.getConfigName(..)

5.1.3.1 Access Restrictions on System Operations

The CSP shall enforce the restrictions for *System Operations* listed in Table 5-10 for all *Access Rights*, *Usage Types*, and *Resource States* supported by the platform. Unauthorized or illegal operation invocations shall be rejected and handled according to the errorMode configured.

Table 5-10: CSP Operations Access Restrictions

Operation	Right Required	Usage Required	State Required	Other Restrictions
CSPEnforce	-	-	-	<ul style="list-style-type: none"> Secured via CSP SD channel
CSPRegisterClient	-	-	-	<ul style="list-style-type: none"> Secured via CSP SD channel
CSPUnregisterClient	-	-	-	<ul style="list-style-type: none"> Secured via CSP SD channel
CSPCreateResource	-	-	-	<ul style="list-style-type: none"> Secured via CSP SD channel
CSPDestroyResource	-	-	UNINITIALIZED	<ul style="list-style-type: none"> Secured via CSP SD channel
CSPConfigureResource	-	-	-	<ul style="list-style-type: none"> Secured via CSP SD channel

Operation	Right Required	Usage Required	State Required	Other Restrictions
CSPSetup	-	-	-	• Secured via CSP SD channel
CSPActivate	-	-	-	• Secured via CSP SD channel
CSPDeactivate	-	-	-	• Secured via CSP SD channel
csp.getConfigVersion	-	-	-	

5.1.3.2 API Level Compliance

The CSP shall support `PROTOCOL_VERSION_1` of the *CSP Protocol* and may support previous versions listed in Table 5-11.

The required `cspProtocolVersion` is included in all CSP Protocol commands.

Any command invocation using an unsupported protocol version shall fail with `ERROR_NOT_SUPPORTED [0x8001]`.

Table 5-11: CSP API Level Compliance

API Version	Value	Description
<code>PROTOCOL_VERSION_1</code>	1	Initial version of the CSP Protocol. Compatible with CSP API version 1.0.x.

5.1.3.3 Sensitive Results Checks

The CSP shall implement `org.globalplatform.csp.api.CSPService.assertSensitiveResult ([GP CSP API])` for the *Modules* supported and shall store security-relevant results in transient memory, as specified in Table 5-12.

To detect unauthorized modification, Client Applications should verify operation results where integrity checks are recommended using the `assertSensitiveResult` operation.

Table 5-12: CSP Assert Sensitive Results Operations

Operation	Description
assertSensitiveResult	<p>Asserts that a CSP-internally stored result was computed without any abnormalities and matches the provided value.</p> <ul style="list-style-type: none"> CSP API: <code>org.globalplatform.csp.api.CSPService.assertSensitiveResult(..)</code> <p>Implemented by:</p> <ul style="list-style-type: none"> Table 5-22 for <i>Sensitive Results computed by Resource Operations</i> Table 6-11 for <i>Sensitive Results Computed by Cipher Operations</i> Table 6-22 for <i>Sensitive Results Computed by Signature Operations</i> Table 6-27 for <i>Sensitive Results computed by Transform Operations</i> Table 6-43 for <i>Sensitive Results Computed by Secure Channel Operations</i> Table 6-49 for <i>Sensitive Results computed by Confidential Data Transfer Operations</i> Table 6-56 for <i>Sensitive Results Computed by Attestation Operations</i> Table 6-73 for <i>Sensitive Results Computed by Key Operations</i> Table 6-82 for <i>Sensitive Results Computed by Certificate Operations</i> Table 6-90 for <i>Sensitive Results computed by Password Operations</i> Table 6-102 for <i>Sensitive Results computed by Counter Operations</i> Table 6-117 for <i>Sensitive Results computed by Time Operations</i> Table 6-125 for <i>Sensitive Results Computed by Audit Operations</i> Table 6-134 for <i>Sensitive Results Computed by Offloading Operations</i>

5.1.3.4 Sensitive Arrays

The CSP shall provide integrity protection for sensitive arrays used to exchange data between Client Application and CSP. It shall invoke `javacard.framework.SensitiveArrays.assertIntegrity(..)` ([JCAPI]) on the input and output array parameters listed in the following tables:

- Table 6-12 for *Sensitive Arrays Required for Cipher Operations*
- Table 6-23 for *Sensitive Arrays Required for Signature Operations*
- Table 6-28 for *Sensitive Arrays required for Transform Operations*
- Table 6-44 for *Sensitive Arrays Required for Secure Channel Operations*
- Table 6-57 for *Sensitive Arrays Required for Attestation Operations*
- Table 6-74 for *Sensitive Arrays Required for Key Operations*
- Table 6-83 for *Sensitive Arrays Required for Certificate Operations*
- Table 6-91 for *Sensitive Arrays Required for Password Operations*
- Table 6-103 for *Sensitive Arrays Required for Counter Operations*
- Table 6-118 for *Sensitive Arrays Required for Time Operations*
- Table 6-126 for *Sensitive Arrays Required for Audit Operations*
- Table 6-135 for *Sensitive Arrays Required for Offloading Operations*

Client Application should invoke `javacard.framework.SensitiveArrays.assertIntegrity(..)` on sensitive output buffers computed by the CSP after processing them.

Note: To share integrity-protected sensitive array objects with the CSP, Client Applications must prepare them as described in section 8.3, *Byte Buffer Parameters*.

For further information, see section 8.3, *Byte Buffer Parameters*.

5.1.4 System Lifecycle

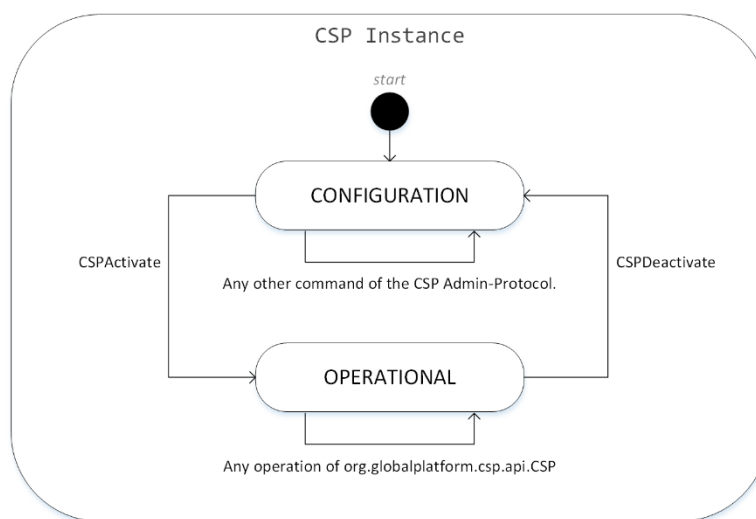
The commands of the CSP Protocol shall trigger internal lifecycle changes of a CSP Instance as shown in Figure 5-1. The first time the CSP Application reaches the lifecycle state `SELECTABLE`, the CSP Instance is set to `CSP_MODE_CONFIGURATION`. In this state, it remains inaccessible to Client Applications, including those with registered AIDs. Any attempt to retrieve CSP services in this condition will trigger an `ERROR_ILLEGAL_CONFIG` [0x3001] exception, signaling that the CSP Configuration is incomplete.

A CSP Instance becomes operational, and thus usable for a Client Application, after executing the `CSPActivate` command, which transitions the CSP Instance to `CSP_MODE_OPERATIONAL`. Once activated, further changes to its configuration are not allowed anymore. The CSP Instance must be deactivated (i.e., set back to `CSP_MODE_CONFIGURATION`) using the `CSPDeactivate` command before it can be modified again.

Note: By activating the CSP Instance, the CSP Admin confirms that it is fully configured and ready for use.

For further information, see section 8.1, CSP API Prerequisites.

Figure 5-1: CSP Instance-internal Lifecycle



5.1.4.1 Internal System Parameters

The CSP shall maintain the internal parameters listed in Table 5-13.

Table 5-13: CSP-internal Parameters

Parameter	Description	Parameter Type
<code>cspApiVersion</code>	Version of the CSP API (Major, Minor, Patch). This is a fixed (not configurable) value, used in the <i>Platform Attestation</i> and the <i>Config Attestation</i> . Technical Details: <ul style="list-style-type: none"> Data type: <code>CSPAPIVersion</code> Retrieved: <code>CSPPlatform</code> 	Global Parameter
<code>cspELFVersion</code>	Version of the CSP ELF file (Major, Minor). This is a fixed (not configurable) value, used in the <i>Platform Attestation</i> and the <i>Config Attestation</i> . Technical Details: <ul style="list-style-type: none"> Data type: <code>CSPELFVersion</code> Retrieved: <code>CSPPlatform</code> 	Global Parameter

Parameter	Description	Parameter Type
platformName	<p>Name and version of the SE platform encoded in ASCII. This is a fixed (not configurable) value, used in the <i>Platform Attestation</i> and the <i>Config Attestation</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: byte[32] • Retrieved: CSPPlatform 	Global Parameter

5.1.5 System Structures

This section lists ASN.1 structures related to CSP system functionality, including CSP settings, error types, API versions, and core data types.

Note: A complete ASN.1 definition is available in [GP CSP ASN1].

5.1.5.1 CSPCoreSupport

This data structure represents features, types, and algorithms of the *Core Modules*.

It is included in the CSPEnforce command to detect their support by the platform.

ASN 5-1: Access: ASN.1 Definition for CSPCoreSupport

```
-- ASN1START
-- Supported core features.
CSPCoreSupport ::= SEQUENCE {
    -- CSP error handling modes.
    errorModes          SET OF CSPErrorMode OPTIONAL
}
-- ASN1STOP
```

5.1.5.2 CSPSettings

This data structure represents general settings of the *System Configuration*.

It can be configured using the CSPSetup structure.

It can be retrieved using the CSPConfiguration structure.

ASN 5-2: Core: ASN.1 Definition for CSPSettings

```
-- ASN1START
-- Structure representing general configuration settings of the CSP Instance.
CSPSettings ::= SEQUENCE {
    -- Custom name of the CSP Configuration chosen by the CSP Admin.
    configName          CSPConfigName,

    -- Custom version of the CSP Configuration chosen by the CSP Admin.
    configVersion       CSPConfigVersion OPTIONAL,

    -- The key that shall be used to compute the config attestation.
    configAttestationKey CSPResourceId OPTIONAL,
}
```

```
-- Specify how the CSP shall handle exceptions.
errorMode          CSPErrorMode DEFAULT error-mode-basic
}
-- ASN1STOP
```

5.1.5.3 CSPProtocolVersion

This data structure represents the version of the *CSP Protocol* and is used in all commands of the protocol to ensure *API Level Compliance*.

It is returned as part of the CSPPatform structure.

Changes due to CSP software updates are logged via CSPEventDataUpdateCSP.

For further information, see section 3.8.1, Platform Attestation, and section 5.1.3.2, API Level Compliance.

ASN 5-3: Core: ASN.1 Definition for CSPProtocolVersion

```
-- ASN1START
-- API level for the CSP Protocol used.
CSPProtocolVersion ::= INTEGER { protocolVersion1(1) }
-- ASN1STOP
```

5.1.5.4 CSPConfiguration

This data structure represents the entire *System Configuration* for a CSP Instance, tailored to a specific use case.

It can be modified using CSPSetup, CSPRegisterClient, CSPUnregisterClient, CSPCreateResource, CSPDestroyResource, CSPConfigureResource, CSPActivate, and CSPDeactivate.

It can be retrieved using the CSPGetConfiguration command.

Modifications of the CSP config are logged through the CSPEventDataUpdateConfig audit event.

ASN 5-4: Core: ASN.1 Definition for CSPConfiguration

```
-- ASN1START
-- Structure containing the entire CSP Configuration.
CSPConfiguration ::= SEQUENCE {

    -- Indicates whether this CSP configuration is currently activated.
    cspMode          CSPMode DEFAULT csp-mode-configuration,

    -- Name, version, attestation key and error handling of the CSP Instance.
    cspSettings      CSPSettings OPTIONAL,

    -- Static (read-only) information about the CSP platform.
    cspPlatform      CSPPatform,

    -- List of CSP Clients registered to this CSP Instance.
    clients          SET OF CSPClient,
```

```
-- List of resources along with their cryptographic configurations.
resources                SET OF CSPResource,

-- Set the general secure channel authentication timeout.
secureChannelSettings    CSPSecureChannelSettings OPTIONAL,

-- Select policy mode for handling unavailable policy types.
policySettings          CSPPolicySettings OPTIONAL,

-- Select counter mode for handling unavailable counter types and sizes.
counterSettings         CSPCounterSettings OPTIONAL,

-- Configure time management and handling of unavailable time.
timeSettings            CSPTimeSettings OPTIONAL,

-- Configure audit event logging and handling of unavailable event types.
auditSettings           CSAuditSettings OPTIONAL,

-- Select field mode for handling unavailable signature fields.
fieldSettings           CSPFieldSettings OPTIONAL
}
-- ASN1STOP
```

5.1.5.5 CSPMode

This data structure defines the list of available *Operation Modes* for a CSP Instance.

The operation mode of a CSP Instance can be toggled using CSPActivate and CSPDeactivate.

It can be retrieved using the CSPConfiguration structure.

ASN 5-5: Core: ASN.1 Definition for CSPMode

```
-- ASN1START
-- List of operations modes of the CSP.
CSPMode ::= ENUMERATED {

    -- CSP_MODE_CONFIGURATION:
    -- The CSP can be configured by the CSP Admin.
    csp-mode-configuration      (0),

    -- CSP_MODE_OPERATIONAL:
    -- The CSP can be used by Client Applications.
    csp-mode-operational        (1)
}
-- ASN1STOP
```

5.1.5.6 CSPConfigVersion

This data structure represents the version of the configuration of a CSP Instance.

It is part of the CSPActivate, CSPSettings, CSPConfigAttestation, and CSPEventDataUpdateConfig structures.

ASN 5-6: Core: ASN.1 Definition for CSPConfigVersion

```
-- ASN1START
-- Version of the configuration of this CSP Instance.
CSPConfigVersion ::= INTEGER (0..32767)
-- ASN1STOP
```

5.1.5.7 CSPConfigName

This data structure represents the name of the configuration of a CSP Instance.

It is part of the CSPSettings, CSPConfigAttestation, and CSPEventDataUpdateConfig structures.

ASN 5-7: Core: ASN.1 Definition for CSPConfigName

```
-- ASN1START
-- Name of the configuration of this CSP Instance; set by the CSP Admin.
CSPConfigName ::= OCTET STRING (SIZE(32))
-- ASN1STOP
```

5.1.5.8 CSPErrorMode

This data structure defines the list of available *Error Modes*.

An error mode can be configured for a CSP Instance using the CSPSettings structure.

It can be retrieved using the CSPConfiguration structure.

The error modes supported can be detected using the CSPEnforce command.

ASN 5-8: Core: ASN.1 Definition for CSPErrorMode

```
-- ASN1START
-- List of error operations modes to specify the error handling of the CSP.
CSPErrorMode ::= ENUMERATED {

    -- ERROR_MODE_BASIC:
    -- The CSP uses only codes 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000.
    error-mode-basic          (0),

    -- ERROR_MODE_DETAILED:
    -- The CSP uses detailed error codes 1xxx-8xxx.
    error-mode-detailed      (1)
}
-- ASN1STOP
```

5.1.5.9 CSPPlatform

This data structure represents a container with information about the CSP platform.

It is used in CSPPlatformAttestation, CSPConfigAttestation, and CSPConfiguration.

ASN 5-9: Core: ASN.1 Definition for CSPPlatform

```
-- ASN1START
-- General information about the CSP platform.
CSPPlatform ::= SEQUENCE {

    -- Version of the CSP Protocol (Version).
    cspProtocolVersion    CSPProtocolVersion,

    -- Version of the Java Card CSP API (Major, Minor, Patch).
    cspApiVersion         CSPAPIVersion,

    -- Version of the CSP ELF file (Major, Minor).
    cspELFVersion         CSPELFVersion,

    -- Name and version of the SE platform encoded in ASCII.
    platformName          OCTET STRING (SIZE (32)),

    -- AID of the CSP Instance.
    cspInstanceAID        CSPAID
}
-- ASN1STOP
```

5.1.5.10 CSPAPIVersion

This data structure represents the version of the Java Card CSP API.

It is returned as part of the CSPPlatform structure.

For further information, see section 3.8.1, Platform Attestation, and section 5.1.3.2, API Level Compliance.

ASN 5-10: Core: ASN.1 Definition for CSPAPIVersion

```
-- ASN1START
-- Version of the Java Card CSP API (Major, Minor, Patch).
CSPAPIVersion ::= OCTET STRING (SIZE(3))
-- ASN1STOP
```

5.1.5.11 CSPELFVersion

This data structure represents the cspELFVersion, the version of the CSP ELF file.

It is returned as part of the CSPPlatform structure and available as FIELD_CSP_ELF_VERSION signature field.

Changes due to CSP software updates are logged via CSPEventDataUpdateCSP.

For further information, see section 3.8.1, Platform Attestation, and section 5.1.3.2, API Level Compliance.

ASN 5-11: Core: ASN.1 Definition for CSPELFVersion

```
-- ASN1START
-- Version of a CSP Application Executable Load File (Major, Minor).
CSPELFVersion ::= OCTET STRING (SIZE(2))
-- ASN1STOP
```

5.1.5.12 CSPAID

This data structure represents an Application Identifier (AID).

It is used in CSPTIMESettings, CSPPlatform, CSPConfiguration, CSPUnregisterClient, and CSPClientApplication.

ASN 5-12: Core: ASN.1 Definition for CSPAID

```
-- ASN1START
-- Application Identifier (AID) of an Application or a Security Domain.
CSPAID ::= OCTET STRING (SIZE (5..16))
-- ASN1STOP
```

5.1.5.13 CSPBoolean

This data structure represents a secure Boolean value. Instead of using a conventional single-bit Boolean, the CSP uses a 2-byte encoded value to increase resistance against fault attacks such as laser fault injection.

The value 0x7878 represents TRUE, and 0x7877 represents FALSE, reducing the risk of accidental or intentional bit flips leading to incorrect behavior.

In the CSP API, the CSP returns the short values CSP.RESULT_TRUE or CSP.RESULT_FALSE instead of the Boolean primitive type.

ASN 5-13: Core: ASN.1 Definition for CSPBoolean

```
-- ASN1START
-- Secure boolean: TRUE is 0x7878, FALSE is 0x7877, other are not defined.
CSPBoolean ::= OCTET STRING (SIZE (2))
-- ASN1STOP
```

5.1.5.14 CSPChallenge

This data structure represents a challenge or nonce.

It is used in CSPDeriveKey, CSPKeyPoPAttestation, CSPPlatformAttestation, CSPConfigAttestation, CSPDataAttestation, and time.getChallenge.

ASN 5-14: Core: ASN.1 Definition for CSPChallenge

```
-- ASN1START
-- Challenge or nonce.
CSPChallenge ::= OCTET STRING (SIZE (16..512))
-- ASN1STOP
```

5.1.5.15 CSPSignature

This data structure represents a digital signature.

It is used in CSPPlatformAttestation, CSPConfigAttestation, CSPDataAttestation, CSPKeyPoPAttestation, and CSPSetTime.

ASN 5-15: Core: ASN.1 Definition for CSPSignature

```
-- ASN1START
-- Digital signature. Covers traditional RSA/ECDSA (512 bytes) and PQC (65KB).
CSPSignature ::= OCTET STRING (SIZE (64..65536))
-- ASN1STOP
```

5.2 Resource Module

The CSP shall use CSP Resource data structures defined in this section to implement *Resource Management*.

For further information, see section 3.1, Resource Management.

5.2.1 Resource Definitions

5.2.1.1 Resource Types

The CSP may support the resource types listed in Table 5-14; the exact list shall be subject to *Modularity*.

The CSP Admin may select the resourceType for a resource using CSPResourceType structure.

The CSP Admin may detect if a resource type is supported using the CSPResourceSupport structure. For each module and each resource type supported shall adhere to the algorithm combinations specified in:

- Table 6-4 for compatible *Cipher, Padding, and Key Size Combinations*
- Table 6-16 for compatible *Signature, Padding, Hash, Key Size, and Curve Combinations*
- Table 6-31 for compatible *Protocol, Resource, Size, and Curve Combinations*
- Table 6-51 for compatible *Attestation Type, Component, and Algorithm Combinations*
- Table 6-64 for compatible *Key Derivation, Resource Type, Hash, Size, and Curve Combinations*
- Table 6-67 for compatible *Key Agreement, Key Type, Curve, and Size Combinations*

For further information, see section 3.1.1, Create Resources.

Table 5-14: Resource Types

Resource Type	Value (Hex)	Value (Int)	Description
RESOURCE_KEY	0x01	1	<p>Resource containing a cryptographic key value and corresponding key parameters. It is utilized in the following CSP services:</p> <ul style="list-style-type: none"> • <i>Cipher Module</i> • <i>Signature Module</i> • <i>Transform Module</i> • <i>Secure Channel Module</i> • <i>Confidential Data Transfer Module</i> • <i>Attestation Module</i> • <i>Key Module</i> <p>Data Type:</p> <ul style="list-style-type: none"> • CSPKey <p>Configuration Parameters:</p> <ul style="list-style-type: none"> • <i>Key Types</i> • <i>Key Sizes</i> • <i>ECC Curves</i>

Resource Type	Value (Hex)	Value (Int)	Description
RESOURCE_CERTIFICATE	0x02	2	<p>Resource containing a certificate and corresponding certificate parameters. It is utilized in the following CSP services:</p> <ul style="list-style-type: none"> • <i>Certificate Module</i> • <i>Secure Channel Module</i> (if required by <i>Protocol Types</i> supported) <p>Data Type:</p> <ul style="list-style-type: none"> • CSPCertificate <p>Configuration Parameters:</p> <ul style="list-style-type: none"> • <i>Certificate Types</i>
RESOURCE_PASSWORD	0x03	3	<p>Resource containing a PIN or passphrase as a byte array of variable size and corresponding password parameters. It is utilized in the following CSP services:</p> <ul style="list-style-type: none"> • <i>Password Module</i> • <i>Secure Channel Module</i> (if required by the <i>Protocol Types</i> supported) <p>Data Type:</p> <ul style="list-style-type: none"> • CSPPassword <p>Configuration Parameters:</p> <ul style="list-style-type: none"> • <i>Password Types</i> • minSize, maxSize, tryLimit
RESOURCE_COUNTER	0x04	4	<p>Resource containing a counter value and corresponding parameters. It is utilized in the following CSP services:</p> <ul style="list-style-type: none"> • <i>Counter Module</i> <p>Data type:</p> <ul style="list-style-type: none"> • CSPCounter <p>Configuration Parameters:</p> <ul style="list-style-type: none"> • <i>Counter Types</i> • <i>Counter Capacities</i> • counterLimit
RESOURCE_TIMER	0x05	5	<p>Resource containing a time value with corresponding timer parameters. It is utilized in the following CSP services:</p> <ul style="list-style-type: none"> • <i>Time Module</i> <p>Data type:</p> <ul style="list-style-type: none"> • CSPManualTimer <p>Configuration Parameters:</p> <ul style="list-style-type: none"> • <i>Timer Types</i> • <i>Time Formats</i> • <i>Timeout Types</i> • timeLimit

5.2.1.2 Usage Types

For supported *Modules*, the CSP shall evaluate the usage types listed in Table 5-15, as specified in:

- Table 5-21 for *Access Restrictions on Resource Operations*

- Table 6-10 for *Access Restrictions on Cipher Operations*
- Table 6-21 for *Access Restrictions on Signature Operations*
- Table 6-26 for *Access Restrictions on Transform Operations*
- Table 6-42 for *Access Restrictions on Secure Channel Operations*
- Table 6-48 for *Access Restrictions on Confidential Data Transfer Operations*
- Table 6-55 for *Access Restrictions on Attestation Operations*
- Table 6-72 for *Access Restrictions on Key Operations*
- Table 6-81 for *Access Restrictions on Certificate Operations*
- Table 6-89 for *Access Restrictions on Password Operations*
- Table 6-101 for *Access Restrictions on Counter Operations*
- Table 6-116 for *Access Restrictions on Time Operations*
- Table 6-124 for *Access Restrictions on Audit Operations*
- Table 6-133 for *Access Restrictions on Offloading Operations*

The CSP Admin may select the usageType of a resource using CSPUsageType structure.

Note: Each resource can be assigned exactly one usage type, dedicating it to a specific task.

For further information, see section 3.2, Access Control, and section 3.2.5, Usage Concept.

Table 5-15: Resource Usage Types

Usage	Value (Hex)	Value (Int)	Description	Resource Type
USAGE_CIPHER	0x01	1	Only resources with USAGE_CIPHER can be used encryption and decryption as part of <i>Cipher and Signatures</i> services, through the CSP's <i>Cipher Operations</i> . Dedicated for: <ul style="list-style-type: none"> • <i>Cipher Module</i> 	Keys
USAGE_SIGNATURE	0x02	2	Only resources with USAGE_SIGNATURE can be used for creating and verifying signatures as part of <i>Cipher and Signatures</i> services, through the CSP's <i>Signature Operations</i> . Dedicated for: <ul style="list-style-type: none"> • <i>Signature Module</i> • <i>Time Module</i> 	Keys
USAGE_TRANSFORM	0x03	3	Only resources with USAGE_TRANSFORM can be used to decrypt within encryption transformation from one key or algorithm to another as part of <i>Cipher and Signatures</i> services, through the CSP's <i>Transform Operations</i> . Dedicated for: <ul style="list-style-type: none"> • <i>Transform Module</i> 	Keys

Usage	Value (Hex)	Value (Int)	Description	Resource Type
USAGE_SECCHANNEL	0x04	4	<p>Only resources with USAGE_SECCHANNEL can be used for <i>Secure Messaging</i> through the CSP's <i>Secure Channel Operations</i>.</p> <p>Note: Specifically, these resources are not permitted for use in <i>Password Operations</i>.</p> <p>Dedicated for:</p> <ul style="list-style-type: none"> • <i>Secure Channel Module</i> 	All
USAGE_CONFIDENTIAL	0x05	5	<p>Only resources with USAGE_CONFIDENTIAL can be used for transferring encryption from confidential-layer to storage-layer, through the CSP's <i>Confidential Data Transfer Operations</i>.</p> <p>Note: Specifically, these resources are not permitted for use in standard Cipher Operations.</p> <p>Dedicated for:</p> <ul style="list-style-type: none"> • <i>Confidential Data Transfer Module</i> • <i>Transform Module</i> 	Keys
USAGE_ATTESTATION	0x06	6	<p>Only resources with USAGE_ATTESTATION be used as signing key for computing <i>Attestations</i>, through the CSP's <i>Attestation Operations</i>.</p> <p>Note: Specifically, these resources are not permitted for use in standard <i>Signature Operations</i>.</p> <p>Dedicated for:</p> <ul style="list-style-type: none"> • <i>Attestation Module</i> • <i>System Attestations</i> 	Keys
USAGE_KEY	0x07	7	<p>Only resources with USAGE_KEY can be used as source keys for <i>Key Management</i>, such as key derivation and key agreement, through the CSP's <i>Key Operations</i>.</p> <p>Dedicated for:</p> <ul style="list-style-type: none"> • <i>Key Module</i> 	All
USAGE_PASSWORD	0x09	9	<p>Only resources with USAGE_PASSWORD can be used for <i>Password Verification</i>, through the CSP's <i>Password Operations</i>.</p> <p>Dedicated for:</p> <ul style="list-style-type: none"> • <i>Password Module</i> 	Passwords
USAGE_AUDIT	0x0C	12	<p>Only resources with USAGE_AUDIT can be used as signing key for <i>Creating Log Messages</i> through the CSP's <i>Audit Operations</i>.</p> <p>Note: Specifically, these resources are not permitted for use in standard <i>Signature Operations</i>.</p> <p>Dedicated for:</p> <ul style="list-style-type: none"> • <i>Audit Module</i> 	Keys

Usage	Value (Hex)	Value (Int)	Description	Resource Type
USAGE_OFFLOADING	0x0D	13	<p>Only resources with USAGE_OFFLOADING can be used as encryption key for <i>Streaming Resources for Offloading</i>, i.e., the encrypted import or export, through the CSP's <i>Offloading Operations</i>.</p> <p>Note: Specifically, these resources are not permitted for use in standard <i>Cipher Operations</i>.</p> <p>Dedicated for:</p> <ul style="list-style-type: none"> • <i>Offloading Module</i> 	Keys
USAGE_SYSTEM	0xFE	254	<p>Only resources with USAGE_SYSTEM can be used to decrypt resource values imported by the CSP Admin through the CSPSetValue command.</p> <p>Note: Specifically, these resources are not permitted for use in standard <i>Cipher Operations</i>.</p> <p>Dedicated for:</p> <ul style="list-style-type: none"> • <i>System Module</i> 	Keys

5.2.1.3 Resource States

For supported *Modules*, the CSP shall trigger *Resource Lifecycle* changes using the CSPResourceState, stored in persistent memory, listed in Table 5-16 and evaluate the currently active resourceState as specified in:

- Table 5-21 for *Access Restrictions on Resource Operations*
- Table 6-10 for *Access Restrictions on Cipher Operations*
- Table 6-21 for *Access Restrictions on Signature Operations*
- Table 6-26 for *Access Restrictions on Transform Operations*
- Table 6-42 for *Access Restrictions on Secure Channel Operations*
- Table 6-48 for *Access Restrictions on Confidential Data Transfer Operations*
- Table 6-55 for *Access Restrictions on Attestation Operations*
- Table 6-72 for *Access Restrictions on Key Operations*
- Table 6-81 for *Access Restrictions on Certificate Operations*
- Table 6-89 for *Access Restrictions on Password Operations*
- Table 6-101 for *Access Restrictions on Counter Operations*
- Table 6-116 for *Access Restrictions on Time Operations*
- Table 6-124 for *Access Restrictions on Audit Operations*
- Table 6-133 for *Access Restrictions on Offloading Operations*

Table 5-16: Resource States

State	Value (Hex)	Value (Int)	Description	Resource Type
STATE_UNINITIALIZED	0x01	1	Uninitialized state for resources. The resource may lack full configuration and is not ready for cryptographic operations. Events: <ul style="list-style-type: none"> EVENT_RESOURCE_CLEARED 	all Resources
STATE_OPERATIONAL	0x02	2	Operational state for resources. The resource is ready to be used for cryptographic or security-related operations. Resources transition to OPERATIONAL state through: <ul style="list-style-type: none"> Load Resource for Personalization Key Generation, Key Derivation, Key Agreement Import Certificates Updating Passwords, Unblocking Passwords Only resources in OPERATIONAL state can be used for: <ul style="list-style-type: none"> Key Management Export Certificates, Certificate Trust Chain Verification Password Verification Cipher and Signatures Attestations Creating Log Messages 	all Resources
STATE_BLOCKED	0x09	9	Blocked state for resources, that occurs when the built-in <i>Retry Counter with Maximum Try Limit</i> is exceeded during <ul style="list-style-type: none"> Password Verification PACE authentication The Resource cannot be used for any cryptographic or security-related operations until it is unblocked with <ul style="list-style-type: none"> Unblocking Passwords Events: <ul style="list-style-type: none"> EVENT_PASSWORD_BLOCKED EVENT_PASSWORD_UNBLOCKED 	only Passwords Resources
STATE_EXHAUSTED	0x0B	11	The state of a Resource when a manual counter or a built-in usage counter for keys or passwords exceeds the configured <i>counterLimit</i> . This state is triggered by the following <i>Counter Types</i> : <ul style="list-style-type: none"> COUNT_MANUAL COUNT_USAGE, COUNT_USAGE_PER_BLOCK COUNT_USAGE_SUCCESS_ONLY, COUNT_USAGE_FAILURE_ONLY COUNT_TRANSPORT_USAGE An exhausted Resource cannot be used for any cryptographic or security-related operations until it is cleared and re-initialized. Events: <ul style="list-style-type: none"> EVENT_COUNTER_EXHAUSTED 	Key, Password and Counter Resources

State	Value (Hex)	Value (Int)	Description	Resource Type
STATE_EXPIRED	0x0C	12	<p>The state of a resource when a manual timer or a built-in validity timer for keys, password and certificates expires. This state is triggered by the following <i>Timer Types</i>:</p> <ul style="list-style-type: none"> TIMER_MANUAL_PERIOD, TIMER_VALIDITY_PERIOD TIMER_MANUAL_DATE, TIMER_VALIDITY_DATE TIMER_VALIDITY_CERTIFICATE <p>An expired resource cannot be used for any cryptographic or security-related operations until it is cleared and re-initialized.</p> <p>Events:</p> <ul style="list-style-type: none"> EVENT_TIMER_EXPIRED 	Key, Password, Certificate and Timer Resources

5.2.1.4 Resource Identifiers

Table 5-17 lists recommended ranges for CSP Resource Identifiers depending on their type and purpose. Using standard key, certificate and password identifiers is not obligatory. It is the decision of the CSP Admin which resource identifiers are to be used.

Table 5-17: Resource ID Ranges

Type	ID Range	Purpose
Cipher Resources	0x01XX	Resources for USAGE_CIPHER.
Signature Resources	0x02XX	Resources for USAGE_SIGNATURE.
Transform Resources	0x03XX	Resources for USAGE_TRANSFORM.
Secure Channel Resources	0x04XX	Resources for USAGE_SECCHANNEL.
Confidential Resources	0x05XX	Resources for USAGE_CONFIDENTIAL.
Attestation Resources	0x06XX	Resources for USAGE_ATTESTATION.
Key Management Resources	0x07XX	Resources for USAGE_KEY.
Password Management Resources	0x09XX	Resources for USAGE_PASSWORD.
Counter Resources	0x0AXX	Resources of type RESOURCE_COUNTER.
Time Resources	0x0BXX	Resources of type RESOURCE_TIMER.
Audit Resources	0x0CXX	Resources for USAGE_AUDIT.
Offloading Resources	0x0DXX	Resources for USAGE_OFFLOADING.

5.2.1.5 Core Resource Events

The AuditModule may log the resource-related *Audit Events* listed in Table 5-18; the exact list shall be subject to *Modularity*. The event-specific data listed in the table shall be stored according to the configured auditMode.

The CSP Admin may configure the resourceEvents to be logged using CSPResourceEvent.

The CSP Admin may detect if an event type is supported using the CSPAuditSupport structure.

For further information, see section 3.11.4, Selecting Audit Events, and section 6.12.1.2, Audit Events.

Table 5-18: Resource Events for Auditing

Event Type	Value (Hex)	Value (Int)	Description	Event Category
EVENT_RESOURCE_CLEARED	0x1000	4096	<p>Resource cleared successfully. This event is logged when a resource value is removed, and the resource is reverted to STATE_UNINITIALIZED.</p> <p>Note: This event does not occur if the resource is already in STATE_UNINITIALIZED.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Format: CSPEventDataResource Operations: resource.clear <p>Event Data:</p> <ul style="list-style-type: none"> resourceId from which the data was removed 	Resource Event
EVENT_RESOURCE_VALUE_SET	0x1001	4097	<p>Resource modified. This event is logged after setting a resource value.</p> <p>Note: Setting a concrete public key or certificate value or updating a password is not included in this event.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Format: CSPEventDataResource Operations: CSPSetValue <p>Event Data:</p> <ul style="list-style-type: none"> resourceId of the resource modified 	Resource Event

5.2.2 Resource Configuration

The CSP shall support the configuration parameters defined in Table 5-19.

For further information, see section 3.1.4, Configure Resources.

Table 5-19: Resource Configuration Parameters

Parameter	Description	Parameter Type
resourceId	<p>Unique identifier of the resource. <i>Resource Identifiers</i> are used in most of the CSP API operations to identify the resource and corresponding configurations, such as:</p> <ul style="list-style-type: none"> Key Management Certificate Management Password Management Cipher and Signatures Secure Messaging <p>It is logged for all <i>Audit Events</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Data type: CSPResourceId Administration: CSPResource 	Resource Parameter

Parameter	Description	Parameter Type
resourceType	Defines the type of the resource, selected from the available <i>Resource Types</i> . Examples are key, certificate and password resource. Technical Details: <ul style="list-style-type: none"> Data type: CSPResourceType Administration: CSPResource Operations: resource.getResourceType 	Resource Parameter
accessControl	Represents the <i>Access Configuration</i> of the resource. The CSP Admin may <i>Configure Resources</i> to change these access control settings. Technical Details: <ul style="list-style-type: none"> Data type: CSPAccessControl Administration: CSPResource, CSPConfigureResource 	Resource Parameter
usageType	Defines the concrete purpose of the resource, restricting its use to a specific module, selected from the available <i>Usage Types</i> . The CSP Admin may <i>Configure Resources</i> to change the usage type. Technical Details: <ul style="list-style-type: none"> Data type: CSPUsageType Administration: CSPAlgorithms 	Resource Parameter
algorithms	Contains the concrete algorithm configuration the resource shall be utilized with. The CSP Admin may <i>Configure Resources</i> to change the algorithms. Technical Details: <ul style="list-style-type: none"> Data type: CSPAlgorithms Administration: CSPResource, CSPConfigureResource 	Resource Parameter
counters	Contains the concrete algorithm configuration the resource shall be utilized with. The CSP Admin may <i>Configure Resources</i> to change the counter settings. Technical Details: <ul style="list-style-type: none"> Data type: CSPCounters Administration: CSPResource, CSPConfigureResource 	Resource Parameter
timers	Built-in timers configured to the resource. The CSP Admin may <i>Configure Resources</i> to change the timer settings. Technical Details: <ul style="list-style-type: none"> Data type: CSPTimers Administration: CSPResource, CSPConfigureResource 	Resource Parameter

5.2.3 Resource Operations

The CSP shall implement the operations listed in Table 5-20.

For each operation, the CSP shall handle supported *Counter Types*, *Timer Types*, and *Audit Events*.

These operations shall comply with the *Access Restrictions on Resource Operations* specified in Table 5-21.

Table 5-20: Resource Operations

Operation	Description
resource.getState	Returns the current resourceState of the resource. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.ResourceService.getState(..) Evaluate Timers <ul style="list-style-type: none"> TIMER_VALIDITY_DATE, TIMER_VALIDITY_PERIOD
resource.getResourceType	Returns the resourceType of the resource. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.ResourceService.getResourceType(..)
resource.clear	Reverts a resource to STATE_UNINITIALIZED, restoring all attributes to their default settings. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.ResourceService.clear(..) CSP Protocol: <ul style="list-style-type: none"> CSPClearResource Fire Events: <ul style="list-style-type: none"> EVENT_RESOURCE_CLEARED
resource.clearTransient	Resets all transient data within the CSP Instance, including clearing <i>Transient Keys</i> marked with transient, resetting the authenticated flags of all passwords, and reinitializing the security settings same as invoking sc.resetSecurity for all secure channel service instances. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.ResourceService.clearTransient(..) Fire Events: <ul style="list-style-type: none"> EVENT_RESOURCE_CLEARED (for each resource cleared)
CSPSetValue	Imports the value of a resource to <i>Load Resource for Personalization</i> . CSP Protocol: <ul style="list-style-type: none"> CSPSetValue Fire Events: <ul style="list-style-type: none"> EVENT_RESOURCE_VALUE_SET

5.2.3.1 Access Restrictions on Resource Operations

The CSP shall evaluate the access restrictions listed in Table 5-21 for all *Access Rights*, *Usage Types*, *Resource States*, and *Policy Types* supported by the platform. Unauthorized or illegal operation invocations shall be rejected and handled according to the errorMode configured.

Table 5-21: Resource Operations Access Restrictions

Operation	Right Required	Usage Required	State Required	Other Restrictions
resource.clear	CLEAR	-	-	<ul style="list-style-type: none"> Client Authentication required
resource.clearTransient	USE	-	-	<ul style="list-style-type: none"> Client Authentication required
resource.getState	USE	-	-	<ul style="list-style-type: none"> Client Authentication required

Operation	Right Required	Usage Required	State Required	Other Restrictions
CSPSetValue (resource to set the value)	SETUP	-	UNINITIALIZED	<ul style="list-style-type: none"> For CSP Admins secured via CSP SD <i>Client Authentication</i> required
CSPSetValue (decryption key)	USE	SYSTEM	OPERATIONAL	

5.2.3.2 Sensitive Results computed by Resource Operations

The CSP shall temporarily store the results of methods listed in Table 5-22 in transient memory and shall implement the `assertSensitiveResult` operation, enabling Client Applications to verify that these results were computed without any abnormality.

Table 5-22: Resource Operations Requiring Sensitive Results Checks

Operation	Result Type	Result Description
<code>resource.getState</code>	short	The state of the resource.

5.2.4 Resource Lifecycle

For *Modules* supported, the CSP shall trigger lifecycle changes for resources as specified in

- Section 6.7.4 *Key Lifecycle*
- Section 6.8.4 *Certificate Lifecycle*
- Section 6.9.4 *Password Lifecycle*
- Section 6.10.4 *Counter Lifecycle*
- Section 6.11.4 *Time Lifecycle*

For further information, see section 3.10.1, Overview Timers, and section 3.9.1 Overview Counters.

5.2.4.1 Internal Resource Parameters

The CSP shall maintain the internal parameters listed in Table 5-23.

Table 5-23: Resource-internal Parameters

Parameter	Description	Parameter Type
<code>resourceState</code>	<p>Current state of the resource, one of the available <i>Resource States</i>. The CSP evaluates this state to enforce <i>Access Control</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Initial value: <code>0x01 (STATE_UNINITIALIZED)</code> Operations: <code>resource.getState</code> 	Resource Parameter

5.2.5 Resource Structures

This section lists ASN.1 structures related to general resource configurations.

Note: A complete ASN.1 definition is available in [GP CSP ASN1].

5.2.5.1 CSPResourceSupport

This data structure represents features, types, and algorithms of the *Resource Module*.

It is included in the CSPEnforce command to detect their support by the platform.

ASN 5-16: Resource: ASN.1 Definition for CSPResourceSupport

```
-- ASN1START
-- Supported resource features.
CSPResourceSupport ::= SEQUENCE {

    -- Supported resource types.
    resourceTypes      SET OF CSPResourceType OPTIONAL
}
-- ASN1STOP
```

5.2.5.2 CSPResource

This data structure represents a container for a *Resource Configuration*.

It is part of the CSPConfiguration structure.

It can be created using the CSPCreateResource command.

It can be modified using the CSPConfigureResource command.

It can be removed using the CSPDestroyResource command.

For further information, see section 3.1.5, Resource Identifiers.

ASN 5-17: Resource: ASN.1 Definition for CSPResource

```
-- ASN1START
-- Data structure representing a resource.
CSPResource ::= SEQUENCE {

    -- Unique identifier of the resource chosen by the CSP Admin.
    resourceId          CSPResourceId,

    -- The resource-specific parameters (e.g., key type, curve, min-size).
    resourceParams      CHOICE {

        -- Parameters specific to key resources.
        keyParams        CSPKey,

        -- Parameters specific to certificate resources.
        certificateParams CSPCertificate,

        -- Parameters specific to password resources.
        passwordParams   CSPPassword,

        -- Parameters specific to manual counter resources.
    }
}
```

```

        counterParams      CSPCounter,

        -- Parameters specific to manual timer resources.
        timerParams        CSPManualTimer
    },

    -- The usage type specifying the CSP operations allowed for the resource.
    usageType              CSPUsageType,

    -- Access control configuration for the resource.
    accessControl          CSPAccessControl OPTIONAL,

    -- The algorithm configuration of the resource.
    algorithms              CSPAlgorithms OPTIONAL,

    -- Built-in counters (only for keys and passwords).
    counters               CSPCounters OPTIONAL,

    -- Built-in timers (only for keys, certificates and passwords).
    timers                 CSPTimers OPTIONAL,

    -- The events that shall be audited for this resource.
    resourceEvents         SET OF CSPResourceEvent OPTIONAL
}
-- ASN1STOP

```

5.2.5.3 CSPResourceId

This data structure represents a unique resource identifier.

It is part of the CSPResource structure.

It is assigned by the CSP Admin using the CSPCreateResource command.

It is used in several operations and commands to refer to a resource.

It is used to remove a resource using the CSPDestroyResource command.

For further information, see section 3.1.5, Resource Identifier Concept, and section 5.2.1.4, Resource Identifiers.

ASN 5-18: Resource: ASN.1 Definition for CSPResourceId

```

-- ASN1START
-- Unique identifier of a Resource.
CSPResourceId ::= INTEGER (0..32767)
-- ASN1STOP

```

5.2.5.4 CSPResourceState

This data structure represents the state of a resource.

For further information, see section 5.2.1.3, Resource States.

ASN 5-19: Resource: ASN.1 Definition for CSPResourceState

```
-- ASN1START
-- List of available resource states.
CSPResourceState ::= ENUMERATED {

    -- STATE_UNINITIALIZED:
    -- Uninitialized state for resources.
    state-uninitialized      (1),

    -- STATE_OPERATIONAL:
    -- Operational state for resources.
    state-operational       (2),

    -- STATE_BLOCKED:
    -- Blocked state when the password try limit is reached.
    state-blocked           (9),

    -- STATE_EXHAUSTED:
    -- Exhausted state when a counter exceeds the configured limit.
    state-exhausted         (11),

    -- STATE_EXPIRED:
    -- Expired state when a timer expires.
    state-expired           (12)
}
-- ASN1STOP
```

5.2.5.5 CSPResourceValue

This data structure represents the plain (unencrypted) value of a resource.

It is part of the CSPDataAttestation and CSPKeyPoPAttestation structures.

Note: This data structure is only used for public keys, counter and timer values.

ASN 5-20: Resource: ASN.1 Definition for CSPResourceValue

```
-- ASN1START
-- The value of a public key, counter or timer resource.
CSPResourceValue ::= OCTET STRING (SIZE(64..32768))
-- ASN1STOP
```

5.2.5.6 CSPResourceType

This data structure defines the list of available *Resource Types*.

A resource type can be selected using the CSPResource structure.

The resource types supported can be detected using the CSPResourceSupport structure.

ASN 5-21: Resource: ASN.1 Definition for CSPResourceType

```
-- ASN1START
-- List of available resource types.
CSPResourceType ::= ENUMERATED {

    -- RESOURCE_KEY:
    -- Key resource type.
    resource-key          (1),

    -- RESOURCE_CERTIFICATE:
    -- Certificate resource type.
    resource-certificate  (2),

    -- RESOURCE_PASSWORD:
    -- Password resource type.
    resource-password     (3),

    -- RESOURCE_COUNTER:
    -- Counter resource type.
    resource-counter      (4),

    -- RESOURCE_TIMER:
    -- Timer resource type.
    resource-timer        (5)
}
-- ASN1STOP
```

5.2.5.7 CSPUsageType

This data structure defines the list of available *Usage Types*.

A usage can be selected using the CSPResource structure.

For further information, see section 3.2, Access Control.

ASN 5-22: Resource: ASN.1 Definition for CSPUsageType

```
-- ASN1START
-- Resource usage types to limit its usage to certain cryptographic operations.
CSPUsageType ::= ENUMERATED {

    -- USAGE_CIPHER:
    -- Restrict the use to cipher operations (keys).
    usage-cipher          (1),

    -- USAGE_SIGNATURE:
    -- Restrict the use to signature operations (keys).
    usage-signature        (2),
}
```

```

-- USAGE_TRANSFORM:
-- Restrict the use to encryption transformation operations (keys).
usage-transform      (3),

-- USAGE_SECCHANNEL:
-- Restrict the use to secure message establishment (any).
usage-secchannel     (4),

-- USAGE_CONFIDENTIAL:
-- Restrict the use to confidential data transfer (keys).
usage-confidential   (5),

-- USAGE_ATTESTATION:
-- Restrict the use as attestation signing key (keys).
usage-attestation    (6),

-- USAGE_KEY:
-- Restrict the use to key derivation and key agreement (any).
usage-key            (7),

-- USAGE_PASSWORD:
-- Restrict the use to password verification (passwords only).
usage-password       (9),

-- USAGE_AUDIT:
-- Restrict the use as audit log message signing key (keys).
usage-audit          (12),

-- USAGE_OFFLOADING:
-- Restrict the use as offloading key for resource import/export (keys).
usage-offloading     (13)
}
-- ASN1STOP

```

5.2.5.8 CSPAlgorithms

This data structure represents a container for a resource-specific algorithm configuration.

It is part of the CSPResource.

It can be modified using the CSPConfigureResource command.

For further information, see section 3.2, Access Control, and section 3.2.5, Usage Concept.

ASN 5-23: Resource: ASN.1 Definition for CSPAlgorithms

```

-- ASN1START
-- Data structure for the algorithm configuration of a resource.
CSPAlgorithms ::= CHOICE {

```

```
-- No algorithm configured.
noneAlgorithm          NULL,

-- USAGE_CIPHER, USAGE_TRANSFORM, USAGE_CONFIDENTIAL.
cipherAlgorithms       CSPPCipherAlgorithms,

-- USAGE_SIGNATURE, USAGE_AUDIT.
signatureAlgorithms    CSPSignatureAlgorithms,

-- USAGE_SECCHANNEL: secure channel authentication.
secChannelAlgorithms   CSPSecureChannelAlgorithms,

-- USAGE_ATTESTATION, USAGE_AUDIT.
attestationAlgorithm    CSPAttestationAlgorithms,

-- USAGE_KEY: Key derivation algorithm.
keyDerivationAlgorithm  CSPKeyDerivationAlgorithms,

-- USAGE_KEY: Key agreement scheme.
keyAgreementScheme     CSPKeyAgreementScheme
}
-- ASN1STOP
```

5.3 Access Module

The CSP shall support *Access Control* mechanisms for CSP Resources defined in this section.

For further information, see section 3.2, Access Control.

5.3.1 Access Definitions

5.3.1.1 Client Authentication

The CSP shall provide the option to restrict access to its services only when a specific secure channel is successfully established.

The CSP shall support the use of `PROTOCOL_PACE` with `SEC_PACE_PIN` for this scenario.

The CSP Admin may select the `authProtocol` from *Protocol Types* supported and configure the `authResources` (CSP resources required to establish the secure channel) through `CSPRegisterClient`.

If not configured, the CSP shall deny any invocations from off-card Clients.

If configured but not fully established, the CSP shall deny any invocation from CSP Clients to the following CSP operations:

- Table 5-21 for *Access Restrictions on Resource Operations*
- Table 6-10 for *Access Restrictions on Cipher Operations*
- Table 6-21 for *Access Restrictions on Signature Operations*
- Table 6-26 for *Access Restrictions on Transform Operations*
- Table 6-55 for *Access Restrictions on Attestation Operations*
- Table 6-72 for *Access Restrictions on Key Operations*
- Table 6-81 for *Access Restrictions on Certificate Operations*
- Table 6-89 for *Access Restrictions on Password Operations*
- Table 6-101 for *Access Restrictions on Counter Operations*
- Table 6-116 for *Access Restrictions on Time Operations*
- Table 6-124 for *Access Restrictions on Audit Operations*
- Table 6-133 for *Access Restrictions on Offloading Operations*

For further information, see section 3.2.2, Enforce Client Authentication.

5.3.1.2 Access Rights

The CSP shall evaluate the *ACR Bit Positions* for resources based on the access rights listed in Table 5-24 for all *Modules* supported.

The CSP Admin shall be able to configure these access rights to resources as `CSPAccessControlRules` through the `CSPAccessControl` structure.

The CSP shall evaluate the access control rules as specified in:

- Table 5-21 for *Access Restrictions on Resource Operations*
- Table 6-10 for *Access Restrictions on Cipher Operations*
- Table 6-21 for *Access Restrictions on Signature Operations*

- Table 6-26 for *Access Restrictions on Transform Operations*
- Table 6-42 for *Access Restrictions on Secure Channel Operations*
- Table 6-48 for *Access Restrictions on Confidential Data Transfer Operations*
- Table 6-55 for *Access Restrictions on Attestation Operations*
- Table 6-72 for *Access Restrictions on Key Operations*
- Table 6-81 for *Access Restrictions on Certificate Operations*
- Table 6-89 for *Access Restrictions on Password Operations*
- Table 6-101 for *Access Restrictions on Counter Operations*
- Table 6-116 for *Access Restrictions on Time Operations*
- Table 6-124 for *Access Restrictions on Audit Operations*
- Table 6-133 for *Access Restrictions on Offloading Operations*

For further information, see section 3.2.4, Access Control Rules.

Table 5-24: Access Rights

Access Rights	Value (Hex)	Value (Int)	Operations
ACCESS_USE	0x01	1	<p>The access right USE permits the use of a resource for cryptographic or security-related operations such as</p> <ul style="list-style-type: none"> • <i>Key Derivation</i> and <i>Key Agreement</i> (source key) • <i>Import and Export Public Keys</i> (export) • <i>Import Certificates</i> (export) • <i>Password Verification</i> • <i>Cipher and Signatures</i> • <i>Secure Messaging</i> • <i>Attestations</i>
ACCESS_SETUP	0x02	2	<p>The access right SETUP permits modifications to the value of resources, covering:</p> <ul style="list-style-type: none"> • <i>Key Generation, Key Derivation, Key Agreement</i> (destination key) • <i>Import and Export Public Keys</i> (import) • <i>Import Certificates</i> (import), <i>Certificate Rollover</i> • <i>Updating Passwords, Unblocking Passwords</i> <p>Note: A POLICY_UNBLOCK_PASSWORD may be used to further restrict unblocking a password.</p>
ACCESS_CLEAR	0x04	4	<p>The access right CLEAR permits to <i>Clear Resources</i>, i.e., resetting a resource to STATE_UNINITIALIZED, including securely wiping the value of the resource, such as key or password values.</p> <p>Note: This right is a prerequisite for performing most SETUP operations, as they typically require a resource to be in STATE_UNINITIALIZED.</p>
ACCESS_MOVE	0x08	8	<p>The access right MOVE permits the encrypted import and export of resources, encompassing all resource types, including private keys and passwords.</p> <ul style="list-style-type: none"> • <i>Streaming Resources</i> for Offloading (import and export)

5.3.1.3 ACR Bit Positions

The CSP shall evaluate *Access Control Rules* (ACR) configured for a resource according to the bit values specified in Table 5-26, with explanations provided in Table 5-25, which refer to the *Access Rights* defined in Table 5-24. Specifically, the:

- Bit positions 1-8 define the actions that a Client Application is allowed to perform on the resource.
- Bit positions 9-12 define the actions that the CSP Admin is allowed to perform on the resource.
- Bit positions 13-16 define the actions that an off-card Client is allowed to perform on the resource.

The CSP Admin shall be able to configure these access control rules to resources as `CSPAccessControlRules` using the `CSPAccessControl` structure.

For further information, see section 3.2.3, Resource Ownership, and section 3.2.4, Access Control Rules.

Table 5-25: Access Control Rules

ACR	Description
ANY_USE	All on-card Client Applications have the ACCESS_USE right for the resource.
ANY_SETUP	All on-card Client Applications have the ACCESS_SETUP right for the resource.
ANY_CLEAR	All on-card Client Applications have the ACCESS_CLEAR right for the resource.
ANY_MOVE	All on-card Client Applications have the ACCESS_MOVE right for the resource.
OWNER_USE	Only the owner of the resource has the ACCESS_USE right.
OWNER_SETUP	Only the owner of the resource has the ACCESS_SETUP right.
OWNER_CLEAR	Only the owner of the resource has the ACCESS_CLEAR right.
OWNER_MOVE	Only the owner of the resource has the ACCESS_MOVE right.
ADMIN_USE	The CSP Admin has the ACCESS_USE right for the resource.
ADMIN_SETUP	The CSP Admin has the ACCESS_SETUP right for the resource.
ADMIN_CLEAR	The CSP Admin has the ACCESS_CLEAR right for the resource.
ADMIN_MOVE	The CSP Admin has the ACCESS_MOVE right for the resource.
CLIENT_USE	All authenticated off-card Clients have the ACCESS_USE right for the resource.
CLIENT_SETUP	All authenticated off-card Clients have the ACCESS_SETUP right for the resource.
CLIENT_CLEAR	All authenticated off-card Clients have the ACCESS_CLEAR right for the resource.
CLIENT_MOVE	All authenticated off-card Clients have the ACCESS_MOVE right for the resource.

Table 5-26: Access Control Rules Bit Positions

b16	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	ACR
-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	1	ANY_USE
-	-	-	-	-	-	-	-	-	-	0	-	-	-	1	-	ANY_SETUP
-	-	-	-	-	-	-	-	-	0	-	-	-	1	-	-	ANY_CLEAR
-	-	-	-	-	-	-	-	0	-	-	-	1	-	-	-	ANY_MOVE
-	-	-	0	-	-	-	-	-	-	-	1	-	-	-	0	OWNER_USE
-	-	0	-	-	-	-	-	-	-	1	-	-	-	0	-	OWNER_SETUP
-	0	-	-	-	-	-	-	-	1	-	-	-	0	-	-	OWNER_CLEAR
0	-	-	-	-	-	-	-	1	-	-	-	0	-	-	-	OWNER_MOVE
-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	ADMIN_USE
-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	ADMIN_SETUP
-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	ADMIN_CLEAR
-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	ADMIN_MOVE
-	-	-	1	-	-	-	-	-	-	-	0	-	-	-	-	CLIENT_USE
-	-	1	-	-	-	-	-	-	-	0	-	-	-	-	-	CLIENT_SETUP
-	1	-	-	-	-	-	-	-	0	-	-	-	-	-	-	CLIENT_CLEARCLI ENT_CLEAR
1	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	CLIENT_MOVE

X = Bit is not evaluated. - = Any bit can be used. 0 = Bit cannot be set. 1 = Bit determines the action.

5.3.2 Access Configuration

The CSP shall support the configuration parameters defined in Table 5-27.

Table 5-27: Access Configuration Options

Parameter	Description	Parameter Type
authProtocol	<p>The secure channel protocol type, selected from available <i>Protocol Types</i>, restricts access to CSP services, allowing access only when the secure channel is fully established using the configured authResources.</p> <p>Note: If not configured, off-card Clients cannot access CSP services. However, Client Applications can still use CSP API operations.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPProtocolType • Initial value: - • Administration: CSPRegisterClient 	CSP Instance Parameter

Parameter	Description	Parameter Type
authResources	<p>The CSP Resources used for <i>Client Authentication</i> to establish the secure channel protocol configured by authProtocol.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: SET OF CSPResourceId • Initial value: - • Administration: CSPRegisterClient 	CSP Instance Parameter
owner	<p>The AID of the Client Application owning this resource. The <i>Resource Ownership</i> is used to configure different ACR for the resource for the owner and other Client Applications.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPAID • Initial value: null • Administration: CSPAccessControl 	Resource Parameter
accessControlRules	<p>The Access Control Rules bit mask configured to a resource permits a specific entity to use the resource for designated cryptographic operations.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPAccessControlRules • Initial value: 0x0001 • Administration: CSPAccessControl 	Resource Parameter

5.3.3 Access Operations

The CSP does not offer any access-control-specific methods or commands.

5.3.4 Access Lifecycle

Access control is stateless and does not have a lifecycle.

5.3.5 Access Structures

This section lists ASN.1 structures related to access control settings.

Note: A complete ASN.1 definition is available in [GP CSP ASN1].

5.3.5.1 CSPClient

This data structure used for *Client Application Registration*.

It is part of the CSPConfiguration structure.

It can be created using the CSPRegisterClient command.

It can be removed using the CSPUnregisterClient command.

ASN 5-24: Core: ASN.1 Definition for CSPClient

```
-- ASN1START
-- Data structure representing a Client Application or an off-card Client.
CSPClient ::= SEQUENCE {

    -- An identifier of the CSP Client, chosen by the CSP Admin.
```

```

clientId          INTEGER (0..32767) OPTIONAL,

-- The Client Application registered to the CSP.
clientApplication  CSPClientApplication OPTIONAL,

-- Restrict access to CSP services when a sec channel is fully established.
authProtocol      CSPProtocolType OPTIONAL,

-- Resources required for the configured authProtocol.
authResources     SET OF CSPResourceId OPTIONAL
}
-- ASN1STOP

```

5.3.5.2 CSPClientReference

This data structure is used to identify a CSP Client, either by the AID of the Client Application or by an identifier assigned by the CSP Admin.

It is used within the CSPAccessControl structure and the CSPUnregisterClient command.

ASN 5-25: Core: ASN.1 Definition for CSPClientReference

```

-- ASN1START
-- The identifier of an off-card Client or the AID of an Client Application.
CSPClientReference ::= CHOICE {
    -- The identifier of the CSP Client to be de-registered.
    clientId          INTEGER (0..32767),

    -- The AID of the Client Application that shall be de-registered.
    aid              CSPAID
}
-- ASN1STOP

```

5.3.5.3 CSPClientApplication

This data structure represents a container for a Client Application.

It is part of the CSPClient structure.

ASN 5-26: Core: ASN.1 Definition for CSPClientApplication

```

-- ASN1START
-- Data structure representing a Client Application.
CSPClientApplication ::= SEQUENCE {

    -- AID of the Client Application.
    applicationAID    CSPAID,

    -- Hash of the load data file block of the Client Application.
    loadFileDataBlockHash OCTET STRING OPTIONAL,

```

```
-- Client Application must be DAP-verified during load.
requiredDAPVerification BOOLEAN DEFAULT FALSE,

-- AID of the SD of the Client Application.
applicationSD          CHOICE {
    -- The Client Application uses same SD AID as the CSP Application.
    useCspSD            BOOLEAN,

    -- The Client Application uses an other SD AID as the CSP Application.
    applicationSDAID    CSPAID
} OPTIONAL
}
-- ASN1STOP
```

5.3.5.4 CSPAccessControl

This data structure represents a container for a resource-specific *ACR Bit Positions* configuration.

It is part of the CSPResource structure.

It can be modified using the CSPConfigureResource command.

For further information, see section 3.2, Access Control, and section 3.2.3, Resource Ownership.

ASN 5-27: Access: ASN.1 Definition for CSPAccessControl

```
-- ASN1START
-- Data structure for access control configuration of a resource.
CSPAccessControl ::= SEQUENCE {

    -- ACR bitmask.
    accessControlRules    CSPAccessControlRules DEFAULT { any-use },

    -- CSP Client that owns this resource (default: null).
    owner                 CSPClientReference OPTIONAL,

    -- Dynamic policy rules.
    policies               SET OF CSPPolicy OPTIONAL
}
-- ASN1STOP
```

5.3.5.5 CSPAccessControlRules

This data structure represents the *ACR Bit Positions* bitmask of a resource.

An ACR can be configured for a resource using the CSPAccessControl structure.

For further information, see section 3.2.4, Access Control Rules.

ASN 5-28: Access: ASN.1 Definition for CSPAccessControlRules

```
-- ASN1START
-- Bitmask to configure access rules for a resource.
CSPAccessControlRules ::= BIT STRING {

    -- ANY_USE:
    -- 1. Bit: all Client Applications are granted with ACCESS_USE.
    any-use          (0),

    -- ANY_SETUP:
    -- 2. Bit: all Client Applications are granted with ACCESS_SETUP.
    any-setup        (1),

    -- ANY_CLEAR:
    -- 3. Bit: all Client Applications are granted with ACCESS_CLEAR.
    any-clear        (2),

    -- ANY_MOVE:
    -- 4. Bit: all Client Applications are granted with ACCESS_MOVE.
    any-move         (3),

    -- OWNER_USE:
    -- 5. Bit: only Owner Application is granted with ACCESS_USE.
    owner-use        (4),

    -- OWNER_SETUP:
    -- 6. Bit: only Owner Application is granted with ACCESS_SETUP.
    owner-setup      (5),

    -- OWNER_CLEAR:
    -- 7. Bit: only Owner Application is granted with ACCESS_CLEAR.
    owner-clear      (6),

    -- OWNER_MOVE:
    -- 8. Bit: only Owner Application is granted with ACCESS_MOVE.
    owner-move       (7),

    -- ADMIN_USE:
    -- 9. Bit: the CSP Admin is granted with ACCESS_USE.
    admin-use        (8),

    -- ADMIN_SETUP:
    -- 10. Bit: the CSP Admin is granted with ACCESS_SETUP.
    admin-setup      (9),

    -- ADMIN_CLEAR:
```

```
-- 11. Bit: the CSP Admin is granted with ACCESS_CLEAR.  
admin-clear      (10),  
  
-- ADMIN_MOVE:  
-- 12. Bit: the CSP Admin is granted with ACCESS_MOVE.  
admin-move      (11),  
  
-- CLIENT_USE:  
-- 13. Bit: all off-card Clients are granted with ACCESS_USE.  
client-use      (12),  
  
-- CLIENT_SETUP:  
-- 14. Bit: all off-card Clients are granted with ACCESS_SETUP.  
client-setup    (13),  
  
-- CLIENT_CLEAR:  
-- 15. Bit: all off-card Clients are granted with ACCESS_CLEAR.  
client-clear    (14),  
  
-- CLIENT_MOVE:  
-- 16. Bit: all off-card Client are granted with ACCESS_MOVE.  
client-move     (15)  
}  
-- ASN1STOP
```

6 OPTIONAL MODULES

6.1 Cipher Module

The CSP may implement the `CipherModule` to offer cipher services, including encryption and decryption as part of *Cipher and Signatures*; the choice shall be subject to *Modularity*.

The CSP Admin shall be able to create resources with `USAGE_CIPHER` and configure cipher using the `CSPCipherAlgorithms` structure to select:

- The `paddingAlgorithm` from available *Padding Algorithms*.
- The `cipherAlgorithm` from available *Cipher Algorithms*.

The Client Application shall be able to pass the `resourceId` referring to this cipher configuration to the `cipher.init` operation and trigger:

- Cipher processes using the `cipher.doFinal` operation.
- Multi-part cipher processes using the `cipher.update` operation.

The CSP Admin may detect whether the platform supports this module using the `CSPEnforce` command.

For further information, see section 3.6, *Cipher and Signatures*.

6.1.1 Cipher Definitions

6.1.1.1 Cipher Modes

The `CipherModule` shall support the cipher operation modes listed in Table 6-1. This mode defines if the cipher service operates in encryption or decryption mode.

The Client Application may select the `cipherMode` using the `cipher.init` operation.

Table 6-1: Cipher Modes

Mode	Value (Hex)	Value (Int)	Description
<code>CIPHER_MODE_DECRYPT</code>	<code>0x01</code>	1	Decrypt incoming data. The <code>CipherModule</code> shall decrypts the provided input buffers.
<code>CIPHER_MODE_ENCRYPT</code>	<code>0x02</code>	2	Encrypt incoming data. The <code>CipherModule</code> shall encrypt the provided input buffers.

6.1.1.2 Padding Algorithms

The `CipherModule` may support the algorithms listed in Table 6-2; the exact list shall be subject to *Modularity*.

The CSP Admin may configure the `paddingAlgorithm` using the `CSPPaddingAlgorithm` structure.

The CSP Admin may detect if a padding algorithm is supported using the `CSPCipherSupport` structure. Each algorithm supported shall adhere to the algorithm combinations specified in

- Table 6-4 for compatible *Cipher, Padding, and Key Size Combinations*
- Table 6-16 for compatible *Signature, Padding, Hash, Key Size, and Curve Combinations*

For further information, see section 3.6.1, *Padding Schemes*, and section 3.7.3 *Building Blocks*.

Table 6-2: Padding Algorithms

Algorithm	Value (Hex)	Value (Int)	Description
PAD_NULL ¹ (No Padding Required)	0x00	0	Specifies that no padding is applied to the data, typically used with signature algorithms that do not need block-alignment.
PAD_NOPAD ² (Exact-Length Required)	0x01	1	Specifies that no padding is applied to the data, specifically for algorithms that normally require block-alignment. This type requires the data length to exactly match the cipher algorithm's block size; any misalignment will fail with <code>ERROR_ILLEGAL_VALUE</code> .
PAD_ISO9797_1_M2_ALG3	0x05	5	Padding is applied according to ISO 9797-1 method 2 ([ISO 9797-1]).
PAD_PKCS1 ³ (legacy)	0x07	7	Padding based on the PKCS#1 (v1.5) scheme ([RFC 8017]). Note: Considered legacy due to vulnerability to padding oracle attacks.
PAD_PKCS1_PSS	0x08	8	Padding based on the PKCS#1-PSS scheme ([IEEE 1363-2000]).
PAD_PKCS1_OAEP_SHA256	0x0E	14	Padding based on the PKCS#1-OAEP scheme ([IEEE 1363-2000]) with SHA256 as hash function.
PAD_PKCS1_OAEP_SHA384	0x0F	15	Padding based on the PKCS#1-OAEP scheme ([IEEE 1363-2000]) with SHA384 as hash function.
PAD_PKCS1_OAEP_SHA512	0x10	16	Padding based on the PKCS#1-OAEP scheme ([IEEE 1363-2000]) with SHA512 as hash function.
PAD_PKCS1_OAEP_SHA3_256	0x12	18	Padding based on the PKCS#1-OAEP scheme ([IEEE 1363-2000]) with SHA3-256 as hash function.
PAD_PKCS1_OAEP_SHA3_384	0x13	19	Padding based on the PKCS#1-OAEP scheme ([IEEE 1363-2000]) with SHA3-384 as hash function.
PAD_PKCS1_OAEP_SHA3_512	0x14	20	Padding based on the PKCS#1-OAEP scheme ([IEEE 1363-2000]) with SHA3-512 as hash function.
PAD_PKCS7	0x02	2	Padding according to PKCS#7 ([RFC 5652]).

6.1.1.3 Cipher Algorithms

The CipherModule may support the algorithms listed in Table 6-3; the exact list shall be subject to *Modularity*.

The CSP Admin may configure the cipherAlgorithm using the CSPCipherAlgorithm structure.

The CSP shall enforce access restrictions on cipher algorithm computations, as specified in:

- Table 6-10 for *Access Restrictions on Cipher Operations*
- Table 6-26 for *Access Restrictions on Transform Operations*
- Table 6-48 for *Access Restrictions on Confidential Data Transfer Operations*

¹ PAD_NULL is mandatory, as it is required for *Config Attestation* (see section 3.8.2).

² PAD_NOPAD for CIPHER_RSA and SIG_RSA is considered as *Legacy Cryptographic Algorithms* (see section 2.4.2).

³ PAD_PKCS1 is considered as *Legacy Cryptographic Algorithms* (see section 2.4.2).

The CSP Admin may detect if a cipher algorithm is supported using the CSPCipherSupport structure. Each cipher algorithm supported shall adhere to the algorithm combinations specified in Table 6-4 for compatible *Cipher, Padding, and Key Size* Combinations

The CSP shall support keys being used up to the usage limits listed in Table 6-3. These limits apply per cipher block and define when a key should no longer be used, so as to avoid security risks such as key exhaustion or statistical attacks.

Usage limits may be enforced by the CSP through usage counters or by external organizational controls. Counters should not be used redundantly if other mechanisms ensure compliance with the defined limits. However, CSP Admins and Client Applications should consider these limits during use-case planning, ensuring keys are replaced or rotated before reaching the specified limits.

For further information, see section 3.6.3, Cipher, section 3.6.4, Cipher Block Modes, section 3.7.3, Building Blocks, and section 3.7.4, Confidential Data Transfer.

Table 6-3: Cipher Algorithms

Algorithm	Value (Hex)	Value (Int)	Key Usage Limit per block	Description
CIPHER_AES_CBC ⁴	0x01	1	100,000	Cipher using AES ([FIPS 197]) with block size of 128 bytes in CBC mode ([FIPS 81]).
CIPHER_AES_CFB	0x1C	28	100,000	Cipher using AES ([FIPS 197]) in Cipher Feedback (CFB) mode ([FIPS 81]).
CIPHER_AES_CTR	0xF0	240	100,000	Cipher using AES ([FIPS 197]) in counter (CTR) mode ([ISO 10116]).
CIPHER_AES_GCM	0xF1	241	100,000	Cipher using AES ([FIPS 197]) Galois/Counter Mode [SP800-38D].
CIPHER_AES_CCM	0xF2	242	100,000	Cipher using AES ([FIPS 197]) in Counter with CBC-MAC mode ([RFC 3610] / [SP800-38C]).
CIPHER_AES_XTS	0x0A	10	100,000	Cipher using AES ([FIPS 197]) in XEX Tweakable Block Cipher with Ciphertext Stealing (XTS) mode as defined in [IEEE 1619-2018]. Note: The two key values required by AES-XTS have to be provided in one CSP resource. The CSP is parsing the resource value as a concatenation of two fields of equal size Key1 and Key2 such that Key = Key1 Key2. Key size 256 should be used to have Key1 and Key2 each with 128-bit length. Key size 512 should be used to have Key1 and Key2 each with 256-bit length.
CIPHER_RSA	0x07	7	10,000	Cipher using RSA ([PKCS #1]).

⁴ CIPHER_AES_CBC is mandatory, as it is required for PROTOCOL_PACE (see section 6.4.1.1).

6.1.1.4 Cipher, Padding, and Key Size Combinations

The CipherModule shall support the compatible combinations of *Cipher Algorithms* listed in Table 6-4 for all *Padding Algorithms*, *Key Types*, *Certificate Types*, and *Key Sizes* supported by the platform. Unsupported or illegal combinations shall be rejected and handled according to the errorMode configured.

For further information, see section 3.6.3, Cipher, and section 6.1.3.1, Cipher Restrictions.

Table 6-4: Cipher, Padding, and Key Size Combinations

Cipher Algorithm	Padding Algorithm	Key Type	Key Size
CIPHER_AES_CBC	PAD_NOPAD (Exact-Length Required)	KEY_AES	128 bit
	PAD_PKCS7		256 bit
CIPHER_AES_CFB	PAD_NULL (No Padding Required)	KEY_AES	
CIPHER_AES_CTR	PAD_NULL (No Padding Required)	KEY_AES	
CIPHER_AES_GCM	PAD_NULL (No Padding Required)	KEY_AES	
CIPHER_AES_CCM	PAD_NULL (No Padding Required)	KEY_AES	
CIPHER_AES_XTS	PAD_NULL (No Padding Required)	KEY_AES	2x128 bit (see KEY_AES_2_128) 2x256 bit (see KEY_AES_2_256)
CIPHER_RSA	PAD_PKCS1_OAEP_SHA256	KEY_RSA_PUBLIC	2048 bit ⁵ (legacy)
	PAD_PKCS1_OAEP_SHA384	KEY_RSA_PRIVATE	3072 bit
	PAD_PKCS1_OAEP_SHA512		4096 bit
	PAD_PKCS1_OAEP_SHA3_256		
	PAD_PKCS1_OAEP_SHA3_384		
	PAD_PKCS1_OAEP_SHA3_512		
	PAD_PKCS1 ⁶ (legacy)		
	PAD_NOPAD ⁷ (deprecated)		

6.1.1.5 Cipher Initialization Data

The CipherModule shall support initialization data for cipher algorithms as specified in Table 6-5. Client Applications shall provide and manage this input data as specified in the table.

For further information, see section 3.6.4, Cipher Block Modes.

⁵ CIPHER_RSA with KEY_RSA_2048 bit is considered as *Legacy Cryptographic Algorithms* (see section 2.4.2).

⁶ CIPHER_RSA with PAD_PKCS1 (v1.5) is considered as *Legacy Cryptographic Algorithms* (see section 2.4.2).

⁷ CIPHER_RSA with PAD_NOPAD is considered as *Legacy Cryptographic Algorithms* (see section 2.4.2).

Table 6-5: Cipher Initialization Data

Algorithm	Initialization Data
CIPHER_AES_CBC	Requires an unpredictable 128-bit initialization vector (IV). The IV must be unique per <code>cipher.init</code> call, and must be randomly generated.
CIPHER_AES_CFB	Requires an unpredictable 128-bit initialization vector (IV). The IV must be unique per <code>cipher.init</code> call, and must be randomly generated.
CIPHER_AES_CTR	Requires a 64 to 128-bit nonce. The nonce must be unique per <code>cipher.init</code> call, and must be randomly generated or derived. The counter is handled by the CSP.
CIPHER_AES_GCM	Requires a 96-bit nonce and optional Additional Authenticated Data (AAD). The nonce must be unique per <code>cipher.init</code> call, and must be randomly generated or derived. The counter is handled by the CSP.
CIPHER_AES_CCM	Requires a 7 to 13-byte nonce, according to the formatting function specified in Appendix A.1 of [SP800-38C], and optional Additional Authenticated Data (AAD). The nonce must be unique per <code>cipher.init</code> call, and must be randomly generated or derived. The counter is handled by the CSP.
CIPHER_AES_XTS	Requires a 128-bit tweak. The tweak must be unique per <code>cipher.init</code> call, and must be randomly generated or securely derived.
CIPHER_RSA	Depends on padding scheme: OAEP requires a random seed, and PSS requires a salt. Both seed and salt must be unique per <code>cipher.init</code> call, must be randomly generated, and must be equal to the hash output size (e.g., 256 bit for SHA-256).

6.1.1.6 Cipher Events

The AuditModule may log the cipher-related *Audit Events* listed in Table 6-6; the exact list shall be subject to *Modularity*. The event-specific data listed in the table shall be stored according to the configured `auditMode`.

The CSP Admin may configure the `resourceEvents` to be logged using `CSPResourceEvent`.

The CSP Admin may detect if an event type is supported using the `CSPAuditSupport` structure.

For further information, see section 3.11.4, Selecting Audit Events, and section 6.12.1.2, Audit Events.

Table 6-6: Cipher Events for Auditing

Event Type	Value (Hex)	Value (Int)	Description	Event Category
EVENT_CIPHER_ENCRYPTED	0x1010	4112	<p>Data encrypted successfully. This event is logged after successful invocations of the final cipher operation for cipher encryption mode.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Message Format: <code>CSPLogMessage</code> Operations: <code>cipher.doFinal</code> Event Data: <code>CSPEventDataResource</code> 	Resource Event

Event Type	Value (Hex)	Value (Int)	Description	Event Category
EVENT_CIPHER_DECRYPTED	0x1011	4113	Data decrypted successfully. This event is logged after successful invocations of the final cipher operation for cipher decryption mode. Technical Details: <ul style="list-style-type: none"> • Message Format: CSPLogMessage • Operations: cipher.doFinal • Event Data: CSPEventDataResource 	Resource Event

6.1.1.7 Cipher Error Codes

If ERROR_MODE_DETAILED is supported and activated, the CipherModule shall use the error codes listed in Table 6-7 as reasons for exceptions.

For further information, see section 5.1.1.6, Error Types.

Table 6-7: Cipher Error Reason Codes

Reason	Description
0x2010	Unknown cipher mode: The cipher mode provided is not defined.
0x2011	Illegal resource type: Only key resources can be used for cipher operations.
0x2012	Invalid initialization data: Initialization data not supported by the algorithm.
0x2013	Invalid initialization data length: Incorrect byte length of the algorithm-specific initialization data.
0x2014	Unsupported tag length: The requested tag length is not supported for this Authenticated Encryption with Associated Data (AEAD) cipher.
0x2015	AAD update failed: Total Additional Associated Data (AAD) length specified in the cipher.init method is not identical to the AAD length.
0x3010	Inconsistent config: Padding, key type, or key size incompatible with the cipher algorithm.
0x4010	Cipher not initialized: The service has not been successfully initialized through one of the init methods.
0x4011	Cipher not initialized: This Authenticated Encryption with Associated Data (AEAD) cipher operates in offline mode and requires nonce, AAD length, and message length to be specified during initialization.
0x4012	Cipher not initialized: This Authenticated Encryption with Associated Data (AEAD) cipher operates in online mode; specifying nonce, AAD length, and message length during initialization is not supported.
0x5010	Wrong usage: Resource not configured for USAGE_CIPHER.
0x6010	Not block aligned: PAD_NOPAD configured, but the input data is not a multiple of the block size.
0x6011	Missing input: PAD_NOPAD or PAD_NULL configured and no input data was provided.
0x6012	Invalid input: The input data is not supported by the algorithm.
0x6013	Modulus exceeded: The input value is greater than or equal to the modulus.
0x6014	Decryption failed: The decrypted data lacks the expected padding, indicating a decryption issue.
0x6015	CCM cipher failed: Additional Associated Data (AAD) was not provided in the init method
0x6016	CCM cipher failed: Total message length specified in the cipher.init method is not identical to the message length.
0x6017	AEAD tag retrieval failed: doFinal not yet called.

Reason	Description
0x6018	AAD update failed: Updating Additional Associated Data (AAD) is not permitted in the current cipher state.
0x8010	Unsupported: The cipher module is not supported.
0x8011	Unsupported: The padding algorithm is not supported.
0x8012	Unsupported: The cipher algorithm is not supported.

6.1.2 Cipher Configuration

The CipherModule shall support the configuration parameters defined in Table 6-1.

Table 6-8: Cipher Configuration Parameters

Parameter	Description	Parameter Type
cipherMode	Operation mode of the CipherModule, either encryption or decryption mode, selected from the available <i>Cipher Modes</i> . Technical Details: <ul style="list-style-type: none"> Data type: byte Operations: cipher.init 	Service Parameter
cipherAlgorithm	Cipher algorithm configured for a resource, selected from the available <i>Cipher Algorithms</i> . Technical Details: <ul style="list-style-type: none"> Data type: CSPCipherAlgorithm Administration: CSPCipherAlgorithms 	Resource Parameter
paddingAlgorithm	Padding configured for a resource, selected from the available <i>Padding Algorithms</i> . Technical Details: <ul style="list-style-type: none"> Data type: CSPPaddingAlgorithm Administration: CSPCipherAlgorithms 	Resource Parameter

6.1.3 Cipher Operations

The CipherModule shall implement the operations listed in Table 6-9.

For each operation, the CSP shall handle supported *Counter Types*, *Timer Types*, and *Audit Events*.

The operations shall solely be utilizable with resources configured for USAGE_CIPHER and shall comply with the *Access Restrictions on Cipher Operations* specified in Table 6-10.

Note: Specifically, resources with USAGE_TRANSFORM, USAGE_SECCHANNEL, USAGE_CONFIDENTIAL, or USAGE_OFFLOADING, dedicated for use in *Transform Operations*, *Secure Channel Operations*, *Confidential Data Transfer Operations*, and *Offloading Operations*, shall not be permitted for decryption activities through this module.

For further information, see section 3.6, Cipher and Signatures, section 3.6.8, Encryption Transformation, and section 3.7.3, Building Blocks.

Table 6-9: Cipher Operations

Operation	Details
<code>cipher.init</code>	<p>Initializes this service using a key resource configured with a specific cipher configuration selected from available <i>Cipher Algorithms</i> and <i>Padding Algorithms</i>, and sets the <code>cipherMode</code> to either encrypt or decrypt, as specified in <i>Cipher Modes</i>.</p> <p>CSP API:</p> <ul style="list-style-type: none"> <code>org.globalplatform.csp.api.CipherService.init(..)</code> <p>Evaluate Timers:</p> <ul style="list-style-type: none"> <code>TIMER_VALIDITY_DATE</code>, <code>TIMER_VALIDITY_PERIOD</code> <p>Fire Events:</p> <ul style="list-style-type: none"> <code>EVENT_TIMER_EXPIRED</code>, <code>EVENT_CSP_ERROR</code>
<code>cipher.doFinal</code>	<p>Depending on the <code>cipherMode</code>, this method decrypts or encrypts the provided input data.</p> <p>CSP API:</p> <ul style="list-style-type: none"> <code>org.globalplatform.csp.api.CipherService.doFinal(..)</code> <p>CSP Protocol:</p> <ul style="list-style-type: none"> <code>CSPEncrypt</code> <p>Increment Counters:</p> <ul style="list-style-type: none"> <code>COUNT_USAGE</code>, <code>COUNT_USAGE_PER_BLOCK</code> <code>COUNT_USAGE_SUCCESS_ONLY</code>, <code>COUNT_USAGE_FAILURE_ONLY</code> <p>Fire Events:</p> <ul style="list-style-type: none"> <code>EVENT_CIPHER_ENCRYPTED</code>, <code>EVENT_CIPHER_DECRYPTED</code> <code>EVENT_COUNTER_EXHAUSTED</code>, <code>EVENT_CSP_ERROR</code>
<code>cipher.update</code>	<p>Optional method to handle multi-part encryption or decryption by processing a chunk of data. The method can be invoked multiple times for sequential data processing and must be concluded with the <code>cipher.doFinal</code> method.</p> <p>CSP API:</p> <ul style="list-style-type: none"> <code>org.globalplatform.csp.api.CipherService.update(..)</code> <p>Increment Counters:</p> <ul style="list-style-type: none"> <code>COUNT_USAGE_PER_BLOCK</code> <p>Fire Events:</p> <ul style="list-style-type: none"> <code>EVENT_COUNTER_EXHAUSTED</code>, <code>EVENT_CSP_ERROR</code>
<code>cipher.updateAAD</code>	<p>Multipart update of the Additional Associated Data (AAD) for AEAD cipher.</p> <p>CSP API:</p> <ul style="list-style-type: none"> <code>org.globalplatform.csp.api.CipherService.updateAAD(..)</code>
<code>cipher.retrieveTag</code>	Retrieves the authentication tag for AEAD cipher.
<code>cipher.verifyTag</code>	Verifies the provided authentication tag.

6.1.3.1 Access Restrictions on Cipher Operations

The `CipherModule` shall enforce the access restrictions for *Cipher Operations* listed in Table 6-10 for all *Access Rights*, *Usage Types*, *Resource States*, and *Policy Types* supported by the platform. Unauthorized or illegal operation invocations shall be rejected and handled according to the `errorMode` configured.

Table 6-10: Cipher Operations Access Restrictions

Operation	Right Required	Usage Required	State Required	Other Restrictions
cipher.init	USE	CIPHER	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required Only consistent Cipher, Padding, and Key Size Combinations. POLICY_PASSWORD POLICY_SECCHANNEL_ESTABLISHED POLICY_TA2_ACCESS_FLAG
cipher.update	-	-	OPERATIONAL	
cipher.doFinal	-	-	OPERATIONAL	
cipher.updateAAD cipher.retrieveTag cipher.verifyTag	-	-	-	<ul style="list-style-type: none"> Only cipher algorithms supporting Authenticated Encryption with Associated Data (AEAD).

6.1.3.2 Sensitive Results Computed by Cipher Operations

The CipherModule shall temporarily store the results of methods listed in Table 6-11 in transient memory and shall implement the assertSensitiveResult operation, enabling Client Applications to verify that these results were computed without any abnormality.

Table 6-11: Cipher Operations Requiring Sensitive Results Checks

Operation	Result Type	Result Description
cipher.update (CIPHER_MODE_DECRYPT)	short	The length of the decrypted data.
cipher.doFinal (CIPHER_MODE_DECRYPT)	short	The length of the decrypted data.
cipher.retrieveTag (CIPHER_MODE_ENCRYPT)	short	The length of the authentication tag returned.
cipher.verifyTag (CIPHER_MODE_DECRYPT)	short (CSPBoolean)	The result of the authentication tag verification.

6.1.3.3 Sensitive Arrays Required for Cipher Operations

The CipherModule shall invoke javacard.framework.SensitiveArrays.assertIntegrity(..) ([JCAPI]) on the parameters listed in Table 6-12.

For further information, see section 5.1.3.4, Sensitive Arrays, and section 8.3, Byte Buffer Parameters.

Table 6-12: Cipher Operations Requiring Sensitive Arrays

Operation	Mode	Parameter	Description
cipher.init	-	Input buffer	Algorithm-specific initialization data.
cipher.updateAAD	-	Input buffer	Input buffer containing Additional Associated Data (AAD).
cipher.retrieveTag	CIPHER_MODE_ENCRYPT	Output buffer	Computed authentication tag after processing.
cipher.verifyTag	CIPHER_MODE_DECRYPT	Input buffer	Authentication tag to verify.
cipher.update	CIPHER_MODE_ENCRYPT	Input buffer	Data block to encrypt or decrypt (multi-part).
cipher.update	-	Output buffer	The decrypted or encrypted data block (multi-part).

Operation	Mode	Parameter	Description
cipher.doFinal	CIPHER_MODE_ENCRYPT	Input buffer	Data to encrypt or decrypt.
cipher.doFinal	CIPHER_MODE_DECRYPT	Output buffer	The decrypted or encrypted data.

6.1.4 Cipher Lifecycle

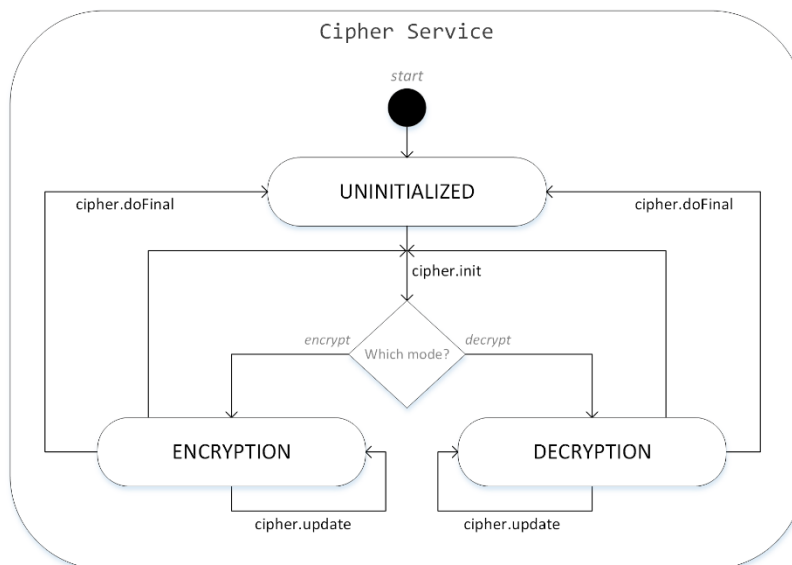
Figure 6-1 shows lifecycle changes of a cipher service triggered by *Cipher Operations*. A cipher service is an instance of the CipherModule. When a new instance is created, it starts in the UNINITIALIZED state. In this state, it is not possible to perform any cipher operations. The Client Application needs to invoke the cipher.init operation to set the cipher service to either CIPHER_MODE_ENCRYPT or CIPHER_MODE_DECRYPT mode before any cipher process can be triggered. After completing the cipher.doFinal operation, the cipher service will return to the UNINITIALIZED state.

The cipher service stores *Sensitive Results Computed by Cipher Operations*.

The cipher service triggers lifecycle changes to the key resources used, as specified in

- Section 6.7.4 Key Lifecycle

Figure 6-1: Cipher Lifecycle



6.1.5 Cipher Structures

This section lists ASN.1 structures related to the configuration of the cipher service.

Note: A complete ASN.1 definition is available in [GP CSP ASN1].

6.1.5.1 CSPCipherSupport

This data structure represents features, types, and algorithms of the *Cipher Module*.

It is included in the CSPEnforce command to detect their support by the platform.

ASN 6-1: Cipher: ASN.1 Definition for CSPCipherSupport

```

-- ASN1START
-- Checks for cipher functionality support and supported algorithms.
CSPCipherSupport ::= SEQUENCE {

```

```
-- Padding algorithms.
paddingAlgorithms      SET OF CSPPaddingAlgorithm OPTIONAL,

-- Cipher algorithms.
cipherAlgorithms       SET OF CSPCipherAlgorithm OPTIONAL
}
-- ASN1STOP
```

6.1.5.2 CSPCipherAlgorithms

This data structure represents a container for a resource-specific *Cipher Configuration*.

It can be configured for a resource using the CSPAlgorithms structure.

For further information, see section 3.6, Cipher and Signatures.

ASN 6-2: Cipher: ASN.1 Definition for CSPCipherAlgorithms

```
-- ASN1START
-- Cipher configuration for a specific resource.
CSPCipherAlgorithms ::= SEQUENCE {

    -- The cipher algorithm.
    cipherAlgorithm      CSPCipherAlgorithm,

    -- The padding algorithm.
    paddingAlgorithm     CSPPaddingAlgorithm DEFAULT pad-null
}
-- ASN1STOP
```

6.1.5.3 CSPPaddingAlgorithm

This data structure defines the list of available *Padding Algorithms*.

A padding algorithm can be selected using CSPCipherAlgorithms or CSPSignatureAlgorithms

The padding algorithms supported can be detected using the CSPCipherSupport.

ASN 6-3: Cipher: ASN.1 Definition for CSPPaddingAlgorithm

```
-- ASN1START
-- List of available padding algorithms.
CSPPaddingAlgorithm ::= ENUMERATED {

    -- PAD_NULL (No Padding Required):
    -- No padding; for use when padding is not supported by the algorithm.
    pad-null              (0),

    -- PAD_NOPAD (Exact-Length Required):
    -- No padding is applied to the data, even when the algorithm usually pads.
    pad-nopad             (1),
```

```

-- PAD_ISO9797_1_M2_ALG3:
-- Padding based on the ISO 9797-1 MAC algo 3 with method 2 [ISO 9797-1].
pad-iso9797-1-m2-alg3      (5),

-- PAD_PKCS1:
-- Padding based on the PKCS#1 v1.5 scheme [RFC 8017].
pad-pkcs1                  (7),

-- PAD_PKCS1_PSS:
-- Padding based on the PKCS#1-PSS scheme [IEEE 1363-2000].
pad-pkcs1-pss              (8),

-- PAD_PKCS1_OAEP_SHA256:
-- Padding based on the PKCS#1-OAEP scheme [IEEE 1363-2000] with SHA256.
pad-pkcs1-oaep-sha256     (14),

-- PAD_PKCS1_OAEP_SHA384:
-- Padding based on the PKCS#1-OAEP scheme [IEEE 1363-2000] with SHA384.
pad-pkcs1-oaep-sha384     (15),

-- PAD_PKCS1_OAEP_SHA512:
-- Padding based on the PKCS#1-OAEP scheme [IEEE 1363-2000] with SHA512.
pad-pkcs1-oaep-sha512     (16),

-- PAD_PKCS1_OAEP_SHA3_256:
-- Padding based on the PKCS#1-OAEP scheme [IEEE 1363-2000] with SHA3-256.
pad-pkcs1-oaep-sha3-256   (18),

-- PAD_PKCS1_OAEP_SHA3_384:
-- Padding based on the PKCS#1-OAEP scheme [IEEE 1363-2000] with SHA3-384.
pad-pkcs1-oaep-sha3-384   (19),

-- PAD_PKCS1_OAEP_SHA3_512:
-- Padding based on the PKCS#1-OAEP scheme [IEEE 1363-2000] with SHA3-512.
pad-pkcs1-oaep-sha3-512   (20),

-- PAD_PKCS7:
-- Padding based on the PKCS#7 scheme [RFC 5652].
pad-pkcs7                  (2)
}
-- ASN1STOP

```

6.1.5.4 CSPCipherAlgorithm

This data structure defines the list of available *Cipher Algorithms*.

A cipher algorithm can be selected using the CSPCipherAlgorithms structure.

The cipher algorithms supported can be detected using the CSPCipherSupport.

ASN 6-4: Cipher: ASN.1 Definition for CSPCipherAlgorithm

```
-- ASN1START
-- List of available cipher algorithms.
CSPCipherAlgorithm ::= ENUMERATED {

    -- CIPHER_AES_CBC:
    -- Cipher using AES [FIPS 197] with block size 128 in CBC mode [FIPS 81].
    cipher-aes-cbc          (1),

    -- CIPHER_AES_CFB:
    -- Cipher using AES [FIPS 197] in Cipher Feedback (CFB) mode [FIPS 81].
    cipher-aes-cfb          (28),

    -- CIPHER_AES_CTR:
    -- Cipher using AES [FIPS 197] in counter (CTR) mode [ISO 10116].
    cipher-aes-ctr          (240),

    -- CIPHER_AES_GCM:
    -- Cipher using AES [FIPS 197] Galois/Counter Mode [SP800-38D].
    cipher-aes-gcm          (241),

    -- CIPHER_AES_CCM:
    -- Cipher using AES [FIPS 197] in Counter with CBC-MAC mode [SP800-38C].
    cipher-aes-ccm          (242),

    -- CIPHER_AES_XTS:
    -- Cipher using AES [FIPS 197] in XTS mode in [IEEE 1619-2018].
    cipher-aes-xts          (10),

    -- CIPHER_RSA:
    -- Cipher using RSA [PKCS #1].
    cipher-rsa              (7)
}
-- ASN1STOP
```

6.2 Signature Module

The CSP may implement the `SignatureModule` to offer signature services, including signature creation and verification as part of *Cipher and Signatures*; the choice shall be subject to *Modularity*.

The CSP Admin shall be able to create resources with `USAGE_SIGNATURE` and configure signing using the `CSPSignatureAlgorithm` structure to select:

- The `paddingAlgorithm` from available *Padding Algorithms*.
- The `signatureAlgorithm` from available *Signature Algorithms*.
- The `messageDigestAlgorithm` from available *Message Digest Algorithms*.

The Client Application shall be able to pass the `resourceId` referring to this signature configuration to the `sig.init` operation and trigger:

- Signing processes using the `sig.sign` operation.
- Signature verification processes using the `sig.verify` operation.
- Multi-part signing or verification processes using the `sig.update` operation.

The CSP Admin may detect whether the platform supports this module using the `CSPEnforce` command.

For further information, see section 3.6, *Cipher and Signatures*.

6.2.1 Signature Definitions

6.2.1.1 Signature Modes

The `SignatureModule` shall support the signature operation modes listed in Table 6-13. This mode defines if the signature service operates in signature creation or signature verification mode.

The Client Application may select the `signatureMode` using the `sig.init` operation.

Table 6-13: Signature Modes

Mode	Value (Hex)	Value (Int)	Description
<code>SIG_MODE_SIGN</code>	<code>0x01</code>	1	Sign incoming data. The <code>SignatureModule</code> shall operate in signing mode, enabling it to sign the provided input buffers.
<code>SIG_MODE_VERIFY</code>	<code>0x02</code>	2	Verify the signature of the incoming data. The <code>SignatureModule</code> shall operate in verification mode, enabling it to verify the signature of the provided input buffers.

6.2.1.2 Message Digest Algorithms

The `SignatureModule` may support the message digest algorithms listed in Table 6-14; the exact list shall be subject to *Modularity*.

The CSP Admin may configure the `messageDigestAlgorithm` using `CSPMessageDigestAlgorithm`, or the `keyDerivationHashAlgorithm` using the `CSPKeyDerivationAlgorithms` structure.

The CSP Admin may detect if a message digest algorithm is supported using the `CSPSignatureSupport` structure. Each message digest algorithm supported shall adhere to the algorithm combinations specified in:

- Table 6-16 for compatible *Signature, Padding, Hash, Key Size, and Curve* Combinations
- Table 6-64 for compatible *Key Derivation, Resource Type, Hash, Size, and Curve* Combinations

For further information, see section 3.6.2, Hash Message Digest, and section 3.7.3, Building Blocks.

Table 6-14: Message Digest Algorithms

Algorithm	Value (Hex)	Value (Int)	Description
ALG_NULL	0x00	0	Specifies that no message digest is applied to the data.
ALG_SHA_256 ⁸	0x04	4	Message digest algorithm SHA-256 ([FIPS 180-4]) with block size of 64 bytes and intermediate hash value size of 32 bytes.
ALG_SHA_384	0x05	5	Message digest algorithm SHA-384 ([FIPS 180-4]) with block size of 128 bytes and intermediate hash value size of 64 bytes.
ALG_SHA_512	0x06	6	Message digest algorithm SHA-512 ([FIPS 180-4]) with block size of 128 bytes and intermediate hash value size of 64 bytes.
ALG_SHA3_256	0x09	9	Message digest algorithm SHA3-256 ([FIPS 180-4]) with block size of 64 bytes and intermediate hash value size of 32 bytes.
ALG_SHA3_384	0x0A	10	Message digest algorithm SHA3-384 ([FIPS 180-4]) with block size of 128 bytes and intermediate hash value size of 64 bytes.
ALG_SHA3_512	0x0B	11	Message digest algorithm SHA3-512 ([FIPS 180-4]) with block size of 128 bytes and intermediate hash value size of 64 bytes.

6.2.1.3 Signature Algorithms

The SignatureModule may support the signature algorithms listed in Table 6-15; the exact list shall be subject to *Modularity*.

The CSP Admin may configure the signatureAlgorithm using the CSPSignatureAlgorithm structure.

The CSP shall enforce access restrictions on signature algorithm computations, as specified in:

- Table 6-21 for *Access Restrictions on Signature Operations*
- Table 6-55 for *Access Restrictions on Attestation Operations*
- Table 6-124 for *Access Restrictions on Audit Operations*

The CSP Admin may detect if a signature algorithm is supported using the CSPSignatureSupport structure. Each signature algorithm supported shall adhere to the algorithm combinations specified in

- Table 6-16 for compatible *Signature, Padding, Hash, Key Size, and Curve* Combinations

For further information, see section 3.6.5, Signature Suites, section 3.7.3, Building Blocks, section 3.8.2, Config Attestation, and section 6.2.1.4, Signature Combinations.

⁸ ALG_SHA_256 is mandatory, as it is required for *Config Attestation* (see section 3.8.2).

Table 6-15: Signature Algorithms

Algorithm	Value (Hex)	Value (Int)	Description
SIG_AES_CMAC128 ⁹	0x0A	10	Create or verify signature according to [ISO 9797-1] by generating a 16-byte Cipher-based MAC (CMAC) using AES with block size of 128 bits in CBC mode.
SIG_AES_MAC128 ¹⁰ (legacy)	0x06	6	Create or verify signature according to [SP800-38B] by generating a 16-byte MAC using AES with block size of 128 bits in CBC mode, without padding the input data. If the input data is not aligned to a 16-byte block, the signature operation will fail with the error ERROR_ILLEGAL_VALUE.
SIG_HMAC	0x07	7	Create or verify a signature using HMAC according to [FIPS 198-1], which involve using a specified hashing algorithm.
SIG_RSA	0x03	3	Create or verify signature using RSA according to [IEEE 1363-2000] with optional pads according to PKCS#1 (v1.5) scheme or PKCS#1-PSS scheme.
SIG_ECDSA ¹¹	0x05	5	Create or verify signature according to [X9.62] using Elliptic Curve Digital Signature Algorithm (ECDSA) encoded as an ASN.1 sequence {r INTEGER, s INTEGER}.
SIG_ECDSA_PLAIN	0x09	9	Create or verify signature according to [TR-03111] section 5.2.1 using ECDSA encoded as octet string: r s.
SIG_EC_SCHNORR	0x0B	11	Create or verify signature using Elliptic Curve Based Schnorr Digital Signature Algorithm (ECSDSA) according to [TR-03111] section 4.2.3.
SIG_POA_FIAT_SHAMIR	0x0C	12	Create or verify a proof of association (PoA) for two public-private key pairs based on a Schnorr non-interactive zero-knowledge proof using the Fiat-Shamir heuristic [Fiat-Shamir], as specified in [Verheul-PoA] section 3. Both key pairs to be associated must use the same elliptic curve. Note 1: This PoA is similar to [RFC 8235].

6.2.1.4 Signature, Padding, Hash, Key Size, and Curve Combinations

The SignatureModule shall support the compatible combinations of *Signature Algorithms* listed in Table 6-16 for all *Padding Algorithms*, *Message Digest Algorithms*, *Resource Types*, *Certificate Types*, *Key Types*, *Key Sizes*, and *ECC Curves* supported by the platform. Unsupported or illegal combinations shall be rejected and handled according to the errorMode configured.

For further information, see section 3.6.5, Signature Suites, section 3.8.2, Config Attestation, and section 3.11.1, Creating Log Messages.

⁹ SIG_AES_CMAC128 is mandatory, as it is required for PROTOCOL_PACE (see section 6.4.1.1).

¹⁰ SIG_AES_MAC128 is considered as *Legacy Cryptographic Algorithms* (see section 2.4.2).

¹¹ SIG_ECDSA is mandatory since it is used for *Config Attestation* (see section 3.8.2).

Table 6-16: Signature, Padding, Hash, and Key Size Combinations

Signature Algorithm	Padding Algorithm	Message Digest Algorithm	Key Types	Key Size or Curve
SIG_AES_CMAC128	PAD_ISO9797_1_M2_ALG3	ALG_NULL	KEY_AES	128 bit
SIG_AES_MAC128 ¹² (legacy)	PAD_NOPAD (Exact-Length Required)	ALG_NULL	KEY_AES	128 bit
SIG_HMAC	PAD_NULL (No Padding Required)	ALG_SHA_256	KEY_HMAC	256 bit
		ALG_SHA_384		384 bit
		ALG_SHA_512		512 bit
		ALG_SHA3_256		256 bit
		ALG_SHA3_384		384 bit
		ALG_SHA3_512		512 bit
SIG_RSA	PAD_PKCS1_PSS	ALG_SHA_256	KEY_RSA_PUBLIC	2048 ¹⁵ bit (legacy)
	PAD_PKCS1 ¹³ (legacy)	ALG_SHA_384	KEY_RSA_PRIVATE	3072 bit
	PAD_NOPAD ¹⁴ (deprecated)	ALG_SHA_512		4096 bit
		ALG_SHA3_256		
		ALG_SHA3_384		
		ALG_SHA3_512		
SIG_ECDSA SIG_ECDSA_PLAIN SIG_EC_SCHNORR	PAD_NULL (No Padding Required)	ALG_SHA_256	KEY_ECC_PRIVATE	CURVE_SEC_P256_R1
			KEY_ECC_PUBLIC	CURVE_BRAINPOOL_P256_R1
		ALG_SHA_384		CURVE_SEC_P384_R1 CURVE_BRAINPOOL_P384_R1
		ALG_SHA_512		CURVE_SEC_P521_R1 CURVE_BRAINPOOL_P512_R1
		ALG_SHA3_256		CURVE_SEC_P256_R1 CURVE_BRAINPOOL_P256_R1
		ALG_SHA3_384		CURVE_SEC_P384_R1 CURVE_BRAINPOOL_P384_R1
		ALG_SHA3_512		CURVE_SEC_P521_R1 CURVE_BRAINPOOL_P512_R1

¹² SIG_AES_MAC128 is considered as *Legacy Cryptographic Algorithms* (see section 2.4.2).

¹³ PAD_PKCS1 is considered as *Legacy Cryptographic Algorithms* (see section 2.4.2).

¹⁴ SIG_RSA with PAD_NOPAD is considered as *Legacy Cryptographic Algorithms* (see section 2.4.2).

¹⁵ SIG_RSA with KEY_RSA_2048 bit is considered as *Legacy Cryptographic Algorithms* (see section 2.4.2).

Signature Algorithm	Padding Algorithm	Message Digest Algorithm	Key Types	Key Size or Curve
SIG_POA_FIAT_SHAMIR	PAD_NULL (No Padding Required)	ALG_SHA_256	KEY_ECC_PRIVATE KEY_ECC_PUBLIC	CURVE_SEC_P256_R1 CURVE_BRAINPOOL_P256_R1

6.2.1.5 Signature Events

The AuditModule may log the signature-related *Audit Events* listed in Table 6-17; the exact list shall be subject to *Modularity*. The event-specific data listed in the table shall be stored according to the configured auditMode.

The CSP Admin may configure the resourceEvents to be logged using CSPResourceEvent.

The CSP Admin may detect if an event type is supported using the CSPAuditSupport structure.

For further information, see section 3.11.4, Selecting Audit Events, and section 6.12.1.2, Audit Events.

Table 6-17: Signature Events for Auditing

Event Type	Value (Hex)	Value (Int)	Description	Event Category
EVENT_SIGNATURE_CREATED	0x1020	4128	Signature created successfully. This event is logged after successful invocations of signature creation operations. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Operations: sig.sign Event Data: CSPEventDataResource 	Resource Event
EVENT_SIGNATURE_VERIFIED	0x1021	4129	Signature verified successfully. This event is logged after successful invocations of signature verification operations. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Operations: sig.verify Event Data: CSPEventDataResource 	Resource Event
EVENT_SIGNATURE_VERIFICATION_FAILED	0x1022	4130	Signature verification failed. Logged when the signature could not be verified. Note: The event is not logged if an exception occurs. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Operations: sig.verify Event Data: CSPEventDataResource 	Resource Event

6.2.1.6 Signature Error Codes

If ERROR_MODE_DETAILED is supported and activated, the SignatureModule shall use the error codes listed in Table 6-18 as reasons for exceptions.

For further information, see section 5.1.1.6, Error Types.

Table 6-18: Signature Error Reason Codes

Reason	Description
0x2020	Unknown signature mode: The signature mode provided is not defined.
0x2021	Illegal resource type: Only key resources can be used for signature operations.
0x2022	Invalid initialization data: Initialization data not supported by the algorithm.
0x2023	Invalid initialization data length: Incorrect byte length of the algorithm-specific initialization data.
0x3020	Inconsistent config: Hash, padding, key type, size, or curve incompatible with the signature algorithm.
0x4020	Signing not initialized: The service has not been successfully initialized through one of the <code>init</code> methods.
0x4021	Verifying not initialized: The service has not been successfully initialized through one of the <code>init</code> methods.
0x5020	Wrong usage: Resource not configured for <code>USAGE_SIGNATURE</code> .
0x6020	Illegal input: The input data is not supported by the algorithm.
0x6021	Consistency check failed: A consistency check of the generated output failed.
0x6022	Invalid hash length: The provided hash length does not match the message digest algorithm's length.
0x6023	Illegal hash: No message digest computed before applying cryptographic operations (<code>ALG_NULL</code>).
0x8020	The signature module is not supported by the platform
0x8021	Unsupported: The message digest algorithm is not supported.
0x8022	Unsupported: The signature algorithm is not supported.

6.2.2 Signature Configuration

The `SignatureModule` shall support the configuration parameters defined in Table 6-19.

Table 6-19: Signature Configuration Parameters

Parameter	Description	Parameter Type
<code>signatureMode</code>	Operation mode of the <code>SignatureModule</code> , either signature creation or verification mode, selected from the available <i>Signature Modes</i> . Technical Details: <ul style="list-style-type: none"> Data type: <code>byte</code> Operations: <code>sig.init</code> 	Service Parameter
<code>signatureAlgorithm</code>	Signature algorithm configured for a resource, selected from the available <i>Signature Algorithms</i> . Technical Details: <ul style="list-style-type: none"> Data type: <code>CSPSignatureAlgorithm</code> Administration: <code>CSPSignatureAlgorithms</code> 	Resource Parameter
<code>paddingAlgorithm</code>	Padding configured for a resource, selected from the available <i>Padding Algorithms</i> . Technical Details: <ul style="list-style-type: none"> Data type: <code>CSPPaddingAlgorithm</code> Administration: <code>CSPSignatureAlgorithms</code> 	Resource Parameter

Parameter	Description	Parameter Type
messageDigestAlgorithm	Hash configured for a resource, selected from the available <i>Message Digest Algorithms</i> . Technical Details: <ul style="list-style-type: none"> • Data type: CSPMessageDigestAlgorithm • Administration: CSPSignatureAlgorithms 	Resource Parameter

6.2.3 Signature Operations

The SignatureModule shall implement the operations listed in Table 6-20.

For each operation, the CSP shall handle supported *Counter Types*, *Timer Types*, and *Audit Events*.

The operations shall solely be utilizable with resources configured for USAGE_SIGNATURE and shall comply with the *Access Restrictions on Signature Operations* specified in Table 6-21.

Note: Specifically, resources with USAGE_SECCHANNEL, USAGE_ATTESTATION, or USAGE_AUDIT, dedicated for use in *Secure Channel Operations*, *Attestation Operations*, and *Audit Operations*, shall not be permitted for signing activities through this module.

For further information, see section 3.6, Cipher and Signatures, and section 3.7.3 Building Blocks.

Table 6-20: Signature Operations

Operation	Details
sig.init	<p>Initializes this service using a key resource configured with a specific signature configuration selected from available <i>Signature Algorithms</i>, <i>Message Digest Algorithms</i>, <i>Padding Algorithms</i>, and <i>Fields</i>, and sets the signatureMode to either signature verification or signature creation, as specified in <i>Signature Modes</i>.</p> <p>CSP API:</p> <ul style="list-style-type: none"> • org.globalplatform.csp.api.SignatureService.init(..) <p>Evaluate Timers:</p> <ul style="list-style-type: none"> • TIMER_VALIDITY_DATE, TIMER_VALIDITY_PERIOD <p>Fire Events:</p> <ul style="list-style-type: none"> • EVENT_TIMER_EXPIRED, EVENT_CSP_ERROR
sig.sign	<p>Generates a signature using the given input data.</p> <p>If any <i>Fields</i> are configured for the signing resource, they shall be added to the data before it is signed, e.g., to create <i>Signatures with Counters & Timestamps</i>.</p> <p>CSP API:</p> <ul style="list-style-type: none"> • org.globalplatform.csp.api.SignatureService.sign(..) • org.globalplatform.csp.api.SignatureService.signPreComputedHash(..) <p>CSP Protocol:</p> <ul style="list-style-type: none"> • CSPSign <p>Increment Counters:</p> <ul style="list-style-type: none"> • COUNT_USAGE, COUNT_USAGE_PER_BLOCK • COUNT_USAGE_SUCCESS_ONLY, COUNT_USAGE_FAILURE_ONLY <p>Fire Events:</p> <ul style="list-style-type: none"> • EVENT_SIGNATURE_CREATED, EVENT_SIGNATURE_VERIFICATION_FAILED, EVENT_COUNTER_EXHAUSTED, EVENT_CSP_ERROR

Operation	Details
sig.verify	<p>Confirms the authenticity of a signature corresponding to the given input data.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.SignatureService.verify(..) org.globalplatform.csp.api.SignatureService.verifyPreComputedHash(..) <p>Increment Counters:</p> <ul style="list-style-type: none"> COUNT_USAGE_SUCCESS_ONLY, COUNT_USAGE_FAILURE_ONLY <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_SIGNATURE_VERIFIED, EVENT_SIGNATURE_VERIFICATION_FAILED EVENT_COUNTER_EXHAUSTED, EVENT_CSP_ERROR
sig.update	<p>Optional method to handle multi-part signature creation or verification through processing data chunks. It allows for multiple invocations to sequentially process data segments and requires a concluding call with either the sig.sign or sig.verify operation.</p> <p>If any <i>Fields</i> are configured for the signing resource, they shall be added to the data before it is signed, e.g., to create <i>Signatures with Counters & Timestamps</i>.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.SignatureService.update(..) <p>Increment Counters:</p> <ul style="list-style-type: none"> COUNT_USAGE_PER_BLOCK <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_COUNTER_EXHAUSTED, EVENT_CSP_ERROR

6.2.3.1 Access Restrictions on Signature Operations

The SignatureModule shall enforce the access restrictions for *Signature Operations* listed in Table 6-21 for all *Access Rights*, *Usage Types*, *Resource States*, and *Policy Types* supported by the platform. Unauthorized or illegal operation invocations shall be rejected and handled according to the *errorMode* configured.

Table 6-21: Signature Operations Access Restrictions

Operation	Right Required	Usage Required	State Required	Other Restrictions
sig.init	USE	SIGNATURE	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required Only consistent <i>Signature</i>, <i>Padding</i>, <i>Hash</i>, <i>Key Size</i>, and <i>Curve Combinations</i>. POLICY_SECCHANNEL_ESTABLISHED POLICY_PASSWORD POLICY_ASSOCIATION
sig.update	-	-	OPERATIONAL	<ul style="list-style-type: none"> Only possible in SIG_MODE_SIGN or SIG_MODE_VERIFY.
sig.sign	-	-	OPERATIONAL	<ul style="list-style-type: none"> Only possible in SIG_MODE_SIGN.
sig.verify	-	-	OPERATIONAL	<ul style="list-style-type: none"> Only possible in SIG_MODE_VERIFY.

6.2.3.2 Sensitive Results Computed by Signature Operations

The SignatureModule shall temporarily store the results of methods listed in Table 6-22 in transient memory and shall implement the *assertSensitiveResult* operation, enabling Client Applications to verify that these results were computed without any abnormality.

Table 6-22: Signature Operations Requiring Sensitive Results Checks

Operation	Result Type	Result Description
sig.sign	short	The length of the signed data.
sig.verify	short (CSPBoolean)	The result of the signature verification (true if valid, false otherwise).

6.2.3.3 Sensitive Arrays Required for Signature Operations

The `SignatureModule` shall invoke `javacard.framework.SensitiveArrays.assertIntegrity(...)` ([JCAPI]) on the parameters listed in Table 6-23.

For further information, see section 5.1.3.4, Sensitive Arrays, and section 8.3, Byte Buffer Parameters.

Table 6-23: Signature Operations Requiring Sensitive Arrays

Operation	Mode	Parameter	Description
sig.init	-	Input Buffer	Algorithm-specific initialization data.
sig.update	SIG_MODE_SIGN	Input buffer	Data block to sign or verify (multi-part).
sig.sign	SIG_MODE_SIGN	Input buffer	Final data to sign.
sig.sign	SIG_MODE_SIGN	Output buffer	The computed signature.
sig.verify	SIG_MODE_VERIFY	Input buffer	The data to verify.
sig.verify	SIG_MODE_VERIFY	Input buffer	The signature.

6.2.4 Signature Lifecycle

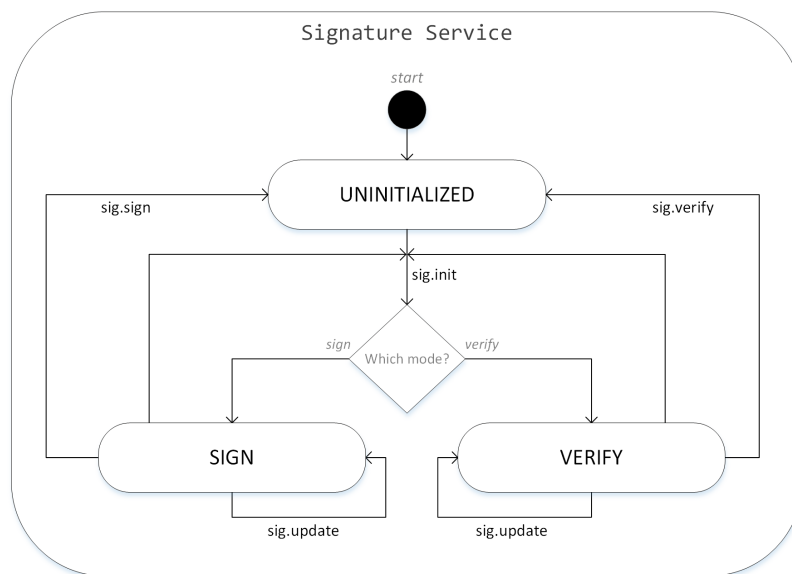
Figure 6-2 shows lifecycle changes of a signature service triggered by *Signature Operations*. A signature service is an instance of the `SignatureModule`. When a new instance is created, it starts in the UNINITIALIZED state. In this state, it is not possible to perform any signature operations. The Client Application needs to invoke the `sig.init` operation to set the signature service to either `SIG_MODE_SIGN` or `SIG_MODE_VERIFY` mode before any signature processes can be triggered. After completing a `sig.sign` or `sig.verify` operation, the signature service will return to the UNINITIALIZED state.

The signature service stores *Sensitive Results Computed by Signature Operations*.

The signature service triggers lifecycle changes to the key resources used, as specified in

- Section 6.7.4 *Key Lifecycle*

Figure 6-2: Signature Lifecycle



6.2.5 Signature Structures

This section lists ASN.1 structures related to the configuration of the signature service.

Note: A complete ASN.1 definition is available in [GP CSP ASN1].

6.2.5.1 CSPSignatureSupport

This data structure represents features, types, and algorithms of the *Signature Module*.

It is included in the CSPEnforce command to detect their support by the platform.

ASN 6-5: Signature: ASN.1 Definition for CSPSignatureSupport

```

-- ASN1START
-- Checks for signature functionality support and supported algorithms.
CSPSignatureSupport ::= SEQUENCE {

    -- Message digest algorithms.
    messageDigestAlgorithms    SET OF CSPMessageDigestAlgorithm OPTIONAL,

    -- Signature algorithms.
    signatureAlgorithms        SET OF CSPSignatureAlgorithm OPTIONAL
}
-- ASN1STOP
  
```

6.2.5.2 CSPMessageDigestAlgorithm

This data structure defines the list of available *Message Digest Algorithms*.

A message digest algorithm can be selected using the structures CSPSignatureAlgorithms or CSPKeyDerivationAlgorithms.

The message digest algorithms supported can be detected using the CSPSignatureSupport structure.

ASN 6-6: Signature: ASN.1 Definition for CSPMessageDigestAlgorithm

```
-- ASN1START
-- List of available message digest algorithms.
CSPMessageDigestAlgorithm ::= ENUMERATED {

    -- ALG_NULL:
    -- No message digest is applied to the data.
    alg-null            (0),

    -- ALG_SHA_256:
    -- SHA-256 [FIPS 81] with block size of 64 and hash size of 32 bytes.
    alg-sha-256         (4),

    -- ALG_SHA_384:
    -- SHA-384 [FIPS 81] with block size of 128 and hash value of 64 bytes.
    alg-sha-384         (5),

    -- ALG_SHA_512:
    -- SHA-512 [FIPS 81] with block size of 128 and hash size of 64 bytes.
    alg-sha-512         (6),

    -- ALG_SHA3_256:
    -- SHA3-256 [FIPS 180-4] with block size of 64 and hash size of 32 bytes.
    alg-sha3-256        (9),

    -- ALG_SHA3_384:
    -- SHA3-384 [FIPS 180-4] with block size of 128 and hash size of 64 bytes.
    alg-sha3-384        (10),

    -- ALG_SHA3_512:
    -- SHA3-512 [FIPS 180-4] with block size of 128 and hash size of 64 bytes.
    alg-sha3-512        (11)
}
-- ASN1STOP
```

6.2.5.3 CSPSignatureAlgorithm

This data structure defines the list of available *Signature Algorithms*.

A signature algorithm can be selected using the CSPSignatureAlgorithms structure.

The signature algorithms supported can be detected using the CSPSignatureSupport structure.

ASN 6-7: Signature: ASN.1 Definition for CSPSignatureAlgorithm

```
-- ASN1START
-- List of available signature algorithms.
CSPSignatureAlgorithm ::= ENUMERATED {
```

```

-- SIG_AES_CMAC128:
-- Signature according to [ISO 9797-1]: AES 128-bit block and 16-byte CMAC.
sig-aes-cmac128      (10),

-- SIG_AES_MAC128:
-- Signature according to [SP800-38B]: AES 128-bit block and 16-byte MAC.
sig-aes-mac128      (6),

-- SIG_HMAC:
-- Signature using HMAC according to [FIPS 198-1].
sig-hmac            (7),

-- SIG_RSA:
-- RSA signature according to [IEEE 1363-2000] with PKCS#1-PSS.
sig-rsa             (3),

-- SIG_ECDSA:
-- Signature according to [X9.62] encoded as sequence {r INTEGER, s INTEGER}.
sig-ecdsa           (5),

-- SIG_ECDSA_PLAIN:
-- Signature according to [TR-03111] as octet string r|s in plain format.
sig-ecdsa-plain     (9),

-- SIG_EC_SCHNORR:
-- Schnorr signature according to [TR-03111].
sig-ec-schnorr      (11),

-- SIG_POA_FIAT_SHAMIR:
-- Proof of association (PoA) for two key pairs according to [Verheul-PoA].
sig-poa-fiat-shamir (12)
}
-- ASN1STOP

```

6.2.5.4 CSPSignatureAlgorithms

This data structure represents a container for a resource-specific *Signature Configuration*.

It can be configured for a resource using the CSPAlgorithms structure.

For further information, see section 3.6, Cipher and Signatures, and section 3.6.7, Signatures with Counters & Timestamps.

ASN 6-8: Signature: ASN.1 Definition for CSPSignatureAlgorithms

```

-- ASN1START
-- Signature configuration for a specific resource.
CSPSignatureAlgorithms ::= SEQUENCE {

```

```
-- The signature algorithm.  
signatureAlgorithm      CSPSignatureAlgorithm,  
  
-- The padding algorithm.  
paddingAlgorithm        CSPPaddingAlgorithm DEFAULT pad-null,  
  
-- The message digest algorithm.  
messageDigestAlgorithm  CSPMessageDigestAlgorithm DEFAULT alg-null  
}  
-- ASN1STOP
```

6.3 Transform Module

The CSP may implement the TransformModule to offer *Encryption Transformations*, enabling data encryption to be transferred from one key or algorithm to another; the choice shall be subject to *Modularity*.

The CSP Admin shall be able to create resources with USAGE_TRANSFORM to configure the decryption and encryption key resources using the CSPCipherAlgorithms structure, selecting for each key:

- The paddingAlgorithm from available *Padding Algorithms*.
- The cipherAlgorithm from available *Cipher Algorithms*.

The TransformModule shall support the *Cipher, Padding, and Key Size Combinations* listed in Table 6-4. Client Applications shall adhere to requirements for *Cipher Initialization Data* listed in Table 6-5.

The Client Application shall be able to pass the resourceIds for both the decryption and re-encryption resource to the transform.init operation and trigger:

- Encryption transformation processes using the transform.doFinal operation.
- Multi-part encryption transformation processes using the transform.update operation.

The CSP Admin may detect whether the platform supports this module using the CSPEnforce command.

For further information, see section 3.6, Cipher and Signatures, and section 3.6.8, Encryption Transformations.

6.3.1 Transform Definitions

6.3.1.1 Transform Error Codes

If ERROR_MODE_DETAILED is supported and activated, the TransformModule shall use the error codes listed in Table 6-24 as reasons for exceptions.

For further information, see section 5.1.1.6, Error Types.

Table 6-24: Transform Error Reason Codes

Reason	Description
0x2030	Illegal resource type: Only key resources can be used for encryption transformation.
0x4030	Transform not initialized: The service has not been successfully initialized through the init method.
0x5030	Wrong usage: The decryption key is not configured for USAGE_TRANSFORM.
0x8030	Unsupported: The transform module is not supported.

6.3.2 Transform Configuration

The TransformModule does not have a dedicated configuration. It reuses the paddingAlgorithm and cipherAlgorithm parameters from the *Cipher Module*.

6.3.3 Transform Operations

The TransformModule shall implement the operations listed in Table 6-25.

For each operation, the CSP shall handle supported *Counter Types*, *Timer Types*, and *Audit Events*.

The operations shall solely be utilizable with resources configured for USAGE_TRANSFORM or USAGE_CONFIDENTIAL and shall comply with the *Access Restrictions on Transform Operations* specified in Table 6-26.

Note: Specifically, resources with `USAGE_TRANSFORM`, shall not be permitted for decryption using *Cipher Operations*, *Confidential Data Transfer Operations*, and *Offloading Operations*.

For further information, see section 3.6, Cipher and Signatures, and section 3.6.8, Encryption Transformations.

Table 6-25: Transform Operations

Operation	Details
<code>transform.init</code>	<p>Initializes this service using a key resource for decryption and a second key resource for encryption, both configured with a specific cipher configuration selected from available <i>Cipher Algorithms</i> and <i>Padding Algorithms</i>.</p> <p>CSP API:</p> <ul style="list-style-type: none"> <code>org.globalplatform.csp.api.TransformService.init(..)</code> <p>Evaluate Timers:</p> <ul style="list-style-type: none"> <code>TIMER_VALIDITY_DATE</code>, <code>TIMER_VALIDITY_PERIOD</code> <p>Fire Events:</p> <ul style="list-style-type: none"> <code>EVENT_TIMER_EXPIRED</code>, <code>EVENT_CSP_ERROR</code>
<code>transform.doFinal</code>	<p>Decrypts the provided data using the decryption key, then encrypts it using the encryption key, and returns the re-encrypted data.</p> <p>CSP API:</p> <ul style="list-style-type: none"> <code>org.globalplatform.csp.api.TransformService.doFinal(..)</code> <p>Increment Counters:</p> <ul style="list-style-type: none"> <code>COUNT_USAGE</code>, <code>COUNT_USAGE_PER_BLOCK</code> <code>COUNT_USAGE_SUCCESS_ONLY</code>, <code>COUNT_USAGE_FAILURE_ONLY</code> <p>Fire Events:</p> <ul style="list-style-type: none"> <code>EVENT_COUNTER_EXHAUSTED</code>, <code>EVENT_CSP_ERROR</code>
<code>transform.update</code>	<p>Optional method to handle multi-part encryption transformations by processing a chunk of data.</p> <p>CSP API:</p> <ul style="list-style-type: none"> <code>org.globalplatform.csp.api.TransformService.update(..)</code> <p>Increment Counters:</p> <ul style="list-style-type: none"> <code>COUNT_USAGE_PER_BLOCK</code> <p>Fire Events:</p> <ul style="list-style-type: none"> <code>EVENT_COUNTER_EXHAUSTED</code>, <code>EVENT_CSP_ERROR</code>

6.3.3.1 Access Restrictions on Transform Operations

The `TransformModule` shall enforce the access restrictions for *Transform Operations* listed in Table 6-26 for all *Access Rights*, *Usage Types*, *Resource States*, and *Policy Types* supported by the platform. Unauthorized or illegal operation invocations shall be rejected and handled according to the `errorMode` configured.

Table 6-26: Transform Operations Access Restrictions

Operation	Right Required	Usage Required	State Required	Other Restrictions
<code>transform.init</code> (decryption key)	USE	TRANSFORM CONFIDENTIAL	OPERATIONAL	<ul style="list-style-type: none"> <i>Client Authentication</i> required Only consistent <i>Cipher</i>, <i>Padding</i>, and <i>Key Size</i> Combinations. <code>POLICY_SECCHANNEL_ESTABLISHED</code> <code>POLICY_PASSWORD</code> <code>POLICY_TA2_ACCESS_FLAG</code> <code>POLICY_ASSOCIATION</code>

Operation	Right Required	Usage Required	State Required	Other Restrictions
transform.init (encryption key)	USE	TRANSFORM CONFIDENTIAL	OPERATIONAL	<ul style="list-style-type: none"> • <i>Client Authentication</i> required • Only consistent <i>Cipher, Padding, and Key Size</i> Combinations.
transform.update	-	-	OPERATIONAL	
transform.doFinal	-	-	OPERATIONAL	

6.3.3.2 Sensitive Results computed by Transform Operations

The TransformModule shall temporarily store the results of methods listed in Table 6-27 in transient memory and shall implement the assertSensitiveResult operation, enabling Client Applications to verify that these results were computed without any abnormality.

Table 6-27: Transform Operations Requiring Sensitive Results Checks

Operation	Result Type	Result Description
transform.update	short	The length of the resulting data.
transform.doFinal	short	The length of the resulting data.

6.3.3.3 Sensitive Arrays required for Transform Operations

The TransformModule shall invoke `javacard.framework.SensitiveArrays.assertIntegrity(..)` ([JCAPI]) on the parameters listed in Table 6-28.

For further information, see section 5.1.3.4: Sensitive Arrays, and section 8.3, Byte Buffer Parameters.

Table 6-28: Transform Operations Requiring Sensitive Arrays

Operation	Parameter	Parameter Description
transform.init	Input buffer	Algorithm-specific initialization data.

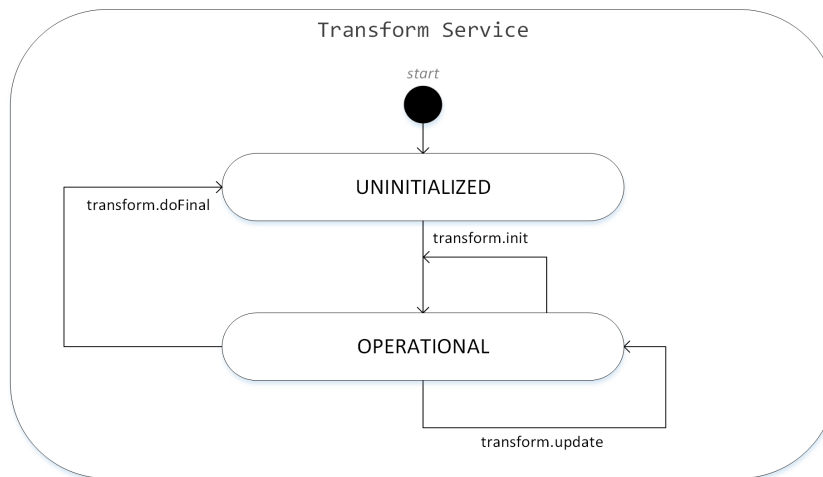
6.3.4 Transform Lifecycle

Figure 6-3 shows lifecycle changes of a transform service triggered by *Transform Operations*. A transform service is an instance of the TransformModule. When a new instance is created, it starts in the UNINITIALIZED state. In this state, it is not possible to perform any transform operations. The Client Application needs to invoke the transform.init to set the transform service to OPERATIONAL mode before any transformation process can be triggered. After completing the transform.doFinal operation, the transform service will return to the UNINITIALIZED state.

The transform service stores *Sensitive Results computed by Transform Operations*.

The transform service triggers lifecycle changes to the key resources used, as specified in section 6.7.4, *Key Lifecycle*.

Figure 6-3: Transform Lifecycle



6.4 Secure Channel Module

The CSP may implement the SecureChannelModule to offer *Secure Messaging* and authentication, with the CSP fully controlling the entire protocol flow; the choice shall be subject to *Modularity*.

The Client Application may select the protocolType from available *Protocol Types* when retrieving an instance of the service using the `org.globalplatform.csp.api.CSP.makeSecureChannelService` operation ([GP CSP API]).

The CSP Admin shall be able to create resources with `USAGE_SECCHANNEL` and configure the security using the `CSPSecureChannelAlgorithms` structure to select:

- The securityFunction from available *Security Functions* of the protocol (e.g., `SEC_CA2`).

The Client Application shall be able to pass the resourceIds referring to this security configuration to the `sc.init` operation and trigger:

- The establishment of the secure channel using the `sc.processSecurity` operation.
- Remove session encryption from incoming APDU messages using the `sc.unwrap` operation.
- Add session encryption to outgoing APDU messages using the `sc.wrap` operation.
- Multi-part secure messaging processes using the `sc.updateUnwrap` and `sc.updateWrap` operations.

The CSP Admin may detect whether the platform supports this module using the `CSPEnforce` command.

For further information, see section 3.7, *Secure Messaging*, and section 3.7.4, *Confidential Data Transfer*.

6.4.1 Secure Channel Definitions

6.4.1.1 Protocol Types

The SecureChannelModule may support the secure channel protocol types listed in Table 6-29; the exact list shall be subject to *Modularity*. The protocol type defines the APDU command sequence for secure channel authentication and messaging. The CSP shall ensure that these APDU commands are strictly invoked in the sequence defined by the external specification of the protocol.

The Client Application may select the protocolType using the `csp.makeSecureChannelService` operation.

The CSP Admin may detect if a protocol type is supported using the `CSPSecureChannelSupport` structure. Each protocol type supported shall adhere to the algorithm combinations specified in

- Table 6-31 for compatible *Protocol, Resource, Size, and Curve* Combinations

For further information, see section 3.7.2, *Built-in Secure Channel Protocols*.

Table 6-29: Protocol Types

Protocol Type	Value (Hex)	Value (Int)	Protocol Description
PROTOCOL_PACE ¹⁶	0x11	17	<p><i>PACE</i> according to [TR-03110-1] section 3.3, using APDU commands and secure messaging specified in [TR-03110-3] Annex B.1 and Annex F.</p> <p>The session encryption uses AES in CBC mode (as specified for CIPHER_AES_CBC), and message authentication uses AES-CMAC (as specified for SIG_AES_CMAC128). Both use 128-bit keys derived from PACE key agreement.</p> <p>Input Resources:</p> <ul style="list-style-type: none"> • Password with SEC_PACE_PIN <p>Other configurations:</p> <ul style="list-style-type: none"> • tryLimit for PIN • POLICY_PRE_BLOCKED
PROTOCOL_EAC_ID	0x12	18	<p><i>EAC for eID</i> according to [TR-03110-3], integrating:</p> <ul style="list-style-type: none"> • PACE as specified in [TR-03110-1] section 3.3 • TA2 as specified in [TR-03110-2] section 3.3 • CA2 as specified in [TR-03110-2] section 3.4 • CA3 as specified in [TR-03110-2] section 3.5 <p>Input Resources:</p> <ul style="list-style-type: none"> • Password with SEC_PACE_PIN • Password with SEC_PACE_PUK • Password with SEC_PACE_CAN • Certificate with SEC_TA_AT_ROOT • Key with SEC_CA2 • Key with SEC_CA2_PRIVILEGED • Key with SEC_CA3 • Key with SEC_CA3_PSA <p>Output Resources:</p> <ul style="list-style-type: none"> • Certificate with SEC_TA_DV • Certificate with SEC_TA_TERMINAL • Certificate with SEC_TA_AT_LINKED for <i>Certificate Rollover</i> <p>Other configurations:</p> <ul style="list-style-type: none"> • tryLimit for PIN, PUK, and CAN • POLICY_PRE_BLOCKED • POLICY_TA2_ACCESS_FLAG • TIME_SYNC_FROM_TA

¹⁶ PROTOCOL_PACE is mandatory, as it is required for *Client Authentication* (see section 5.3.1.1).

Protocol Type	Value (Hex)	Value (Int)	Protocol Description
PROTOCOL_EAC_MRTD	0x13	19	<p>EAC for MRTD according to [ICAO 9303-11], integrating:</p> <ul style="list-style-type: none"> • PACE as specified in [ICAO 9303-11] section 4.4 • CA1 as specified in [ICAO 9303-11] section 6.2 • TA1 as specified in [ICAO 9303-11] section 7.1 <p>Input Resources:</p> <ul style="list-style-type: none"> • Password with SEC_PACE_CAN • Password with SEC_PACE_MRZ • Key with SEC_CA1 • Certificate with SEC_TA_IS_ROOT <p>Output Resources:</p> <ul style="list-style-type: none"> • Certificate with SEC_TA_DV • Certificate with SEC_TA_TERMINAL • Certificate with SEC_TA_IS_LINKED for <i>Certificate Rollover</i> <p>Other configurations:</p> <ul style="list-style-type: none"> • tryLimit for CAN and MRZ • POLICY_PRE_BLOCKED • TIME_SYNC_FROM_TA
PROTOCOL_PACE_CAM	0x14	20	<p>PACE-CAM according to [ICAO 9303-11], including:</p> <ul style="list-style-type: none"> • CAM according to [ICAO 9303-11] section 4.4 • TA1 as specified in [ICAO 9303-11] section 7.1 <p>Input Resources:</p> <ul style="list-style-type: none"> • Password with SEC_PACE_CAN • Password with SEC_PACE_MRZ • Key with SEC_CA1 • Certificate with SEC_TA_IS_ROOT <p>Output Resources:</p> <ul style="list-style-type: none"> • Certificate with SEC_TA_DV • Certificate with SEC_TA_TERMINAL • Certificate with SEC_TA_IS_LINKED for <i>Certificate Rollover</i> <p>Other configurations:</p> <ul style="list-style-type: none"> • tryLimit for CAN and MRZ • POLICY_PRE_BLOCKED • TIME_SYNC_FROM_TA
PROTOCOL_SCP03	0x05	5	<p>SCP03 according to [GP Amd D].</p> <p>Note: The CSP shall support the use of custom CSP Resources as well as the <i>ENC</i> and <i>Key-MAC</i> of the CSP SD.</p> <p>Input Resources:</p> <ul style="list-style-type: none"> • Key with SEC_KENC • Key with SEC_KMAC
PROTOCOL_SCP04	0x15	21	<p>SCP04 according to [GP Amd K].</p> <p>Note: The CSP shall support the use of custom CSP Resources as well as the <i>ENC</i> and <i>Key-MAC</i> of the CSP SD.</p> <p>Input Resources:</p> <ul style="list-style-type: none"> • Key with SEC_KENC • Key with SEC_KMAC

6.4.1.2 Security Functions

The SecureChannelModule may support the security functions listed in Table 6-30; the exact list shall be subject to *Modularity*.

The CSP Admin may configure the securityFunction using the CSPSecurityFunction structure. The Client Application may pass the resourceIds, which refer to the security resources, to the SecureChannelModule through the sc.init operation.

The CSP shall enforce access restrictions on security computations, as specified in:

- Table 6-42 for *Access Restrictions on Secure Channel Operations*

The CSP Admin may detect if a security function is supported using the CSPSecureChannelSupport structure. Each security function supported shall adhere to the algorithm combinations specified in

- Table 6-31 for compatible *Protocol, Resource, Size, and Curve Combinations*

Table 6-30: Security Functions

Security Function	Value (Hex)	Value (Int)	Description
SEC_PACE_PIN ¹⁷	0x01	1	<p>PACE password initialized with a user PIN, as specified in [TR-03110-2].</p> <p>It is used for <i>PACE</i> and <i>EAC for eID</i>.</p> <p>The CSP shall manage a <i>Retry Counter with Maximum Try Limit</i> for it.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_PACE • PROTOCOL_EAC_ID
SEC_PACE_PUK	0x02	2	<p>PACE password initialized with a Personal Unblocking Key (PUK) used to unblock and reset the PACE PIN after incorrect attempts, as specified in [TR-03110-2].</p> <p>It is used in <i>EAC for eID</i>.</p> <p>Note: If not supported, the CSP will not evaluate MSE:SET AT.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_EAC_ID
SEC_PACE_CAN	0x03	3	<p>PACE password initialized with the Card Access Number (CAN), a number printed on the document, as specified in [ICAO 9303-11].</p> <p>It is used for <i>EAC for eID</i>, <i>EAC for MRTD</i> and <i>PACE-CAM</i>.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_EAC_ID • PROTOCOL_EAC_MRTD • PROTOCOL_PACE_CAM
SEC_PACE_MRZ	0x04	4	<p>PACE password initialized with the Machine Readable Zone (MRZ), which consists of the passport number and date of birth, as specified in [ICAO 9303-11].</p> <p>It is used for <i>EAC for MRTD</i> and <i>PACE-CAM</i>.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_EAC_MRTD • PROTOCOL_PACE_CAM

¹⁷ SEC_PACE_PIN is mandatory, as it is required for *Client Authentication* (see section 5.3.1.1).

Security Function	Value (Hex)	Value (Int)	Description
SEC_TA_AT_ROOT	0x05	5	<p>Certificate initialized with the Authentication Terminal Root Certificate (AT-Root), used in <i>EAC for eID</i>. The AT-Root certificate serves as the root of the PKI for Authentication Terminals. The CSP shall use it to verify the certificate chain of Terminal Authentication (TA) certificates received from terminals during EAC protocol execution, as specified in [TR-03110-2].</p> <p>It is used in <i>EAC for eID</i>.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_EAC_ID
SEC_TA_IS_ROOT	0x06	6	<p>Certificate initialized with the Inspection System Root Certificate (IS-Root), used in <i>EAC for MRTD</i> and <i>PACE-CAM</i>. The AT-Root certificate serves as the root of the PKI for Inspection Systems. The CSP shall use it to verify the certificate chain of Terminal Authentication (TA) certificates received from terminals during EAC protocol execution, as specified in [ICAO 9303-11].</p> <p>It is used for <i>EAC for MRTD</i> and <i>PACE-CAM</i>.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_EAC_MRTD • PROTOCOL_PACE_CAM
SEC_CA1	0x07	7	<p>Chip Authentication v1 (CA1) key to verify the chip [ICAO 9303-11].</p> <p>It is used for <i>EAC for MRTD</i> and <i>PACE-CAM</i>.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_EAC_MRTD • PROTOCOL_PACE_CAM
SEC_CA2	0x08	8	<p>Chip Authentication v2 (CA2) to verify the chip [TR-03110-2].</p> <p>It is used in <i>EAC for eID</i>.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_EAC_ID
SEC_CA2_PRIVILEGED	0x09	9	<p>While the CA2 key is typically a group key, the CA2-privileged key is a chip-specific CA2 key used for Privileged Terminals ([TR-03110-3]).</p> <p>It is used in <i>EAC for eID</i>.</p> <p>Note: If not supported, SEC_CA2 is used instead.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_EAC_ID
SEC_CA3	0x0A	10	<p>Chip Authentication v3 (CA3) key to verify the chip in [TR-03110-2].</p> <p>It is used in <i>EAC for eID</i>.</p> <p>Note: If not supported, CA3 is not available in PROTOCOL_EAC_ID.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_EAC_ID
SEC_RI			

Security Function	Value (Hex)	Value (Int)	Description
SEC_CA3_PSA	0x0B	11	<p>While the CA3 key is typically a group key, the CA3-PSA key is a chip-specific CA3 key used for Pseudonymous Secure Authentication (PSA) ([TR-03110-3]).</p> <p>It is used in <i>EAC for eID</i>.</p> <p>Note: If not supported, SEC_CA3 is used instead.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_EAC_ID
SEC_TA_DV	0x0C	12	<p>A certificate received during EAC authentication from the Document Verify (DV) to authorize subordinate terminals ([TR-03110-3]). The CSP parses the incoming APDU and stores the DV certificate content to this output resource.</p> <p>It is used in <i>EAC for eID</i>, <i>EAC for MRTD</i>, and <i>PACE-CAM</i> and for POLICY_TA2_ACCESS_FLAG.</p> <p>Note: If not supported, Client Applications must parse the APDU manually.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_EAC_ID • PROTOCOL_EAC_MRTD • PROTOCOL_PACE_CAM
SEC_TA_TERMINAL	0x0D	13	<p>A certificate received during EAC authentication from individual terminals ([TR-03110-3]). The CSP parses the incoming APDU and stores the terminal certificate content in this output resource.</p> <p>It is used in <i>EAC for eID</i>, <i>EAC for MRTD</i>, and <i>PACE-CAM</i> and for POLICY_TA2_ACCESS_FLAG.</p> <p>Note: If not supported, Client Applications must parse the APDU manually.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_EAC_ID • PROTOCOL_EAC_MRTD • PROTOCOL_PACE_CAM
SEC_TA_AT_LINKED	0x0E	14	<p>Linked certificates are used for <i>Certificate Rollover</i> as specified in [TR-03110-3] section 2.4. The CSP stores the linked certificate received via the APDU in this output resource and updates the AT-Root certificate.</p> <p>It is used in <i>EAC for eID</i>.</p> <p>Note: If not supported, the CSP does not handle certificate rollover.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_EAC_ID
SEC_TA_IS_LINKED	0x0F	15	<p>Linked certificates are used for <i>Certificate Rollover</i> as specified in [TR-03110-3] section 2.4. The CSP stores the linked certificate received via the APDU in this output resource and updates the IS-Root certificate.</p> <p>It is used in <i>EAC for MRTD</i> and <i>PACE-CAM</i>.</p> <p>Note: If not supported, the CSP does not handle certificate rollover.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_EAC_MRTD

Security Function	Value (Hex)	Value (Int)	Description
SEC_KENC	0x20	32	<p>Key-ENC (KENC) is a static symmetric key used to create shared session keys (S-ENC), after mutual authentication between the CSP and an off-card entity, as specified in [GP Amd D].</p> <p>It is used in <i>SCP03</i> and <i>SCP04</i>.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_SCP03 • PROTOCOL_SCP04
SEC_KMAC	0x21	33	<p>KEY-MAC (KMAC) is a static symmetric key used to create shared session MACs (S-MAC), after mutual authentication between the CSP and an off-card entity, as specified in [GP Amd D].</p> <p>It is used in <i>SCP03</i> and <i>SCP04</i>.</p> <p>Protocol types:</p> <ul style="list-style-type: none"> • PROTOCOL_SCP03 • PROTOCOL_SCP04

6.4.1.3 Protocol, Resource, Size, and Curve Combinations

The SecureChannelModule shall support the compatible combinations of *Security Functions* listed in Table 6-31 for all *Resource Types*, *Key Types*, *Certificate Types*, *Password Types*, *Key Sizes*, and *ECC Curves* supported by the platform. Unsupported or illegal combinations shall be rejected and handled according to the errorMode configured.

For further information, see section 3.7.2, Built-in Secure Channel Protocols.

Table 6-31: Protocol, Resource, Size, and Curve Combinations

Protocol Type	Security Usage	Resource Type	Resource Size or Curve
PROTOCOL_PACE	SEC_PACE_PIN	PWD_NUMERIC	4-6 characters
		PWD_ALPHANUMERIC	4-n characters
PROTOCOL_EAC_ID	SEC_PACE_PIN SEC_PACE_PUK SEC_PACE_CAN	PWD_NUMERIC	6 characters
	SEC_TA_AT_ROOT	CERT_CVC (ECC)	CURVE_SEC_P256_R1 CURVE_SEC_P384_R1 CURVE_BRAINPOOL_P384_R1
	SEC_CA2	KEY_ECC_PRIVATE	CURVE_SEC_P256_R1 CURVE_SEC_P384_R1 CURVE_BRAINPOOL_P384_R1
PROTOCOL_EAC_MRTD	SEC_PACE_CAN	PWD_NUMERIC	6 characters
	SEC_PACE_MRZ	PWD_ALPHANUMERIC	50-n characters
	SEC_TA_IS_ROOT	CERT_CVC (ECC)	CURVE_SEC_P256_R1 CURVE_SEC_P384_R1 CURVE_BRAINPOOL_P384_R1
		CERT_CVC (RSA)	2048 bit, 3072 bit

Protocol Type	Security Usage	Resource Type	Resource Size or Curve
	SEC_CA1	KEY_ECC_PRIVATE	CURVE_SEC_P256_R1 CURVE_SEC_P384_R1 CURVE_BRAINPOOL_P384_R1
		KEY_RSA_PRIVATE	2048 bit, 3072 bit
PROTOCOL_PACE_CAM	SEC_PACE_CAN	PWD_NUMERIC	6 characters
	SEC_PACE_MRZ	PWD_ALPHANUMERIC	50-n characters
	SEC_CA1	KEY_ECC_PRIVATE	CURVE_SEC_P256_R1 CURVE_SEC_P384_R1 CURVE_BRAINPOOL_P384_R1
	SEC_TA_IS_ROOT	CERT_CVC (ECC)	CURVE_SEC_P256_R1 CURVE_SEC_P384_R1 CURVE_BRAINPOOL_P384_R1
PROTOCOL_SCP03	SEC_KENC	KEY_AES	128 bit, 256 bit
PROTOCOL_SCP04	SEC_KMAC	KEY_AES	128 bit, 256 bit

6.4.1.4 Security States

The SecureChannelModule shall manage the security states listed in Table 6-29. The security state indicates the current active security of the secure channel.

The securityState is an internal parameter of the secure channel service.

Client Applications may retrieve the security state through the `sc.getSecurityState` operation.

Table 6-32: Security States

Security State	Value (Int)	Value (Hex)	Description
SECURITY_NOT_ESTABLISHED	0	0x00	The secure channel is not authenticated. The <code>sc.wrap</code> and <code>sc.unwrap</code> operations only forward incoming data to the outgoing buffer without adding or removing any session encryption.
SECURITY_IN_PROGRESS	1-126	0x01 - 0x7F	Represents intermediate states of the secure channel establishment process. The specific value within this range is set by the CSP based on the current protocol step. During this phase, <code>sc.wrap</code> and <code>sc.unwrap</code> may partially or fully add or remove session encryption, depending on the protocol's progress.
SECURITY_ESTABLISHED	127	0x7F	Indicates that the secure channel is fully established. The <code>sc.wrap</code> and <code>sc.unwrap</code> operations fully apply the security measures according to the protocol's specifications (e.g., encryption, MACs). Note: Used for <code>POLICY_SECCHANNEL_ESTABLISHED</code> and <code>TIMER_SECURITY_TIMEOUT</code> .

6.4.1.4.1 PACE States

Table 6-33: Security States PACE

Security State	Value (Hex)	Value (Int)	Description
SECURITY_NOT_ESTABLISHED	0x00	0	No secure channel established.
PACE_PASSWORD_DERIVED	0x01	1	Derivation of the mapping key and exchange of nonces between OCE and CSP.
PACE_MAPPING_KEY_COMPUTED	0x02	2	Computation of the shared mapping key K.
PACE_KEY_AGREEMENT_INITIATION	0x03	3	Exchange of mapped public keys between OCE and CSP.
PACE_SHARED_SECRET_COMPUTED	0x04	4	Both parties compute the shared secret Z.
PACE_MUTUAL_AUTHENTICATION	0x05	5	Exchange and verification of cryptographic checksums (MACs).
SECURITY_ESTABLISHED	0x7F	127	Secure channel fully established.

6.4.1.4.2 EAC eID States

Table 6-34: Security States EAC eID

Security State	Value (Hex)	Value (Int)	Description
SECURITY_NOT_ESTABLISHED	0x00	0	No secure channel established.
Defined in Table 6-33	0x01 - 0x05	1-5	Execution of PACE protocol steps.
EAC_ID_TA2_FINISHED	0x10	16	Terminal Authentication 2 completed.
EAC_ID_CA2_FINISHED	0x20	32	Chip Authentication 2 completed.
SECURITY_ESTABLISHED	0x7F	127	Secure channel fully established.

6.4.1.4.3 EAC Passport States

Table 6-35: Security States EAC v1

Security State	Value (Hex)	Value (Int)	Description
SECURITY_NOT_ESTABLISHED	0x00	0	No secure channel established.
Defined in Table 6-33	0x01 - 0x05	1-5	Execution of PACE protocol steps.
EAC_PASS_CA1_FINISHED	0x10	16	Chip Authentication 1 completed.
EAC_PASS_TA1_FINISHED	0x20	32	Terminal Authentication 1 completed.
SECURITY_ESTABLISHED	0x7F	127	Secure channel fully established.

6.4.1.4.4 PACE CAM States

Table 6-36: Security States PACE CAM

Security State	Value (Hex)	Value (Int)	Description
SECURITY_NOT_ESTABLISHED	0x00	0	No secure channel established.
Defined in Table 6-33	0x01 - 0x05	1-5	Execution of PACE protocol steps.
PACE_CAM_FINISHED	0x10	16	Chip Authentication Mapping procedure completed.
PACE_CAM_CA2_FINISHED	0x20	32	Chip Authentication 2 completed.
PACE_CAM_TA2_FINISHED	0x30	64	Terminal Authentication 2 completed.
SECURITY_ESTABLISHED	0x7F	127	Secure channel fully established; secure messaging active.

6.4.1.4.5 SCP03 & SCP04 States

Table 6-37: Security States SCP03 & SCP04

Security State	Value (Hex)	Value (Int)	Description
SECURITY_NOT_ESTABLISHED	0x00	0	No secure channel established.
SCP_KEY_AGREEMENT	0x01	1	Initialization of protocol; exchange of initial data and selection of security level.
SCP_AUTHENTICATE	0x02	2	Host sends EXTERNAL AUTHENTICATE command with cryptographic signature (MAC).
SCP_VERIFIED	0x03	3	CSP verifies the authentication data; mutual authentication achieved.
SECURITY_ESTABLISHED	0x7F	127	Secure channel fully established.

6.4.1.5 Secure Channel Events

The AuditModule may log the secure-channel-related *Audit Events* listed in Table 6-38; the exact list shall be subject to *Modularity*. The event-specific data listed in the table shall be stored according to the configured auditMode.

The CSP Admin may configure the resourceEvents to be logged using CSPResourceEvent.

The CSP Admin may detect if an event type is supported using the CSPAuditSupport structure.

For further information, see section 3.11.4, Selecting Audit Events, and section 6.12.1.2, Audit Events.

Table 6-38: Secure Channel Events for Auditing

Event Type	Value (Hex)	Value (Int)	Description	Event Category
EVENT_SECCHANNEL_ESTABLISHED	0x1040	4160	Secure messaging successfully established. Logged after successful invocations a secure messaging channel is fully authenticated. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Operations: sc.processSecurity Event Data: CSPEventDataResource 	Resource Event
EVENT_SECCHANNEL_AUTHENTICATION_FAILED	0x1041	4161	Authentication for secure messaging failed. Logged after a failed authentication process for a certain secure channel protocol. Note: The event is not logged if an exception occurs. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Operations: sc.processSecurity Event Data: CSPEventDataResource 	Resource Event

6.4.1.6 Secure Channel Error Codes

If ERROR_MODE_DETAILED is supported and activated, the SecureChannelModule shall use the error codes listed in Table 6-39 as reasons for exceptions.

For further information, see section 5.1.1.6, Error Types.

Table 6-39: Secure Channel Error Reason Codes

Reason	Description
0x2040	Unknown protocol type: The protocol type mode provided is not defined.
0x2041	Illegal resource type: The resource type provided is not accepted by the protocol.
0x3040	Inconsistent config: Key type, size, curve, or algorithm incompatible with the protocol.
0x4040	Security not initialized: sc.init not invoked.
0x4041	Wrapping not initialized: sc.initWrap not invoked.
0x4042	Unwrapping not initialized: sc.initUnwrap not invoked.
0x4043	Wrapping mode active: Finalize with sc.wrap before switching mode.
0x4044	Unwrapping mode active: Finalize with sc.unwrap before switching mode.
0x5040	Wrong usage: The resource used is not configured for USAGE_SECCHANNEL.
0x6040	Illegal APDU: The provided APDU is incompatible with the protocol type or state.
0x8040	Unsupported: The secure channel module is not supported.
0x8041	Unsupported: The protocol type is not supported.

6.4.2 Secure Channel Configuration

The SecureChannelModule shall support the configuration parameters defined in Table 6-40.

Table 6-40: Secure Channel Configuration Parameters

Parameter	Description	Parameter Type
protocolType	The protocol type, selected from available <i>Protocol Types</i> , is set when retrieving an instance of <code>org.globalplatform.csp.api.SecureChannelService</code> ([GP CSP API]). Technical Details: <ul style="list-style-type: none"> Data type: <code>CSPProtocolType</code> Operations: <code>CSP.makeSecureChannelService</code> 	Service Parameter
securityFunction	The security function, such as CA or TA, selected from available <i>Security Functions</i> and configured to a resource with <code>USAGE_SECCHANNEL</code> . Technical Details: <ul style="list-style-type: none"> Administration: <code>CSPSecurityFunction</code> 	Resource Parameter
securityTimeout	A general timeout in seconds for secure channel services as specified by <code>TIMER_SECURITY_TIMEOUT</code> . Technical Details: <ul style="list-style-type: none"> Data type: <code>CSPManualTimer</code> Administration: <code>CSPSecureChannelSettings</code> Operations: <code>time.getValue</code> 	CSP Instance Parameter

6.4.3 Secure Channel Operations

The CSP¹⁸ shall implement the operations listed in Table 6-41.

For each operation, the CSP shall handle supported *Counter Types*, *Timer Types* and *Audit Events*.

The operations shall solely be utilizable with resources configured for `USAGE_SECCHANNEL` and shall comply with the *Access Restrictions on Secure Channel Operations* specified in Table 6-42.

Note: Specifically, resources with `USAGE_SECCHANNEL` shall not be permitted for *Cipher Operations*, *Signature Operations*, or *Password Verification*.

For further information, see section 3.7, Secure Messaging, and section 3.7.1, Secure Communication Flow.

Table 6-41: Secure Channel Operations

Operation	Details
sc.init	Initializes this service using key, certificate and password resources, configured with a specific security configuration selected from available <i>Security Functions</i> . CSP API: <ul style="list-style-type: none"> <code>org.globalplatform.csp.api.SecureChannelService.init(..)</code> Fire Events: <ul style="list-style-type: none"> <code>EVENT_CSP_ERROR</code>

¹⁸ *Secure Channel Operations* are mandatory for `PROTOCOL_PACE` with `SEC_PACE_PIN` (see section 2.3, Modularity).

Operation	Details
sc.processSecurity	<p>Process authentication. It processes incoming APDUs and creates response APDUs according to the selected <code>protocolType</code> and the internal state of the secure channel service. This operation handles <i>Certificate Rollover</i> for <code>SEC_TA_AT_LINKED</code> and <code>SEC_TA_IS_LINKED</code>. This operation manages a <i>Retry Counter with Maximum Try Limit</i> for <code>SEC_PACE_PIN</code>. This operation handles <i>TA Certificates as Time Source</i> for <code>TIME_SYNC_FROM_TA</code>.</p> <p>CSP API:</p> <ul style="list-style-type: none"> • <code>org.globalplatform.csp.api.SecureChannelService.processSecurity(..)</code> <p>CSP Protocol:</p> <ul style="list-style-type: none"> • <code>CSPProcessSecurity</code> <p>Evaluate Timers:</p> <ul style="list-style-type: none"> • <code>TIMER_VALIDITY_DATE</code>, <code>TIMER_VALIDITY_PERIOD</code>, <code>TIMER_VALIDITY_CERTIFICATE</code> <p>Start Timers:</p> <ul style="list-style-type: none"> • <code>TIMER_SECURITY_TIMEOUT</code> (if the secure channel is fully established) <p>Increment Counters:</p> <ul style="list-style-type: none"> • <code>COUNT_USAGE</code>, <code>COUNT_USAGE_PER_BLOCK</code> • <code>COUNT_USAGE_SUCCESS_ONLY</code>, <code>COUNT_USAGE_FAILURE_ONLY</code> • <code>COUNT_TRANSPORT_USAGE</code>, <code>tryCounter</code> <p>Fire Events:</p> <ul style="list-style-type: none"> • <code>EVENT_SECCHANNEL_ESTABLISHED</code>, <code>EVENT_SECCHANNEL_AUTHENTICATION_FAILED</code> • <code>EVENT_CSP_TIME_SET</code>, <code>EVENT_CSP_ERROR</code>
sc.resetSecurity	<p>Reset the secure channel session and invalidates all session data.</p> <p>CSP API:</p> <ul style="list-style-type: none"> • <code>org.globalplatform.csp.api.SecureChannelService.resetSecurity(..)</code>
sc.getSecurityState	<p>Returns the current <code>securityState</code> of the secure channel, representing one of the protocol-specific <i>Security States</i>.</p> <ul style="list-style-type: none"> • <code>org.globalplatform.csp.api.SecureChannelService.getSecurityState(..)</code>
sc.initWrap	<p>Prepare service for session encryption and sets it to wrapping mode.</p> <p>CSP API:</p> <ul style="list-style-type: none"> • <code>org.globalplatform.csp.api.SecureChannelService.initWrap(..)</code> <p>Evaluate Timers:</p> <ul style="list-style-type: none"> • <code>TIMER_SECURITY_TIMEOUT</code> <p>Fire Events:</p> <ul style="list-style-type: none"> • <code>EVENT_CSP_ERROR</code>
sc.wrap	<p>Add session encryption to the data, computed according to the external specification defined by the selected <code>protocolType</code>, as shown in Figure 3-2 <i>Secure Communication Flow</i>.</p> <p>CSP API:</p> <ul style="list-style-type: none"> • <code>org.globalplatform.csp.api.SecureChannelService.wrap(..)</code> <p>Fire Events:</p> <ul style="list-style-type: none"> • <code>EVENT_CSP_ERROR</code>
sc.updateWrap	<p>An optional method to handle multi-part session encryption by processing chunks of data.</p> <p>CSP API:</p> <ul style="list-style-type: none"> • <code>org.globalplatform.csp.api.SecureChannelService.updateWrap(..)</code> <p>Fire Events:</p> <ul style="list-style-type: none"> • <code>EVENT_CSP_ERROR</code>

Operation	Details
sc.initUnwrap	Prepare service for session decryption and sets it to unwrapping mode. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.SecureChannelService.initUnwrap(..) Evaluate Timers: <ul style="list-style-type: none"> TIMER_SECURITY_TIMEOUT Fire Events: <ul style="list-style-type: none"> EVENT_CSP_ERROR
sc.unwrap	Remove session encryption from the provided data according to the external specification defined by the selected protocolType, as shown in Figure 3-2 <i>Secure Communication Flow</i> . CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.SecureChannelService.unwrap(..) Fire Events: <ul style="list-style-type: none"> EVENT_CSP_ERROR
sc.updateUnwrap	An optional method to handle multi-part session decryption by processing chunks of data. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.SecureChannelService.updateUnwrap(..) Fire Events: <ul style="list-style-type: none"> EVENT_CSP_ERROR

6.4.3.1 Access Restrictions on Secure Channel Operations

The CSP shall enforce the access restrictions for *Secure Channel Operations* listed in Table 6-42 for all *Access Rights*, *Usage Types*, *Resource States*, and *Policy Types* supported by the platform. Unauthorized or illegal operation invocations shall be rejected and handled according to the errorMode configured.

Table 6-42: Secure Channel Operations Access Restrictions

Operation	Right Required	Usage Required	State Required	Other Restrictions
sc.init	USE	SECCHANNEL	-	<ul style="list-style-type: none"> Only consistent <i>Protocol</i>, <i>Resource</i>, <i>Size</i>, and <i>Curve</i> Combinations
sc.processSecurity	-	-	OPERATIONAL	<ul style="list-style-type: none"> Only possible if required resources are set in sig.init. Only APDUs that meet the internal state of the protocolType used
sc.resetSecurity	-	-	-	<ul style="list-style-type: none"> Note: Possible in any mode and when not authenticated.
sc.initWrap	-	-	-	<ul style="list-style-type: none"> Note: Possible in unwrap mode and when not authenticated.
sc.initUnwrap	-	-	-	<ul style="list-style-type: none"> Not possible in wrapping mode. Note: Possible when not authenticated.
sc.updateWrap sc.wrap	-	-	-	<ul style="list-style-type: none"> Only possible in wrap mode. Note: Returns input data when not authenticated.
sc.updateUnwrap sc.unwrap	-	-	-	<ul style="list-style-type: none"> Only possible in unwrap mode. Note: Returns input data when not authenticated.

6.4.3.2 Sensitive Results Computed by Secure Channel Operations

The CSP shall temporarily store the results of methods listed in Table 6-43 in transient memory and shall implement the `assertSensitiveResult` operation, enabling Client Applications to verify that these results were computed without any abnormality.

Table 6-43: Secure Channel Operations Requiring Sensitive Results Checks

Operation	Result Type	Result Description
<code>sc.processSecurity</code>	short	The length of the resulting response data.
<code>sc.wrap</code>	short	The length of the resulting wrapped data.
<code>sc.unwrap</code>	short	The length of the resulting unwrapped data.
<code>sc.updateWrap</code>	short	The length of the wrapped data.
<code>sc.updateUnwrap</code>	short	The length of the unwrapped data.

6.4.3.3 Sensitive Arrays Required for Secure Channel Operations

The `SecureChannelModule` shall invoke `javacard.framework.SensitiveArrays.assertIntegrity(..)` ([JCAPI]) on the parameters listed in Table 6-44.

For further information, see section 5.1.3.4, Sensitive Arrays, and section 8.3, Byte Buffer Parameters.

Table 6-44: Secure Channel Operations Requiring Sensitive Arrays

Operation	Parameter	Description
<code>sc.processSecurity</code>	APDU input buffer	Input APDU buffer.
<code>sc.processSecurity</code>	APDU output buffer.	Output APDU buffer.
<code>sc.updateWrap</code>	Data input buffer	Data to add session encryption (multi-part).
<code>sc.wrap</code>	Data input buffer	Final data to add session encryption.
<code>sc.updateWrap</code>	Data output buffer	Decrypted data (multi-part).
<code>sc.wrap</code>	Data output buffer	Final decrypted data.

6.4.4 Secure Channel Lifecycle

Figure 6-4 illustrates the lifecycle changes of a secure channel service triggered by *Secure Channel Operations*. A secure channel service is an instance of the `SecureChannelModule`. Client Applications may invoke the `sc.init` operation to provide required protocol resources, `sc.processSecurity` to establish the secure channel, `sc.wrap` to add session encryption to outgoing messages and `sc.unwrap` to remove session encryption from incoming messages. All operations can be invoked repeatedly until the secure channel is fully established. The current `securityState` of the secure channel can be retrieved using the `sc.getSecurityState` operation.

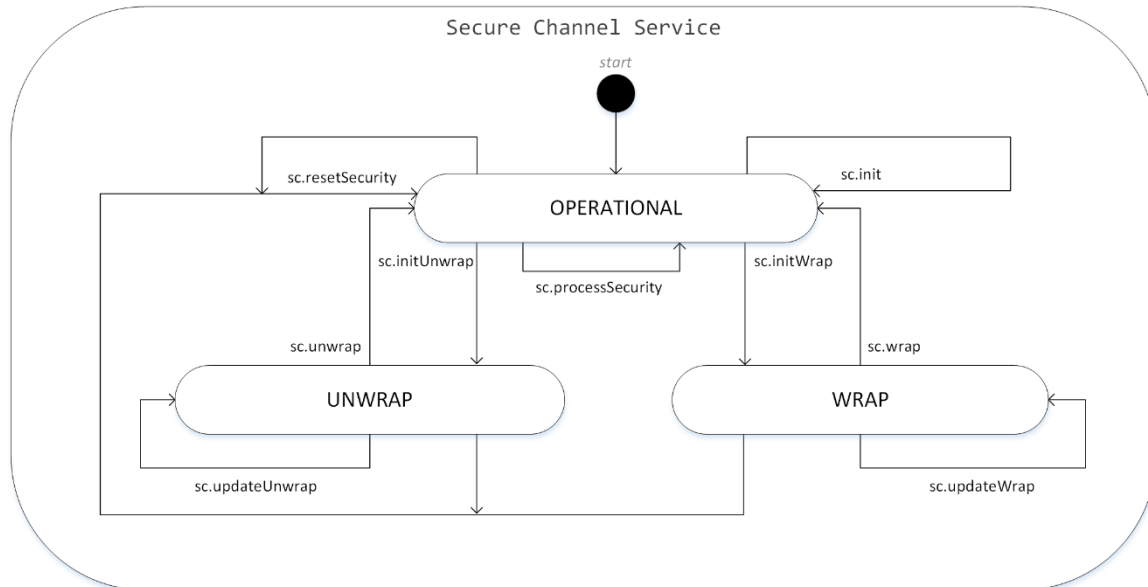
An optional timeout, generally configurable for all secure channel services of a CSP instance, is evaluated each time the wrapping or unwrapping mode changes. If the `securityTimeout` expires, or the Client Application manually invokes `sc.resetSecurity`, the `securityState` of the secure channel service is reset to `SECURITY_NOT_ESTABLISHED`.

The secure channel service stores *Sensitive Arrays Required for Secure Channel Operations*.

The secure channel service triggers lifecycle changes to the resources used, as specified in:

- Section 6.7.4 *Key Lifecycle*
- Section 6.8.4 *Certificate Lifecycle*
- Section 6.9.4 *Password Lifecycle*

Figure 6-4: Secure Channel Lifecycle



6.4.4.1 Internal Secure Channel Parameters

The SecureChannelModule shall maintain the internal parameters listed in Table 6-45.

Table 6-45: Secure Channel-internal Parameters

Parameter	Description	Parameter Type
securityState	<p>Indicates the current security state of the secure channel service. It is set according to the last protocol step processed via <code>sc.processSecurity</code> and is reset when the Client Application invokes either <code>sc.resetSecurity</code> or <code>resource.clear</code>. The <code>securityState</code> is an internal parameter and shall be stored in transient memory with <code>CLEAR_ON_RESET</code>.</p> <p>Depending on the current <code>securityState</code>, the SecureChannelModule shall maintain the authenticated flag of passwords used within the secure channel (e.g., <i>PACE</i> Pin).</p> <p>Note: The <code>securityState</code> might be stored or linked to the resources involved in secure channel authentication for evaluating the <code>POLICY_SECCHANNEL_ESTABLISHED</code>.</p> <p>Events:</p> <ul style="list-style-type: none"> • <code>EVENT_SECCHANNEL_ESTABLISHED</code> • <code>EVENT_SECCHANNEL_AUTHENTICATION_FAILED</code> <p>Operations:</p> <ul style="list-style-type: none"> • <code>sc.getSecurityState</code> 	Service Parameter

6.4.5 Secure Channel Structures

This section lists ASN.1 structures related to the configuration of the secure channel service.

Note: A complete ASN.1 definition is available in [GP CSP ASN1].

6.4.5.1 CSPSecureChannelSupport

This data structure represents features, types, and algorithms of the *Secure Channel Module*.

It is included in the CSPEnforce command to detect their support by the platform.

ASN 6-9: SecChannel: ASN.1 Definition for CSPSecureChannelSupport

```
-- ASN1START
-- Checks for secure channel functionality support and supported protocols.
CSPSecureChannelSupport ::= SEQUENCE {

    -- Secure channel protocol types.
    protocolTypes          CSPProtocolType OPTIONAL,

    -- Secure channel security functions.
    securityFunctions      CSPSecurityFunction OPTIONAL
}
-- ASN1STOP
```

6.4.5.2 CSPSecureChannelSettings

This data structure represents a general *Secure Channel Configuration* for the CSP Instance.

It can be configured using the CSPSetup structure.

It can be retrieved using the CSPConfiguration structure.

For further information, see section 3.7, Secure Messaging.

ASN 6-10: SecChannel: ASN.1 Definition for CSPSecureChannelSettings

```
-- ASN1START
-- General secure channel settings for the CSP Instance.
CSPSecureChannelSettings ::= SEQUENCE {

    -- Secure channel timeout in seconds (TIMER_SECURITY_TIMEOUT).
    securityTimeout        CSPTIMEOUT OPTIONAL
}
-- ASN1STOP
```

6.4.5.3 CSPProtocolType

This data structure defines the list of available *Protocol Types*.

A protocol type is selected by the Client Application.

The protocol types supported can be detected using the CSPSecureChannelSupport structure.

ASN 6-11: SecChannel: ASN.1 Definition for CSPProtocolType

```
-- ASN1START
-- List of available protocol types for secure messaging.
CSPProtocolType ::= ENUMERATED {
```

```

-- PROTOCOL_PACE:
-- Password Authenticated Connection Establishment [TR-03110-3].
protocol-pace                (17),

-- PROTOCOL_EAC_ID:
-- Extended Access Control v2 (PACE, TA2, CA2, CA3) [TR-03110-3].
protocol-eac-id              (18),

-- PROTOCOL_EAC_MRTD:
-- Extended Access Control v1 (PACE, CA1, TA1) [ICAO 9303-11].
protocol-eac-mrtd            (19),

-- PROTOCOL_PACE_CAM:
-- PACE with Chip Authentication Mapping (PACE, CA1, TA1) [ICAO 9303-11].
protocol-pace-cam            (20),

-- PROTOCOL_SCP03:
-- GP Secure Channel Protocol '03' [GP Amd D].
protocol-scp03                (5),

-- PROTOCOL_SCP04:
-- GP Secure Channel Protocol '04' [GP Amd K].
protocol-scp04                (21)
}
-- ASN1STOP

```

6.4.5.4 CSPSecureChannelAlgorithms

This data structure represents a container for a resource-specific *Secure Channel Configuration*.

It can be configured for a resource using the CSPAlgorithms structure.

For further information, see section 3.7, Secure Messaging.

ASN 6-12: SecChannel: ASN.1 Definition for CSPSecureChannelAlgorithms

```

-- ASN1START
-- Secure channel configuration for a specific resource.
CSPSecureChannelAlgorithms ::= SEQUENCE {

    -- The security function of the protocol.
    securityFunction            CSPSecurityFunction
}
-- ASN1STOP

```

6.4.5.5 CSPSecurityFunction

This data structure defines the list of available *Security Functions*.

A security function can be selected using the CSPSecureChannelAlgorithms structure.

The security functions supported can be detected using the CSPSecureChannelSupport structure.

ASN 6-13: SecChannel: ASN.1 Definition for CSPSecurityFunction

```
-- ASN1START
-- List of available security functions for secure messaging establishment.
CSPSecurityFunction ::= ENUMERATED {

    -- SEC_PACE_PIN:
    -- Password Authenticated Connection Establishment (PACE) PIN [TR-03110-2].
    sec-pace-pin                (1),

    -- SEC_PACE_PUK:
    -- Personal Unblocking Key (PUK) for PACE [TR-03110-2].
    sec-pace-puk                (2),

    -- SEC_PACE_CAN:
    -- Personal Unblocking Key (PUK) for PACE [TR-03110-1].
    sec-pace-can                (3),

    -- SEC_PACE_MRZ:
    -- Personal Unblocking Key (PUK) for PACE [TR-03110-2].
    sec-pace-mrz                (4),

    -- SEC_TA_AT_ROOT:
    -- Authentication Terminal Root Certificate (AT-Root) for eID [TR-03110-2].
    sec-ta-at-root              (5),

    -- SEC_TA_IS_ROOT:
    -- Inspection System Root Certificate (IS-Root) for MRTD [TR-03110-2].
    sec-ta-is-root              (6),

    -- SEC_CA1:
    -- Chip Authentication v1 (CA1) for MRTD [TR-03110-1].
    sec-ca1                     (7),

    -- SEC_CA2:
    -- Chip Authentication v2 (CA2) group key for eID [TR-03110-2].
    sec-ca2                     (8),

    -- SEC_CA2_PRIVILEGED:
    -- Chip-specific CA2 key for Privileged Terminals eID [TR-03110-3].
    sec-ca2-privileged          (9),
```

```
-- SEC_CA3:
-- Chip Authentication v3 (CA3) group key for eID [TR-03110-2].
sec-ca3                                (10),

-- SEC_CA3_PSA:
-- Chip-specific CA3 key for Pseudonymous Secure Authentication (PSA) eID.
sec-ca3-psa                            (11),

-- SEC_TA_DV:
-- Output: Document Verify certificate received during EAC [TR-03110-3].
sec-ta-dv                              (12),

-- SEC_TA_TERMINAL:
-- Output: Certificate received from individual terminals [TR-03110-3].
sec-ta-terminal                        (13),

-- SEC_TA_AT_LINKED:
-- Output: Linked TA certificate for rollover in eID [TR-03110-3].
sec-ta-at-linked                      (14),

-- SEC_TA_IS_LINKED:
-- Output: Linked TA certificate for rollover in MRTD [TR-03110-3].
sec-ta-is-linked                      (15),

-- SEC_KENC:
-- Encryption Key for SCP03 [GP Amd D] and SCP04 [GP Amd K].
sec-kenc                              (32),

-- SEC_KMAC:
-- MAC Key for SCP03 [GP Amd D] and SCP04 [GP Amd K].
sec-kmac                              (33)
}
-- ASN1STOP
```

6.5 Confidential Data Transfer Module

The CSP may implement the ConfidentialModule, offering *Confidential Data Transfer* to transform confidential data between session-layer and storage-layer encryption; the choice shall be subject to *Modularity*.

The ConfidentialModule is an extension of the SecureChannelModule, providing the same functionality as specified in section 6.4, *Secure Channel Module*.

In addition to creating resources with USAGE_SECCHANNEL, the CSP Admin shall be able to create resources with USAGE_CONFIDENTIAL to configure the storage-layer cipher key via CSPCipherAlgorithms, selecting:

- The paddingAlgorithm from available *Padding Algorithms*.
- The cipherAlgorithm from available *Cipher Algorithms*.

The Client Application shall be able to pass the resourceIds referring to the secure channel configuration and the storage-layer cipher key resourceId to the sc.init operation and trigger:

- Encryption transfer from session-layer encryption to storage-layer encryption for incoming APDU messages using the sc.confidentialUnwrap operation.
- Encryption transfer from storage-layer encryption to session-layer encryption for outgoing APDU messages using the sc.confidentialWrap operation.
- Multi-part encryption transformation processes using the sc.updateConfidentialUnwrap and sc.updateConfidentialWrap operations.

The CSP Admin may detect whether the platform supports this module using the CSPEnforce command.

For further information, see section 3.7.4, Confidential Data Transfer.

6.5.1 Confidential Data Transfer Definitions

6.5.1.1 Confidential Data Transfer Error Codes

If ERROR_MODE_DETAILED is supported and activated, the ConfidentialModule shall use the error codes listed in Table 6-46 as reasons for exceptions.

For further information, see section 5.1.1.6, Error Types.

Table 6-46: Confidential Data Transfer Error Reason Codes

Reason	Description
0x2050	Illegal resource type: Only key resources can be used for confidential data transfer.
0x4050	Confidential transfer not initialized: Missing storage key; sc.init not invoked with USAGE_CONFIDENTIAL.
0x4051	Confidential wrapping not initialized: sc.initConfidentialWrap not invoked.
0x4052	Confidential unwrapping not initialized: sc.initConfidentialUnwrap not invoked.
0x4053	Confidential wrapping mode active: Finalize with sc.confidentialWrap before switching mode.
0x4054	Confidential unwrapping mode active: Finalize with sc.confidentialUnwrap before switching mode.
0x4054	Security not established: Insufficient for confidential data transfer.
0x5050	Wrong usage: The storage key is not configured for USAGE_SECCHANNEL.
0x8050	Unsupported: The confidential data transfer module is not supported.

6.5.2 Confidential Data Transfer Configuration

The ConfidentialModule does not have a dedicated configuration. It shares the same configuration as the *Secure Channel Configuration* and additionally reuses the paddingAlgorithm and cipherAlgorithm parameters from the *Cipher Module*.

6.5.3 Confidential Data Transfer Operations

The ConfidentialModule shall implement the operations listed in Table 6-47.

For each operation, the CSP shall handle supported *Counter Types*, *Timer Types*, and *Audit Events*.

The operations shall solely be utilizable with resources configured for USAGE_CONFIDENTIAL and shall comply with the *Access Restrictions on Confidential Data Transfer Operations* specified in Table 6-48.

Note: Specifically, resources with USAGE_CONFIDENTIAL, shall not be permitted for decryption using *Cipher Operations* or *Offloading Operations*.

For further information, see section 3.7.4, Confidential Data Transfer.

Table 6-47: Confidential Data Transfer Operations

Operation	Details
sc.init	<p>Extends sc.init to accept resources with USAGE_CONFIDENTIAL for providing the storage key, in addition to resources with USAGE_SECCHANNEL required by the protocol.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.ConfidentialDataTransfer.init(..) <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_CSP_ERROR
sc.initConfidentialWrap	<p>Prepare service to transfer data from storage-layer encryption to session encryption and sets it to confidential-wrapping mode.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.ConfidentialDataTransfer.initConfidentialWrap(..) <p>Evaluate Timers:</p> <ul style="list-style-type: none"> TIMER_VALIDITY_DATE, TIMER_VALIDITY_PERIOD TIMER_SECURITY_TIMEOUT <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_TIMER_EXPIRED, EVENT_CSP_ERROR
sc.confidentialWrap	<p>Transfer data from storage-layer to session encryption.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.SecureChannelService.confidentialWrap(..) <p>Increment Counters:</p> <ul style="list-style-type: none"> COUNT_USAGE, COUNT_USAGE_PER_BLOCK COUNT_USAGE_SUCCESS_ONLY, COUNT_USAGE_FAILURE_ONLY <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_COUNTER_EXHAUSTED, EVENT_CSP_ERROR

Operation	Details
sc.updateConfidentialWrap	Multipart transformation from storage-layer to session encryption. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.SecureChannelService.updateConfidentialWrap.. Increment Counters: <ul style="list-style-type: none"> COUNT_USAGE_PER_BLOCK Fire Events: <ul style="list-style-type: none"> EVENT_COUNTER_EXHAUSTED, EVENT_CSP_ERROR
sc.initConfidentialUnwrap	Prepare the service to transfer data from session to storage-layer encryption and sets it to confidential-unwrapping mode. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api. ConfidentialDataTransfer.initConfidentialUnwrap Evaluate Timers: <ul style="list-style-type: none"> TIMER_VALIDITY_DATE, TIMER_VALIDITY_PERIOD TIMER_SECURITY_TIMEOUT Fire Events: <ul style="list-style-type: none"> EVENT_TIMER_EXPIRED, EVENT_CSP_ERROR
sc.confidentialUnwrap	Transfer data from session to storage-layer encryption. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.SecureChannelService.confidentialTransfer(..) Increment Counters: <ul style="list-style-type: none"> COUNT_USAGE, COUNT_USAGE_PER_BLOCK COUNT_USAGE_SUCCESS_ONLY, COUNT_USAGE_FAILURE_ONLY Fire Events: <ul style="list-style-type: none"> EVENT_COUNTER_EXHAUSTED, EVENT_CSP_ERROR
sc.updateConfidentialUnwrap	Multipart transformation from session to storage-layer encryption. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api. SecureChannelService.updateConfidentialTransfer(..) Increment Counters: <ul style="list-style-type: none"> COUNT_USAGE_PER_BLOCK Fire Events: <ul style="list-style-type: none"> EVENT_COUNTER_EXHAUSTED, EVENT_CSP_ERROR

6.5.3.1 Access Restrictions on Confidential Data Transfer Operations

The ConfidentialModule shall enforce the access restrictions for *Confidential Data Transfer Operations* listed in Table 6-48 for all *Access Rights*, *Usage Types*, *Resource States*, and *Policy Types* supported by the platform. Unauthorized or illegal operation invocations shall be rejected and handled according to the errorMode configured.

Table 6-48: Confidential Data Transfer Access Restrictions

Operation	Right Required	Usage Required	State Required	Other Restrictions
sc.init	USE	CONFIDENTIAL SECCHANNEL	-	<ul style="list-style-type: none"> Only consistent <i>Protocol, Resource, Size, and Curve Combinations</i> Only consistent <i>Cipher, Padding, and Key Size Combinations</i>
sc.initWrap	-	-	-	<ul style="list-style-type: none"> Not possible in confidential-wrapping.
sc.initUnwrap	-	-	-	<ul style="list-style-type: none"> Not possible in wrapping mode. Not possible in confidential-wrapping.
sc.initConfidentialWrap	-	-	OPERATIONAL	<ul style="list-style-type: none"> Not possible when not authenticated. Not possible in wrapping mode. POLICY_SECCHANNEL_ESTABLISHED POLICY_PASSWORD POLICY_TA2_ACCESS_FLAG
sc.initConfidentialUnwrap	-	-	OPERATIONAL	<ul style="list-style-type: none"> Not possible when not authenticated. Not possible in confidential-wrapping. POLICY_SECCHANNEL_ESTABLISHED POLICY_PASSWORD POLICY_TA2_ACCESS_FLAG
sc.updateConfidentialWrap sc.confidentialWrap	-	-	-	<ul style="list-style-type: none"> Only possible in wrap mode. Only possible when authenticated.
sc.updateConfidentialUnwrap sc.confidentialUnwrap	-	-	-	<ul style="list-style-type: none"> Only possible in confidential-unwrap. Only possible when authenticated.

6.5.3.2 Sensitive Results computed by Confidential Data Transfer Operations

The ConfidentialModule shall temporarily store the results of methods listed in Table 6-49 in transient memory and shall implement the assertSensitiveResult operation, enabling Client Applications to verify that these results were computed without any abnormality.

Table 6-49: Confidential Data Transfer Operations Requiring Sensitive Results Checks

Operation	Result Type	Result Description
sc.confidentialWrap	short	The length of the decrypted and wrapped data.
sc.confidentialUnwrap	short	The length of the unwrapped and decrypted data.
sc.updateConfidentialWrap	short	The length of the decrypted and wrapped data.
sc.updateConfidentialUnwrap	short	The length of the unwrapped and decrypted data,

6.5.3.3 No Sensitive Arrays required for Confidential Data Transfer Operations

The ConfidentialModule does not require integrity protection using *Sensitive Arrays*.

6.5.4 Confidential Data Transfer Lifecycle

The confidential data transfer service follows the same lifecycle as the secure channel service illustrates in Figure 6-4 but introduces additional modes for confidential wrap and confidential unwrap, as shown in Figure 6-5. These modes are triggered by *Confidential Data Transfer Operations*. A confidential data transfer service is an instance of the ConfidentialModule.

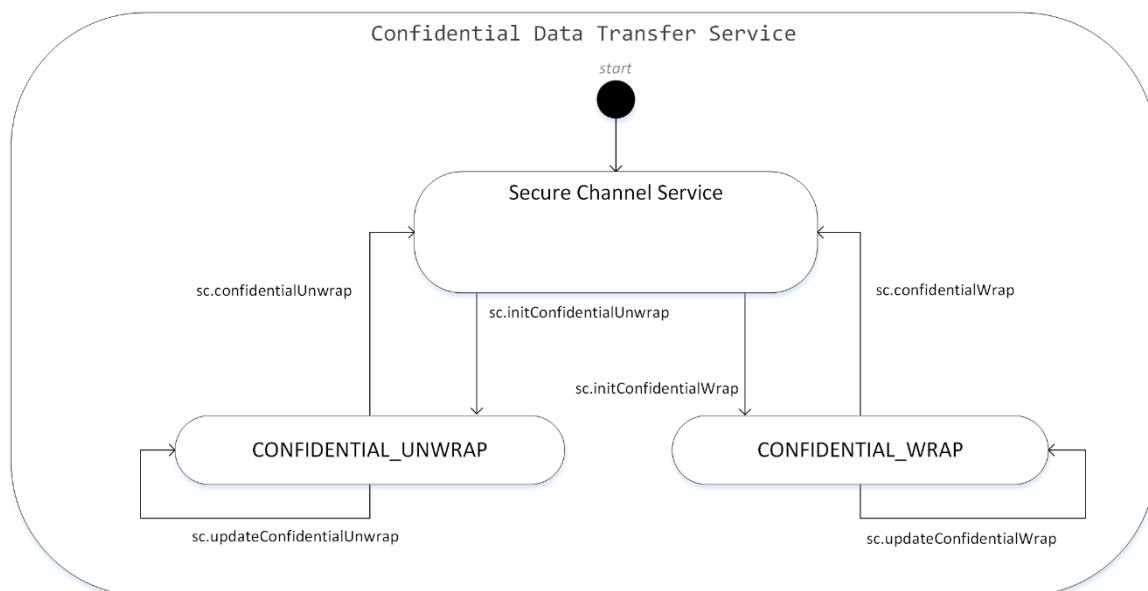
The `sc.confidentialWrap` and the `sc.confidentialUnwrap` operations must be initialized using either `sc.initConfidentialWrap` or `sc.initConfidentialUnwrap`. When the service is in wrapping or confidential wrapping mode, the data must be finalized before switching to any other mode. In contrast, unwrapping mode and confidential unwrapping mode can be interrupted at any time.

The confidential data transfer service stores *Sensitive Results computed by Confidential Data Transfer Operations*.

The confidential data transfer service triggers lifecycle changes to the resources used, as specified in

- Section 6.7.4 Key Lifecycle

Figure 6-5: Confidential Data Transfer Lifecycle



6.6 Attestation Module

The CSP may implement the `AttestationModule` to offer resource-related *Attestations* services; the choice shall be subject to *Modularity*.

The CSP Admin shall be able to create resources with `USAGE_ATTESTATION` and configure `attestationAlgorithms` using the `CSPSignatureAlgorithm` structure to select:

- The `paddingAlgorithm` from available *Padding Algorithms*.
- The `signatureAlgorithm` from available *Signature Algorithms*.
- The `messageDigestAlgorithm` from available *Message Digest Algorithms*.
- Optional `fieldsAddedAsPrefix` and `fieldsAddedAsSuffix` from available *Fields*.

The Client Application shall be able to compute attestations using these resources through `att.computeAttestation`:

- With `ATTESTATION_DATA` to compute *Data Attestation*.
- With `ATTESTATION_KEY_GENERATION` to compute a *Generate Key Pair Attestation*.
- With `ATTESTATION_KEY_POP` to compute *Proof of Possession Key Attestation*.

The CSP Admin may detect whether the platform supports this module using the `CSPEnforce` command.

Note: *Platform Attestation* and *Config Attestation* are specified in section 5.1.1.3, *System Attestations*.

For further information, see section 3.8, *Attestation*.

6.6.1 Attestation Definitions

6.6.1.1 Resource Attestation Types

The `AttestationModule` may support the attestation types listed in Table 6-50; the exact list shall be subject to *Modularity*.

The Client Applications selects the `attestationType` to be computed using the `att.computeAttestation` operation.

The CSP Admin may detect if an attestation type is supported using the `CSPAttestationSupport` structure. Each `CSPResourceAttestationType` supported shall adhere to the algorithm combinations specified in:

- Table 6-51 for compatible *Attestation Type, Component, and Algorithm Combinations*

For further information, see section 5.1.1.3, *System Attestation Types*.

Table 6-50: Resource Attestation Types

Attestation Type	Value (Hex)	Value (Int)	Description
ATTESTATION_DATA	0x03	3	<p><i>Data Attestation</i> generates signed attestations by combining external input data with CSP-internal data, such as a public key, a counter, a timer or additional <i>Fields</i>, and signing them using an attestation key.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Format: <code>CSPDataAttestation</code>

Attestation Type	Value (Hex)	Value (Int)	Description
ATTESTATION_KEY_POP	0x04	4	<p><i>Proof of Possession Key Attestation</i> (PoP Key Attestation) verifies that the private counterpart of a public key is managed by the same CSP Instance.</p> <p>This attestation is applicable to public keys: The corresponding private key signs a PoP (inner signature) and this data is then attested with an attestation key (outer signature) managed by the same CSP Instance.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Format: CSPKeyPoPAttestation
ATTESTATION_KEY_GENERATION	0x05	5	<p><i>Generate Key Pair Attestation</i> (Key Generation Attestation) functions same as ATTESTATION_KEY_POP with the difference that the value of the public-private key pair to attest will be freshly generated before computing the PoP attestation.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Format: CSPKeyPoPAttestation
(vendor-specific)	0xF0-0xFF	240-255	Reserved for vendor-specific attestation types.

6.6.1.2 Attestation Type, Component, and Algorithm Combinations

The AttestationModule shall support the compatible combinations of *Resource Attestation Types* listed in Table 6-51 for all *Resource Types*, *Key Types*, and *Signature Algorithms* supported by the platform. Unsupported or illegal combinations shall be rejected and handled according to the errorMode configured.

Table 6-51: Attestation Type, Component, Key, and Algorithm Combinations

Attestation Type	Attested Component	Attestation Computation
ATTESTATION_PLATFORM	SE Platform	<i>Platform Attestation</i> utilizing the CASD attestation mechanism specified in [GP Amd A] section 3.3.
ATTESTATION_CONFIG	CSP Instance	<i>Config Attestation</i> utilizing the <i>Signature, Padding, Hash, Key Size, and Curve Combinations</i> specified in Table 6-16.
ATTESTATION_DATA	KEY_ECC_PUBLIC KEY_RSA_PUBLIC RESOURCE_COUNTER RESOURCE_TIMER	<i>Data Attestation</i> utilizing the <i>Signature, Padding, Hash, Key Size, and Curve Combinations</i> specified in Table 6-16.
ATTESTATION_KEY_POP	KEY_ECC_PRIVATE KEY_ECC_PUBLIC KEY_RSA_PRIVATE KEY_RSA_PUBLIC	<i>Proof of Possession Key Attestation</i> utilizing the <i>Signature, Padding, Hash, Key Size, and Curve Combinations</i> specified in Table 6-16 for both the attestation private key and the private part of the key being attested, allowing both to use different algorithms and parameters.

6.6.1.3 Attestation Error Codes

If ERROR_MODE_DETAILED is supported and activated, the AttestationModule shall use the error codes listed in Table 6-52 as reasons for exceptions.

For further information, see section 5.1.1.6, Error Types.

Table 6-52: Attestation Error Reason Codes

Reason	Description
0x2060	Illegal resource type: Only public keys, counters and timers can be used for ATTESTATION_DATA.
0x4060	Attestation not initialized: The service has not been successfully initialized through any of the init methods.
0x5060	Wrong usage: Private attestation key is not configured for USAGE_ATTESTATION.
0x8060	Unsupported: The attestation module is not supported.
0x8061	Unsupported: The data attestation is not supported.
0x8062	Unsupported: The proof-of-possession key attestation is not supported.

6.6.2 Attestation Configuration

The AttestationModule shall support the configuration parameters defined in Table 6-53.

Table 6-53: Attestation Configuration Parameters

Parameter	Description	Parameter Type
attestationType	<p>The type of the attestation to be computed, either used to compute <i>System Attestations</i> or to compute <i>Resource Attestation Types</i> such as resource value or public key attestation.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Data type: CSPSystemAttestationType or CSPResourceAttestationType Operations: att.computeAttestation, CSPSystemAttestation Administration: CSPAttestationSupport 	Method Parameter
inputData	<p>Additional data provided as parameter to the attestation operations. It may be used as unique nonce to prevent replay attacks or as additional data added to the attestation.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Data type: CSPChallenge Attestations: all 	Method Parameter
attestationAlgorithm	<p>Signature, padding and message digest algorithm, along with signature fields, configured for a resource with USAGE_ATTESTATION, selected from the available <i>Signature Algorithms</i>, <i>Message Digest Algorithms</i>, <i>Padding Algorithms</i>, and <i>Fields</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Data type: CSPAttestationAlgorithms Administration: CSPAlgorithms 	Resource Parameter

6.6.3 Attestation Operations

The CSP¹⁹ shall implement the operations listed in Table 6-54.

For each operation, the CSP shall handle supported *Counter Types*, *Timer Types*, and *Audit Events*.

¹⁹ *Attestation Operations* are mandatory, as they are required for *System Attestations* (see section 5.1.1.3).

The operations shall solely utilize resources configured for USAGE_ATTESTATION as attestation keys and shall comply with the *Access Restrictions on Attestation Operations* specified in Table 6-55.

Note: Specifically, the resources with USAGE_CONFIDENTIAL, shall not be permitted for signature creation using *Signature Operations* or *Audit Operations*.

For further information, see section 3.8, Attestations.

Table 6-54: Attestation Operations

Operation	Details
att.init	<p>Initializes this service for an attestationType, selected from <i>System Attestations</i> or available <i>Resource Attestation Types</i>, and resources required.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.AttestationService.init(..) <p>Evaluate Timers:</p> <ul style="list-style-type: none"> TIMER_VALIDITY_DATE, TIMER_VALIDITY_PERIOD <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_TIMER_EXPIRED, EVENT_CSP_ERROR
att.computeAttestation	<p>Computes the attestation according to the attestationType set.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.AttestationService.computeAttestation(..) <p>CSP Protocol:</p> <ul style="list-style-type: none"> CSPSystemAttestation, CSPResourceAttestation <p>Increment Counters:</p> <ul style="list-style-type: none"> COUNT_USAGE, COUNT_USAGE_PER_BLOCK COUNT_USAGE_SUCCESS_ONLY, COUNT_USAGE_FAILURE_ONLY <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_COUNTER_EXHAUSTED, EVENT_CSP_ERROR Fires no signature events.
att.updateInputData	<p>Optional method to set multipart input data to be included in the attestation result.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.AttestationService.updateInputData(..)
att.update	<p>Optional method to handle multi-part attestation computation.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.AttestationService.update(..) <p>Increment Counters:</p> <ul style="list-style-type: none"> COUNT_USAGE_PER_BLOCK <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_COUNTER_EXHAUSTED, EVENT_CSP_ERROR Fires no signature events.
att.getAttestationLength	<p>Retrieve the size, in bytes, of the output buffer required to write the computed attestation result.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.AttestationService.getAttestationLength(..)

6.6.3.1 Access Restrictions on Attestation Operations

The CSP shall enforce the restrictions for *Attestation Operations* listed in Table 6-55 for all *Access Rights*, *Usage Types*, *Resource States*, and *Policy Types* supported by the platform. Unauthorized or illegal operation invocations shall be rejected and handled according to the *errorMode* configured.

Table 6-55: Attestation Operations Access Restrictions

Operation	Right Required	Usage Required	State Required	Other Restrictions
att.init (private attestation key)	USE	ATTESTATION	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required Only consistent <i>Attestation Type, Component, and Algorithm Combinations</i> POLICY_KEYPAIR POLICY_ASSOCIATION Any counter exhausted Any timer expired
att.init (resource to attest)	USE	-	OPERATIONAL	
att.init (private key to compute PoP)	USE	SIGNATURE	OPERATIONAL	
att.init (public keys)	USE	-	OPERATIONAL	
att.updateInputData	-	-	-	
att.update	-	-	OPERATIONAL	
att.computeAttestation	-	-	OPERATIONAL	
att.getAttestationLength	-	-	OPERATIONAL	

6.6.3.2 Sensitive Results Computed by Attestation Operations

The CSP shall temporarily store the results of methods listed in Table 6-56 in transient memory and shall implement the *assertSensitiveResult* operation, enabling Client Applications to verify that these results were computed without any abnormality.

Table 6-56: Attestation Operations Requiring Sensitive Results Checks

Operation	Result Type	Result Description
att.computeAttestation	short	The length of the created attestation data.

6.6.3.3 Sensitive Arrays Required for Attestation Operations

The CSP shall invoke `javacard.framework.SensitiveArrays.assertIntegrity(..)` ([JCAPI]) on the parameters listed in Table 6-57.

For further information, see section 5.1.3.4, Sensitive Arrays, and section 8.3, Byte Buffer Parameters.

Table 6-57: Attestation Operations Requiring Sensitive Arrays

Operation	Parameter	Description
att.updateInputData	Input data	Additional data, challenge or nonce to include in the attestation result.
att.update	Output buffer	The multi-part attestation data computed by the CSP.
att.computeAttestation	Output buffer	The attestation data computed by the CSP.

6.6.4 Attestation Lifecycle

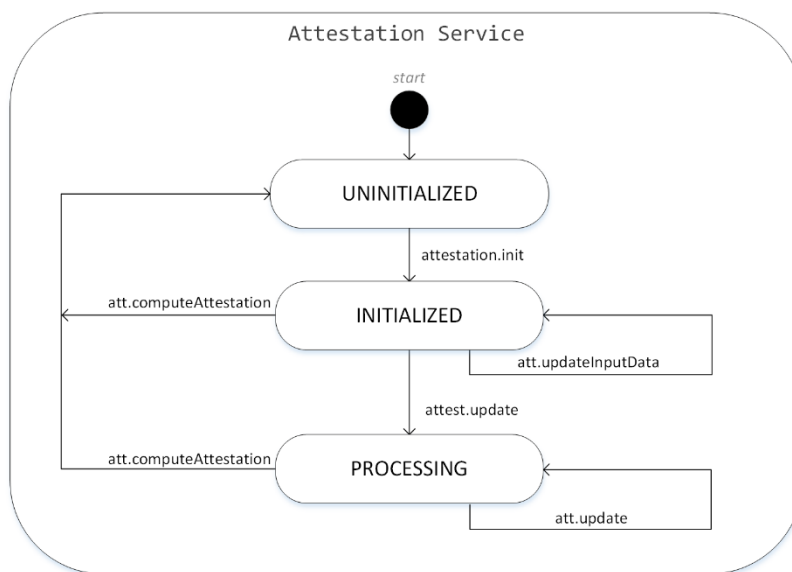
Figure 6-6 shows lifecycle changes of an attestation service triggered by *Attestation Operations*. An attestation service is an instance of the *AttestationModule*. When a new instance is created, it starts in the UNINITIALIZED state. In this state, it is not possible to compute an attestation. The Client Application needs to invoke the `att.init` operation to configure the `attestationType` along with the resources required before computing the attestation. Before starting the computation, the Client Application may optionally provide input data using the `att.updateInputData` operation. After completing the `att.computeAttestation` operation, the attestation service will return to the UNINITIALIZED state.

The attestation service stores *Sensitive Results Computed by Attestation Operations*.

The attestation service triggers lifecycle changes to the resources used, as specified in

- Section 6.7.4 *Key Lifecycle*

Figure 6-6: Attestation Lifecycle



6.6.5 Attestations Structures

This section lists ASN.1 structures related to the configuration of the attestation service.

Note: A complete ASN.1 definition is available in [GP CSP ASN1].

6.6.5.1 CSPAttestationSupport

This data structure represents features, types, and algorithms of the *Attestation Module*.

It is included in the CSPEnforce command to detect their support by the platform.

ASN 6-14: Attestation: ASN.1 Definition for CSPAttestationSupport

```

-- ASN1START
-- Checks for attestation functionality support and supported algorithms.
CSPAttestationSupport ::= SEQUENCE {

    -- Resource attestation types, such as Data or PoP attestations.
    resourceAttestationTypes    SET OF CSPResourceAttestationType OPTIONAL
}
  
```

```
-- ASN1STOP
```

6.6.5.2 CSPSystemAttestationType

This data structure defines the list of available *System Attestations*.

It is used in the CSPSystemAttestation command.

ASN 6-15: Attestations: ASN.1 Definition for CSPSystemAttestationType

```
-- ASN1START
-- List of available system attestations.
CSPSystemAttestationType ::= ENUMERATED {

    -- ATTESTATION_PLATFORM:
    -- Attestation of the SE Platform the CSP is operated on.
    attestation-platform          (1),

    -- ATTESTATION_CONFIG:
    -- Attestation of the configuration of this CSP Instance.
    attestation-config            (2)
}
-- ASN1STOP
```

6.6.5.3 CSPResourceAttestationType

This data structure defines the list of available *Resource Attestation Types*.

The attestation types supported can be detected using the CSPAttestationSupport structure.

ASN 6-16: Attestations: ASN.1 Definition for CSPResourceAttestationType

```
-- ASN1START
-- List of available resource attestations.
CSPResourceAttestationType ::= ENUMERATED {

    -- ATTESTATION_DATA:
    -- Attestation of external data with public key, counter or timer.
    attestation-data              (3),

    -- ATTESTATION_KEY_POP:
    -- Attestation of public key with proof of possession for the private key.
    attestation-key-pop           (4),

    -- ATTESTATION_KEY_GENERATION:
    -- Generate a new key pair and returns a PoP attestation.
    attestation-key-generation    (5)
}
-- ASN1STOP
```

6.6.5.4 CSPAttestationAlgorithms

This data structure represents an algorithm configuration used for the `configAttestationKey`, the `auditSigningKey` but also as `attestationAlgorithm` for other key resources with `USAGE_ATTESTATION`.

It can be configured for a resource using the `CSPAlgorithms` structure.

For further information, see section 3.6.7, Signatures with Counters & Timestamps.

ASN 6-17: Signature: ASN.1 Definition for CSPAttestationAlgorithms

```
-- ASN1START
-- Attestation configuration for a specific resource.
CSPAttestationAlgorithms ::= SEQUENCE {

    -- The signature, padding and message digest algorithms.
    signatureAlgorithms      CSPSignatureAlgorithms,

    -- Add signature fields (e.g., counter) to the beginning of data to sign.
    fieldsAddedAsPrefix      SEQUENCE OF CSPField OPTIONAL,

    -- Add signature fields (e.g., counter) to the end of data to sign.
    fieldsAddedAsSuffix      SEQUENCE OF CSPField OPTIONAL
}
-- ASN1STOP
```

6.6.5.5 CSPPlatformAttestation

This data structure represents the result of a *Platform Attestation*.

The data to be signed is included in this response as the `signedPlatformData` element. Thereby, the entire `signedPlatformData` element, including the tags and lengths of the ASN.1 structure, is signed, except for the `SecurityEnvironmentTemplate`, which is appended by `org.globalplatform.AuthoritySignature` [GP API].

Note: The response of the platform attestation contains the AID of the CSP Instance. Therefore, the platform attestation will generate different results for different CSP Instances on the same SE platform.

For further information, see section 3.8.1, Platform Attestation.

ASN 6-18: Attestations: ASN.1 Definition for CSPPlatformAttestation

```
-- ASN1START
-- Response of a platform attestation.
CSPPlatformAttestation ::= SEQUENCE {

    -- The attestation data to be signed, including tags and lengths.
    signedPlatformData      SEQUENCE {

        -- Version of the CSP Admin Protocol used.
        cspProtocolVersion   CSPProtocolVersion,

        -- The type of the attestation being computed (i.e., platform).
```

```

    attestationType CSPSystemAttestationType DEFAULT attestation-platform,

    -- Information about the CSP platform.
    platform CSPPlatform,

    -- Platform DLOA XML according to A.1 / A.2.1 of [GP DLOA].
    platform-DLOA-XML OCTET STRING (SIZE(2048..32768)),

    -- Application DLOA according to A.1 / A.2.2 of [GP DLOA].
    csp-application-DLOA-XML OCTET STRING (SIZE(2048..32768)) OPTIONAL,

    -- Input data, e.g. a challenge, provided within the command.
    inputData OCTET STRING (SIZE (16..32768)) OPTIONAL,

    -- Data structure according to [GP Amd A] section 5.3.1.
    SecurityEnvironmentTemplate SecurityEnvironmentTemplate
},

-- Signature over the entire signedPlatformData according to [GP Amd A].
signature CSPSignature
}

-- Copied from [GP Amd A] section 5.3.1 to avoid ASN.1 compiler issues.
ClientApplicationInformation ::= SEQUENCE {

    -- The signature mode selected by the CSP Application with init().
    requestedSignatureMode [0] OCTET STRING (SIZE(1)),

    -- The AID of the CSP Application.
    instanceAID [1] OCTET STRING (SIZE(5..16)),

    -- The AID of the CSP ELF.
    executableLoadFileAID [2] OCTET STRING (SIZE(5..16)),

    -- The version of the CSP ELF; e.g., 2-byte major, minor for CAP file.
    executableLoadFileVersionNumber [3] OCTET STRING
}

-- Data structure according to [GP Amd A] section 5.3.1.
SecurityEnvironmentTemplate ::= [APPLICATION 27] SEQUENCE {

    -- Information about the CSP Application.
    clientApplicationInformation [0] ClientApplicationInformation,

    -- Information about the algorithm used by the CASD.
    signatureAlgorithm [1] AlgorithmIdentifier,

```

```

    -- Identifier of the PK.CASD-SIGN.AUT to verify the signature ([GP Amd A]).
    keyIdentifier          [2] OCTET STRING
}

-- Data structure according to [GP Amd A] section 5.3.1.
AlgorithmIdentifier ::= SEQUENCE {

    -- The OID of an ECDSA algorithm as specified in [RFC 5758] section 3.2.
    algorithm             OBJECT IDENTIFIER
}

-- ASN1STOP

```

6.6.5.6 CSPConfigAttestation

This data structure represents the result of a *Config Attestation*.

The data to be signed is included in this response as the signedConfigData element. This signature includes the tags and lengths of the ASN.1 structure.

For further information, see section 3.8.2, Config Attestation.

ASN 6-19: Attestations: ASN.1 Definition for CSPConfigAttestation

```

-- ASN1START
-- Response of a config attestation.
CSPConfigAttestation ::= SEQUENCE {

    -- The attestation data to be signed, including tags and lengths.
    signedConfigData SEQUENCE {

        -- Version of the CSP Admin Protocol used.
        cspProtocolVersion    CSPProtocolVersion,

        -- The type of the attestation being computed (i.e., config).
        attestationType       CSPSystemAttestationType DEFAULT attestation-config,

        -- Additional Fields to be included; configured by the CSP Admin.
        fieldsAddedAsPrefix   SET OF CSPField OPTIONAL,

        -- Custom name set by the CSP Admin via CSPSetup.
        configName             CSPConfigName,

        -- Custom version set by the CSP Admin via CSPSetup.
        configVersion          CSPConfigVersion,

        -- Information about the CSP platform.
        cspPlatform            CSPPlatform,
    }
}

```

```
-- Input data, e.g. a challenge, provided within the command.
inputData          OCTET STRING (SIZE (16..32768)),

-- Additional Fields to be included; configured by the CSP Admin.
fieldsAddedAsSuffix SET OF CSPField OPTIONAL
},

-- Signature over the signedConfigData using the config attestation key.
signature          CSPSignature,

-- The algorithm used to sign the data.
signatureAlgorithm CSPSignatureAlgorithms,

-- Key parameters of the attestation key.
signaturKeySizeOrCurve CSPKeySizeOrCurve
}
-- ASN1STOP
```

6.6.5.7 CSPDataAttestation

This data structure represents the result of a *Data Attestation*.

The data to be signed is included in this response as the signedData element. This signature includes the tags and lengths of the ASN.1 structure.

For further information, see section 3.8.3, Resource Attestation.

ASN 6-20: Attestations: ASN.1 Definition for CSPDataAttestation

```
-- ASN1START
-- Response of a data attestation (signature over external and internal data).
CSPDataAttestation ::= SEQUENCE {

    -- The attestation data to be signed, including tags and lengths.
    signedData SEQUENCE {

        -- Version of the CSP Admin Protocol used.
        cspProtocolVersion CSPProtocolVersion,

        -- The type of the attestation being computed (i.e., data attestation).
        attestationType CSPResourceAttestationType DEFAULT attestation-data,

        -- Additional Fields to be included; configured by the CSP Admin.
        fieldsAddedAsPrefix SET OF CSPField OPTIONAL,

        -- The input data, e.g. a challenge, provided within the command.
        inputData          OCTET STRING (SIZE (16..32768)),
```

```

-- CSP-internal resource value: public key, counter or timer.
resourceValue          CSPResourceValue OPTIONAL,

-- Additional Fields to be included; configured by the CSP Admin.
fieldsAddedAsSuffix    SET OF CSPField OPTIONAL
},

-- Signature over the signedData using the attestation key.
signature              CSPSignature,

-- The algorithm used to sign the data.
signatureAlgorithm     CSPSignatureAlgorithms,

-- Key parameters of the attestation key.
signaturKeySizeOrCurve CSPKeySizeOrCurve
}
-- ASN1STOP

```

6.6.5.8 CSPKeyPoPAttestation

This data structure represents the result of a *Proof of Possession Key Attestation*.

The data to be signed is included in this response as the popData and signedPoPData elements. Both signatures include the tags and lengths of the ASN.1 structure.

For further information, see section 3.8.4, Key PoP Attestation.

ASN 6-21: Attestations: ASN.1 Definition for CSPKeyPoPAttestation

```

-- ASN1START
-- Response of a key attestation including a Proof of Possession (PoP).
CSPKeyPoPAttestation ::= SEQUENCE {

    -- The attestation data to be signed, including tags and lengths.
    signedPoPData SEQUENCE {

        -- Version of the CSP Admin Protocol used.
        cspProtocolVersion      CSPProtocolVersion,

        -- Proof Of Possession data to be signed, including tags and lengths.
        popData SEQUENCE {

            -- The type of the attestation being computed (i.e., PoP).
            attestationType CSPResourceAttestationType DEFAULT attestation-key-pop,

            -- Additional Fields to be included; configured by the CSP Admin.
            fieldsAddedAsPrefix SET OF CSPField OPTIONAL,

            -- Public key value to attest.

```

```

        publicKey          CSPResourceValue,

        -- Public att. key to verify the signature over signedPoPData.
        publicAttestationKey CSPResourceValue OPTIONAL,

        -- Input data, e.g., a challenge, provided within the command.
        inputData          OCTET STRING (SIZE (16..32768)),

        -- Additional Fields to be included; configured by the CSP Admin.
        fieldsAddedAsSuffix SET OF CSPField OPTIONAL
    },

    -- Signature over the popData using the private key.
    popDataSignature       CSPSignature
},

-- Signature over the entire signedPoPData using the attestation key.
signature                CSPSignature,

-- The algorithm used to sign the signedPoPData.
signatureAlgorithm        CSPSignatureAlgorithms,

-- Key parameters of attestation key.
signaturKeySizeOrCurve    CSPKeySizeOrCurve,

-- The algorithm used to sign the popData.
popSignatureAlgorithm     CSPSignatureAlgorithms,

-- Key parameters of the private key.
popSignaturKeySizeOrCurve CSPKeySizeOrCurve
}
-- ASN1STOP

```

6.7 Key Module

The CSP may implement the KeyModule to offer *Key Management* services; the choice shall be subject to *Modularity*.

The CSP Admin shall be able to create resources of type RESOURCE_KEY with the following configurations:

- keyType and keySize from available *Key Types* and *Key Sizes*.
- curve for Elliptic Curve Cryptography (ECC) chosen from supported *ECC Curves*.
- Mark a key resource as transient for use as an ephemeral key.
- Configure keyDerivationAlgorithm with USAGE_KEY when used as the base for key derivation with keyDerivationHashAlgorithm from available *Message Digest Algorithms*.
- Configure keyAgreementScheme with USAGE_KEY when used for key agreement.

The CSP Admin shall be able to initialize these resources using either CSPSetValue, CSPGenerateKey, CSPComputePublicKey, or CSPDeriveKey of the CSP Protocol.

The Client Application may also trigger key management operations by passing the resourceId of key resources to the following key service operations:

- Import and export public keys via key.initManage, key.manage, and key.updateManage.
- Key generation through key.generate, key.generateKeyPair, or key.computePublicKey.
- Key derivation through key.derive and key agreement through key.computeSharedSecret.

The CSP Admin may detect whether the platform supports this module using the CSPEnforce command.

For further information, see section 3.3, Key Management.

6.7.1 Key Definitions

6.7.1.1 Key Types

The KeyModule may support the key types listed in Table 6-58; the exact list shall be subject to *Modularity*. The key type can be configured for resources of type RESOURCE_KEY.

The CSP Admin may select the keyType using CSPKeyType structure.

The CSP Admin may detect if a key type is supported using the CSPKeySupport structure. Each key type supported shall adhere to the algorithm combinations specified in:

- Table 6-4 for compatible *Cipher, Padding, and Key Size Combinations*
- Table 6-16 for compatible *Signature, Padding, Hash, Key Size, and Curve Combinations*
- Table 6-31 for compatible *Protocol, Resource, Size, and Curve Combinations*
- Table 6-51 for compatible *Attestation Type, Component, and Algorithm Combinations*
- Table 6-62 for compatible *Key Generation Processes*
- Table 6-64 for compatible *Key Derivation, Resource Type, Hash, Size, and Curve Combinations*
- Table 6-67 for compatible *Key Agreement, Key Type, Curve, and Size Combinations*

Note: The encoding specified in the table shall be used for the import and export functionalities of the CSP. However, key values may be stored or encoded differently within the CSP Application.

For further information, see section 3.1.3, Load Resources, section 3.3.1, Key Generation, and section 3.3.4, Import/Export Public Keys.

Table 6-58: Key Types

Key Type	Value (Hex)	Value (Int)	Description
KEY_AES	0x01	1	<p>Symmetric cryptographic key used with Advanced Encryption Standard (AES) utilized for encryption and decryption tasks and for generating Message Authentication Codes (MACs).</p> <p>AES keys can be imported by CSP Admin, or can be generated, or derived by CSP Admin or Client Applications.</p> <p>AES keys are encoded as a byte array representing a large unsigned integer in big-endian format by using the following structure: [integer_bytes].</p> <p>The length of this byte array must match the key size configured within the CSP.</p>
KEY_HMAC	0x02	2	<p>Symmetric cryptographic key used for generating hash-based message authentication codes to create HMAC-based signatures.</p> <p>HMAC keys can either be imported or generated by the CSP Admin or by Client Applications.</p> <p>HMAC keys are encoded as a byte array representing a large unsigned integer in big-endian format by using the following structure: [integer_bytes].</p> <p>The length of this byte array must match the key size configured within the CSP.</p>
KEY_ECC_PUBLIC	0x03	3	<p>Public key used for Elliptic Curve Cryptography (ECC).</p> <p>ECC public keys can be imported or computed from ECC private keys by CSP Admin or Client Applications.</p> <p>Public ECC key values are encoded according to section 3.2.1 in [X9.62]. This encoding includes both the x and y coordinates of the point on the curve by using the following structure: 0x04[X_bytes][Y_bytes], where 0x04 indicates that the uncompressed format, as defined in [X9.62], is used.</p> <p>The encoding requires the curve size to be known and compatible to the ECC public key resource configured within the CSP.</p> <p>When decoding, the CSP must verify that the point indeed lies on the elliptic curve and that it is not the point at infinity.</p>
KEY_ECC_PRIVATE ²⁰	0x04	4	<p>Private key used for Elliptic Curve Cryptography (ECC).</p> <p>ECC private keys can be imported by the CSP Admin, or generated or derived by CSP Admin or Client Applications.</p> <p>The private ECC key is encoded as a byte array representing a large unsigned integer in big-endian format by using the following structure: [integer_bytes].</p> <p>The length of this byte array must match the elliptic curve configured for the ECC key. The public key value may either be computed using this private key or imported alongside it.</p>

²⁰ KEY_ECC_PRIVATE is mandatory, as it is required for *Config Attestation* (see section 3.8.2).

Key Type	Value (Hex)	Value (Int)	Description
KEY_RSA_PUBLIC	0x05	5	<p>Public key for Rivest-Shamir-Adleman (RSA) algorithms.</p> <p>RSA public keys can be imported or can be computed from RSA private keys by CSP Admin or Client Applications.</p> <p>The public RSA key is encoded in raw format as a byte array including the modulus (n) and the public exponent (e) as an unsigned integer in big-endian byte order by using the following structure: [modulus_bytes][exponent_3bytes].</p> <p>The length of the modulus byte array must match the RSA key size (e.g., 256 bytes for a 2048-bit key). The public exponent has a fixed size of 3 bytes for the common value of 65537.</p>
KEY_RSA_PRIVATE	0x06	6	<p>Private key for Rivest-Shamir-Adleman (RSA) algorithms.</p> <p>RSA private keys can be imported by the CSP Admin, or generated or derived by CSP Admin or Client Applications.</p> <p>The private RSA key is encoded in raw format as a byte array including the modulus (n) and the private exponent (d) as an unsigned integer in big-endian byte order by using the following structure: [modulus_bytes][exponent_bytes].</p> <p>The length of these arrays must match the RSA key size configured within the CSP (e.g., 256 bytes for a 2048-bit key). The public key value may either be computed using this private key or imported alongside it.</p>
KEY_MASTER_SECRET	0x07	7	<p>A master secret is either imported into or generated within the CSP and is used as input for <i>Key Derivation</i>.</p> <p>Secrets are encoded as a byte array representing a large unsigned integer in big-endian format by using the following structure: [integer_bytes]. The length of this byte array must match the key size configured within the CSP. The CSP shall not support setting secrets using the CSPSetValue command. The CSP shall throw ERROR_ILLEGAL_CONFIG if the secret is used as input for any other cryptographic operation besides <i>Key Derivation</i>.</p>
KEY_DERIVED_SECRET	0x08	8	<p>A derived secret is a result of <i>Key Derivation</i> and is used as input for further key derivation processes.</p> <p>Secrets are encoded as a byte array representing a large unsigned integer in big-endian format by using the following structure: [integer_bytes]. The length of this byte array must match the key size configured within the CSP. The CSP shall not support setting secrets using the CSPSetValue command. The CSP shall throw ERROR_ILLEGAL_CONFIG if the secret is used as input for any other cryptographic operation besides <i>Key Derivation</i>.</p>

Key Type	Value (Hex)	Value (Int)	Description
KEY_SHARED_SECRET	0x09	9	<p>A shared secret is a result of <i>Key Agreement</i>, such as <i>ECKA-DH</i> and <i>ECKA-EG</i>, and can only be used as input for <i>Key Derivation</i>.</p> <p>Secrets are encoded as a byte array representing a large unsigned integer in big-endian format by using the following structure: [integer_bytes]. The length of this byte array must match the key size configured within the CSP.</p> <p>The CSP shall not support setting secrets using the CSPSetValue command.</p> <p>The CSP shall throw ERROR_ILLEGAL_CONFIG if the secret is used as input for any other cryptographic operation besides <i>Key Derivation</i> and <i>Key Agreement</i>.</p>

6.7.1.2 Key Sizes

The KeyModule may support the key sizes listed in Table 6-59; the exact list shall be subject to *Modularity*. The key size can be configured for resources of type RESOURCE_KEY.

The CSP Admin may select the keySize using CSPKeySize structure.

The CSP Admin may detect if a key size is supported using the CSPKeySupport structure. Each key size supported shall adhere to the algorithm combinations specified in:

- Table 6-4 for compatible *Cipher, Padding, and Key Size Combinations*
- Table 6-16 for compatible *Signature, Padding, Hash, Key Size, and Curve Combinations*
- Table 6-31 for compatible *Protocol, Resource, Size, and Curve Combinations*
- Table 6-62 for compatible *Key Generation Processes*
- Table 6-64 for compatible *Key Derivation, Resource Type, Hash, Size, and Curve Combinations*
- Table 6-67 for compatible *Key Agreement, Key Type, Curve, and Size Combinations*

For further information, see section 3.3.1, Key Generation.

Table 6-59: Key Sizes

Key Size	Value (Hex)	Value (Int)	Details	Key Types
KEY_AES_128 ²¹	0x01	1	Advanced Encryption Standard (AES) are available in 128 bit and 256 bit key sizes.	KEY_AES
KEY_AES_256	0x03	2		
KEY_AES_2_128	0x05	5	Cipher mode XTS requires two AES keys. These keys, managed by the CSP within a single key resource, are available in sizes of 2x128 bit or 2x256 bit. For details, see CIPHER_AES_XTS.	
KEY_AES_2_256	0x07	7		
KEY_HMAC_256	0x0A	10	Keys for HMAC-based signatures are available in 256 bit, 384 bit and 512 bit sizes.	KEY_HMAC
KEY_HMAC_384	0x0B	11		
KEY_HMAC_512	0x0C	12		
KEY_ECC_256	0x10	16		KEY_ECC_PUBLIC

²¹ KEY_AES_128 is mandatory, as it is required for PROTOCOL_PACE (see section 6.4.1.1).

Key Size	Value (Hex)	Value (Int)	Details	Key Types
KEY_ECC_384	0x11	17	Public and private keys for Elliptic Curve Cryptography are available for 256 bit, 384 bit, 512 bit and 521 bit curves.	KEY_ECC_PRIVATE
KEY_ECC_512	0x12	18		
KEY_ECC_521	0x13	19		
KEY_RSA_2048 ²²	0x1A	26	Public and private keys for Rivest-Shamir-Adleman (RSA) are available for 2048 bit, 3072 bit and 4096 bit.	KEY_RSA_PUBLIC KEY_RSA_PRIVATE
KEY_RSA_3072	0x1B	27		
KEY_RSA_4096	0x1C	28		
KEY_SECRET_128	0x20	32	Master secrets used as the basis for key derivation, derived secrets resulting from key derivation, and shared secrets generated through key agreement processes are available in 128 bit, 256 bit, 384 bit, and 512 bit sizes.	KEY_MASTER_SECRET KEY_DERIVED_SECRET KEY_SHARED_SECRET
KEY_SECRET_256	0x21	33		
KEY_SECRET_384	0x22	34		
KEY_SECRET_512	0x23	35		
KEY_SECRET_576	0x24	36	Specific secret size required when using KDF_ECC to derive ECC keys for CURVE_BRAINPOOL_P512_R1 and CURVE_SEC_P521_R1.	KEY_MASTER_SECRET KEY_DERIVED_SECRET

6.7.1.3 ECC Curves

The KeyModule may support the ECC curve domain parameter sets listed in Table 6-60; the exact list shall be subject to *Modularity*. Each set includes predefined values for elliptic curves, such as the prime number (p), base point (G), base point order (n) and other parameters, which collectively define the properties of a specific elliptic curve. These domain parameters can be configured for resources of type KEY_ECC_PRIVATE and KEY_ECC_PUBLIC.

The CSP Admin may select the curve using CSPCurve structure.

The CSP Admin may detect if specific domain parameters are supported using the CSPKeySupport structure. Each domain parameter set supported shall adhere to the algorithm combinations specified in:

- Table 6-16 for compatible *Signature, Padding, Hash, Key Size, and Curve Combinations*
- Table 6-31 for compatible *Protocol, Resource, Size, and Curve Combinations*
- Table 6-64 for compatible *Key Derivation, Resource Type, Hash, Size, and Curve Combinations*
- Table 6-67 for compatible *Key Agreement, Key Type, Curve, and Size Combinations*

For further information, see section 3.3.1, Key Generation.

Table 6-60: Curves for ECC Curves

ECC Domain Parameter	Value (Hex)	Value (Int)	Description
CURVE_BRAINPOOL_P256_R1	0x01	1	A 256-bit Brainpool curve 'brainpoolP256r1', offering 128-bit security, as specified in [RFC 5639] section 3.4.
CURVE_BRAINPOOL_P384_R1	0x02	2	A 384-bit Brainpool curve 'brainpoolP384r1', offering 192-bit security, as specified in [RFC 5639] section 3.6.

²² KEY_RSA_2048 is considered as *Legacy Cryptographic Algorithms* (see section 2.4.2).

ECC Domain Parameter	Value (Hex)	Value (Int)	Description
CURVE_BRAINPOOL_P512_R1	0x03	3	A 512-bit Brainpool curve 'brainpoolP512r1', offering 256-bit security, as specified in [RFC 5639] section 3.7.
CURVE_SEC_P256_R1 ²³	0x04	4	NIST's P-256 curve, offering 128-bit security, as specified in [SP800-186] section 3.2.1.3.
CURVE_SEC_P384_R1	0x05	5	NIST's P-384 curve, offering 192-bit security, as specified in [SP800-186] section 3.2.1.4.
CURVE_SEC_P521_R1	0x06	6	NIST's P-521 curve, offering 256-bit security, as specified in [SP800-186] section 3.2.1.5.
CURVE_X25519	0x07	7	A 256-bit curve applicable to key agreement, offering 192-bit security, as specified in [RFC 7748] section 5. Note: This curve cannot be used for signatures.

6.7.1.4 Key Management Modes

The KeyModule shall support the key management operation modes listed in Table 6-1. This mode defines if the key service operates in import or export mode.

The Client Application may select the keyMode using the `key.initManage` operation.

Note: These modes function similarly to *Certificate Management Modes* and *Offloading Modes*.

Table 6-61: Key Management Modes

Mode	Value (Hex)	Value (Int)	Description
MANAGE_MODE_PUBLIC_KEY_IMPORT	0x01	1	Import public key. The KeyModule shall import the public key provided within the input buffer via <code>key.manage</code> or <code>key.updateManage</code> .
MANAGE_MODE_PUBLIC_KEY_EXPORT	0x02	2	Export public key. The KeyModule shall write the public key to the output buffer provided via <code>key.manage</code> or <code>key.updateManage</code> .

6.7.1.5 Key Generation Processes

The KeyModule shall offer key generation according to Table 6-62 for *Key Types* and *Key Sizes* supported by the platform. It shall use a Secure Random Number Generator (SRNG) as specified for the RandomDataModule.

The CSP Admin may generate keys using the CSPGenerateKey and CSPComputePublicKey commands.

The Client Application may trigger key generation using the `key.generate`, `key.generateKeyPair`, and `key.computePublicKey` operations, provided the ACCESS_SETUP right is configured for the resource.

Note: The key generation process specified in the table does not include key derivation. Key derivation algorithms are defined in the subsequent section.

For further information, see section 3.3.1, Key Generation.

²³ CURVE_SEC_P256_R1 is mandatory, as it is required for *Config Attestation* (see section 3.8.2)

Table 6-62: Key Generation Processes

Key Type	Key Sizes (bit)	Description of the Key Generation Process
KEY_AES	128, 256	AES keys shall be generated according to [FIPS 197], using a SRNG as specified above, to produce keys of either 128 bits or 256 bits. Regardless of the key size, AES operates with a fixed block size of 128 bits.
KEY_MASTER_SECRET KEY_DERIVED_SECRET KEY_SHARED_SECRET	128, 256, 384, 512, 521	Keys acting as secrets shall be generated using a SRNG defined above. The key size corresponds to length of the derived key.
KEY_HMAC	256, 384, 512	HMAC keys shall be generated using a SRNG defined above. The key size corresponds to length of the hash output (e.g., 256 bits for SHA-256, etc.).
KEY_ECC_PRIVATE KEY_ECC_PUBLIC	256, 384, 512, 521	<p>The private key for ECC shall be generated as a random integer according to algorithm 2 in section 4.1.1 of [TR-03111] in the range $[1, n-1]$, where n is the order of the base point of the selected domain parameters using a SRNG defined above.</p> <p>The public key in ECC shall be derived from the private key. It is calculated as $Q = dG$, where d is the private key and G is the base point on the curve. The key size corresponds to the length of the curve used (e.g., 256 bits for CURVE_BRAINPOOL_P256_R1, 521 bits for CURVE_SEC_P521_R1).</p> <p>The concrete domain parameters depend on the curve chosen, see section 6.7.1.3, ECC Curves.</p>
KEY_RSA_PUBLIC KEY_RSA_PRIVATE	2048, 3072, 4096	<p>RSA key pairs shall be generated following the procedure and adhering to the format described in [PKCS #1], with the prime number generation being done according to [FIPS 186-5] using a SRNG defined above.</p> <p>The CSP shall support the key sizes 2048, 3072 and 4096 bits, with longer keys providing higher security levels. However, generating 4096-bit keys can be time-consuming and unpredictable, and is not recommended for Client Applications. Instead, 4096-bit RSA keys should be imported only. As an alternative, ECC key pairs could be used, as they offer comparable security with shorter key lengths.</p>

6.7.1.6 Key Derivation Algorithms

The KeyModule may support the key derivation algorithms listed in Table 6-63; the exact list shall be subject to *Modularity*. The derivation algorithm can be configured for resources used as source secret in key derivation.

The CSP Admin may configure the keyDerivationAlgorithm using CSPKeyDerivationAlgorithm.

The Client Application may pass the resourceId, which refers to this key derivation configuration, to the KeyModule through the key.derive operation that supports the following parameters:

- **Source Secret:** The secret or password resource that is used as base secret.
- **Input Data:** Optional additional input data to be included in the derivation computation, as specified in Table 6-65 for *Key Derivation Additional Input Data*.
- **Derived Key:** This is the output resource to which the derived value will be assigned. Based on the algorithm used, the resulting derived key may be of type KEY_AES, KEY_HMAC, KEY_ECC_PRIVATE, or KEY_DERIVED_SECRET.

The CSP shall support source secrets being used up to the usage limits listed in Table 6-63. These limits apply per key derivation operation and define when a source secret should no longer be used, so as to avoid security risks such as secret exhaustion or statistical attacks.

The KeyModule shall compute the selected keyDerivationAlgorithm in accordance with:

- Table 6-72 for *Access Restrictions on Key Operations*

The CSP Admin may detect if a key derivation algorithm is supported using the CSPKeySupport structure. Each key derivation algorithm supported shall adhere to the algorithm combinations specified in

- Table 6-64 for compatible *Key Derivation, Resource Type, Hash, Size, and Curve* Combinations

Note: The term secret used in Table 6-63 indicates a CSP key resource of type KEY_MASTER_SECRET, KEY_DERIVED_SECRET, or KEY_SHARED_SECRET.

For further information, see section 3.3.2, Key Derivation, and section 3.7.3, Building Blocks.

Table 6-63: Key Derivation Algorithms

Algorithm	Value (Int)	Value (Hex)	Usage limit per source secret	Description
KDF_AES_CMAC	1	0x01	100,000	Two-step key derivation according to [SP800-56C] section 5 with AES-CMAC randomness extraction as defined in [SP800-38B] to create an AES key value from an input secret.
KDF_ECC	2	0x02	100,000	<p>ECC-based key derivation function according to step 2 in algorithm 2 in section 4.1.1 of [TR-03111] but using the input secret as input value instead of a random number to create a cryptographic key pair suitable for ECC.</p> <p>This algorithm does not process additional input data and supports only the combinations for KDF_ECC specified in Table 6-64, requiring an input secret of at least $k + 64$ bits to ensure a uniformly distributed ECC private key (k being the size of the curve).</p> <p>The operation will fail with ERROR_ILLEGAL_CONFIG [0x3070] if the input secret has an incompatible size and with ERROR_ILLEGAL_USE [0x6072] if additional input data is provided.</p> <p>Note: Use KDF_HKDF in combination with KDF_ECC if additional input data shall be incorporated into ECC key derivation.</p>
KDF_HKDF	3	0x03	1,000,000	HMAC-based Key Derivation Function (HKDF) according to section 2 in [RFC 5869] to create a hash key from an input secret.
KDF_PBKDF2	4	0x04	no limit	<p>Password-based Key Derivation Function (PBKDF) according to section 5.2 PBKDF2 in [PKCS #5] to create a KEY_DERIVED_SECRET from a password.</p> <p>Note: This algorithm has no source password usage limitation, provided that a strong salt is used for each derivation.</p>

6.7.1.7 Key Derivation, Resource Type, Hash, Size, and Curve Combinations

The KeyModule shall support the compatible combinations of *Key Derivation Algorithms* listed in Table 6-64 for all *Resource Types*, *Key Types*, *Password Types*, *Message Digest Algorithms*, *Key Sizes*, and *ECC Curves* supported by the platform. Unsupported or illegal combinations shall be rejected and handled according to the *errorMode* configured.

For further information, see section 3.3.2, Key Derivation.

Table 6-64: Key Derivation, Resource Type, Hash, Size, and Curve Combinations

Source Resource	Algorithms	Derived Resource	Hash	Source Size	Derived Size or Curve
KEY_MASTER_SECRET KEY_DERIVED_SECRET KEY_SHARED_SECRET	KDF_AES_CMAC	KEY_AES	ALG_NULL	min 128 bit	128 bit
				min 256 bit	256 bit
KEY_MASTER_SECRET KEY_DERIVED_SECRET	KDF_ECC ²⁴	KEY_ECC_PRIVATE	ALG_NULL	min 384 bit	CURVE_SEC_P256_R1 CURVE_BRAINPOOL_P256_R1
				min 512 bit	CURVE_SEC_P384_R1 CURVE_BRAINPOOL_P384_R1
				512 + 64 bit	CURVE_BRAINPOOL_P512_R1 CURVE_SEC_P521_R1
KEY_MASTER_SECRET KEY_DERIVED_SECRET KEY_SHARED_SECRET	KDF_HKDF	KEY_AES KEY_HMAC KEY_DERIVED_SECRET	ALG_SHA_256 ALG_SHA3_256	min 256 bit	256 bit
			ALG_SHA_384 ALG_SHA3_384	min 384 bit	384 bit
			ALG_SHA_512 ALG_SHA3_512	min 512 bit	512 bit
		KEY_DERIVED_SECRET	ALG_SHA_512 ALG_SHA3_512	512 + 64 bit	512 + 64 bit

²⁴ To ensure uniform distribution when deriving ECC keys with KDF_ECC, the input secret should be longer than the bit length of the curve order *r*. This mitigates distribution bias when mapping to the private key domain.

Source Resource	Algorithms	Derived Resource	Hash	Source Size	Derived Size or Curve
PWD_STRONG	KDF_PBKDF2 ²⁵	KEY_DERIVED_SECRET	ALG_SHA_256	min 26 characters	128 bit
			ALG_SHA_256 ALG_SHA_384	min 52 characters	256 bit
			ALG_SHA_384 ALG_SHA_512	min 78 characters	384 bit
			ALG_SHA_512	min 104 characters	512 bit
			ALG_SHA_512	min 117 characters	512 + 64 bit

6.7.1.8 Key Derivation Additional Input Data

The KeyModule shall support additional input data, such as salt and context information, for key derivation algorithms as specified in Table 6-65. Client Applications shall provide and manage additional input data as specified in the table.

For further information, see section 3.3.2, Key Derivation.

Table 6-65: Key Derivation Additional Input Data

Algorithm	Source Resource	Derived Resource	Additional Input Data
KDF_AES_CMAC	KEY_MASTER_SECRET KEY_DERIVED_SECRET KEY_SHARED_SECRET	KEY_AES	Optional, used as application context information for context binding. Should be unique per derivation use case.
KDF_ECC	KEY_MASTER_SECRET KEY_DERIVED_SECRET KEY_SHARED_SECRET	KEY_ECC_PRIVATE	Not supported. Fails with ERROR_ILLEGAL_USE [0x6072] if provided.
KDF_HKDF	KEY_MASTER_SECRET KEY_DERIVED_SECRET KEY_SHARED_SECRET	KEY_AES KEY_HMAC KEY_DERIVED_SECRET	Supports an optional salt and optional application context information as specified in [RFC 5869]. Salt should be at least the hash output size (e.g., 256 bits for SHA-256). The application context information should be unique per derivation use case.
KDF_PBKDF2	PWD_STRONG	KEY_DERIVED_SECRET	Supports an optional salt that is strongly recommended to prevent rainbow table attacks. Salt should be at least 64 bit; 128 bit or more is strongly recommended. Application context information is not supported.

²⁵ The minimum character length specified for KDF_PBKDF2 is a recommendation and is not enforced by the CSP. However, passwords used for password-based key derivation should ensure a suitable level of entropy for the key to be derived. Such cryptographic recommendations are out of scope of this specification.

6.7.1.9 Key Agreement Schemes

The KeyModule may support the key agreement schemes (KAS) listed in Table 6-66; the exact list shall be subject to *Modularity*. The key agreement scheme can be configured for private key resources used in key agreement processes.

The CSP Admin may configure the keyAgreementScheme using the CSPKeyAgreementScheme.

The Client Application may pass the resourceId, which refers to this key agreement configuration, to the KeyModule through the key.computeSharedSecret operation that supports the following parameters:

- Input Private Key: The private KEY_ECC_PRIVATE key resource typically used by the initiator in the key exchange process, but this can vary depending on the specific algorithm used.
- Input Public Key: The public KEY_ECC_PUBLIC key resource of the opposite party. This key is combined with the private key to compute the shared secret.
- Shared Secret: This is the output resource to which the shared secret value will be assigned. It is a resource of type KEY_SHARED_SECRET that can be further processed, for instance, as source secret for key derivation.

The KeyModule shall only compute the selected keyAgreementScheme in accordance with:

- Table 6-72 for *Access Restrictions on Key Operations*

The CSP Admin may detect if a key agreement scheme is supported using the CSPKeySupport structure. Each key agreement scheme supported shall adhere to the algorithm combinations specified in

- Table 6-67 for compatible *Key Agreement, Key Type, Curve, and Size Combinations*

For further information, see section 3.3.3, Key Agreement, and section 3.7.3, Building Blocks.

Table 6-66: Key Agreement Schemes

Algorithm	Value (Int)	Value (Hex)	Description
KAS_ECKA_DH	1	0x01	The Elliptic Curve Key Agreement based on Diffie-Hellman (ECKA-DH) according to section 4.3.2.1 in [TR-03111] utilizes curves specified in Table 6-60 <i>ECC Curves</i> to create a shared secret.
KAS_ECKA_EG	2	0x02	The Elliptic Curve Key Agreement based on the ElGamal scheme (ECKA-EG), according to section 4.3.2.2 in [TR-03111] utilizing curves specified in Table 6-60 <i>ECC Curves</i> to create a shared secret.

6.7.1.10 Key Agreement, Key Type, Curve, and Size Combinations

The KeyModule shall support the compatible combinations of *Key Agreement Schemes* listed in Table 6-67 for all *Resource Types*, *Key Types*, *Key Sizes*, and *ECC Curves* supported by the platform. Unsupported or illegal combinations shall be rejected and handled according to the errorMode configured.

For further information, see section 3.3.3, Key Agreement.

Table 6-67: Key Agreement, Key Type, Curve, and Size Combinations

Algorithms	Private Input Resource	Public Input Resource	Output Resource	Source Curve	Output Size (bit)
KAS_ECKA_DH, KAS_ECKA_EG	KEY_ECC_PRIVATE	KEY_ECC_PUBLIC	KEY_SHARED_SECRET	CURVE_SEC_P256_R1 CURVE_BRAINPOOL_P256_R1 CURVE_X25519	256

Algorithms	Private Input Resource	Public Input Resource	Output Resource	Source Curve	Output Size (bit)
				CURVE_SEC_P384_R1	384
				CURVE_BRAINPOOL_P384_R1	
				CURVE_BRAINPOOL_P512_R1	512
				CURVE_SEC_P521_R1	

6.7.1.11 Key Events

The AuditModule may log the key-management-related *Audit Events* listed in Table 6-68; the exact list shall be subject to *Modularity*. The event-specific data listed in the table shall be stored according to the configured auditMode.

The CSP Admin may configure the resourceEvents to be logged using CSPResourceEvent.

The CSP Admin may detect if an event type is supported using the CSPAuditSupport structure.

For further information, see section 3.11.4, Selecting Audit Events, and section 6.12.1.2, Audit Events.

Table 6-68: Key Events for Auditing

Event Type	Value (Hex)	Value (Int)	Description	Event Category
EVENT_KEY_GENERATED	0x1070	4208	Key successfully generated. Logged after successful key generation using random data. Note: Key derivation is not included in this event. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Operations: key.generate Event Data: CSPEventDataResource 	Resource Event
EVENT_KEY_DERIVED	0x1071	4209	Key derived successfully. Logged after successful key derivation. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Operations: key.derive Event Data: CSPEventDataKeyDerivation 	Resource Event
EVENT_KEY_SHARED_SECRET_COMPUTED	0x1072	4210	Successful key agreement. This event is logged after the computation of a shared secret through a successful key agreement process. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Operations: key.computeSharedSecret Event Data: CSPEventDataKeyAgreement 	Resource Event
EVENT_PUBLIC_KEY_IMPORT	0x1073	4211	Successfully set a new public key value. This event is logged after successful setting a public key value. Note: Setting a key value via the administrative CSPSetValue command is not included in this event. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Operations: key.manage (IMPORT) Event Data: CSPEventDataResource 	Resource Event

6.7.1.12 Key Error Codes

If `ERROR_MODE_DETAILED` is supported and activated, the `KeyModule` shall use the error codes listed in Table 6-69 as reasons for exceptions.

For further information, see section 5.1.1.6, Error Types.

Table 6-69: Key Error Reason Codes

Reason	Description
0x2070	Illegal resource type: Only key resources can be used for key operations.
0x2071	Illegal key type: The resource provided is not a symmetric or private key.
0x2072	Illegal key type: The resource provided is not a private key.
0x2073	Illegal key type: The resource provided is not a public key.
0x2074	Unknown key mode: The key management mode provided is not defined.
0x3070	Inconsistent config: Key type, hash, size, or curve incompatible with the key derivation algorithm.
0x3071	Inconsistent config: Key type, size or curve incompatible with the key agreement scheme.
0x4070	Key service not initialized: The key management has not been successfully initialized through the <code>key.initManage</code> method.
0x5070	Wrong usage: The source key used for key management operations is not configured for <code>USAGE_KEY</code> .
0x6070	Illegal public key import: The public key value provided is not compatible to the resource it shall be imported to.
0x6071	Export buffer too small: Buffer length is insufficient to export the key.
0x6072	Illegal input data: Additional input data is not supported by the algorithm.
0x8070	Unsupported: The key module is not supported.
0x8071	Unsupported: The key type is not supported.
0x8072	Unsupported: The key size is not supported.
0x8073	Unsupported: The curve is not supported.
0x8074	Unsupported: The key derivation algorithm is not supported.
0x8075	Unsupported: The key agreement scheme is not supported.

6.7.2 Key Configuration

The `KeyModule` shall support the configuration parameters defined in Table 6-70.

Table 6-70: Key Configuration Parameters

Parameter	Description	Parameter Type
keyType	<p>Type of the key, such as <code>KEY_AES</code> and <code>KEY_RSA_PRIVATE</code>, selected from available <i>Key Types</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Data type: <code>CSPKeyType</code> Administration: <code>CSPKey</code> Operations: <code>key.getType</code> 	Key Parameter

Parameter	Description	Parameter Type
keyMode	Operation mode of the KeyModule, either public key import or export mode, selected from the available <i>Key Management Modes</i> . Technical Details: <ul style="list-style-type: none"> Data type: byte Operations: key.initManage 	Service Parameter
keySize	Size of the key in number of bits, selected from available <i>Key Sizes</i> . Technical Details: <ul style="list-style-type: none"> Data type: CSPKeySize Administration: CSPKey 	Key Parameter
curve	ECC curve, e.g., from Brainpool or NIST, selected from available <i>ECC Curves</i> . Only required for keys of type KEY_ECC_PRIVATE and KEY_ECC_PUBLIC. Technical Details: <ul style="list-style-type: none"> Data type: CSPCurve Administration: CSPKey 	Key Parameter
transient	Represents a flag indicating <i>Transient Keys</i> . The value of the key is temporarily stored in volatile memory as transient array. Transient key resources will be reset to STATE_UNINITIALIZED upon invocation of either resource.clear or resource.clearTransient methods. Technical Details: <ul style="list-style-type: none"> Data type: BOOLEAN Initial value: false Administration: CSPKey Operations: key.isTransient, resource.clearTransient 	Key Parameter
keyDerivationAlgorithm	Key derivation algorithm configured for a resource with USAGE_KEY, selected from the available <i>Key Derivation Algorithms</i> . Technical Details: <ul style="list-style-type: none"> Data type: CSPKeyDerivationAlgorithm Administration: CSPKeyDerivationAlgorithms 	Key Parameter
keyDerivationHashAlgorithm	The hash algorithm used for KDF_HKDF, KDF_PBKDF2 key derivation algorithms, selected from the available <i>Message Digest Algorithms</i> . Technical Details: <ul style="list-style-type: none"> Data type: CSPMessageDigestAlgorithm Administration: CSPKeyDerivationAlgorithms 	Key Parameter
keyAgreementScheme	Key agreement scheme configured for a resource with USAGE_KEY, selected from the available <i>Key Agreement Schemes</i> . Technical Details: <ul style="list-style-type: none"> Data type: CSPKeyAgreementScheme Administration: CSPAlgorithms 	Key Parameter

6.7.3 Key Operations

The KeyModule shall implement the operations listed in Table 6-71.

For each operation, the CSP shall handle supported *Counter Types*, *Timer Types* and *Audit Events*.

The operations shall solely utilize resources configured with USAGE_KEY as source keys for key derivation and key agreement and shall comply with the *Access Restrictions on Key Operations* specified in Table 6-72.

Copyright © 2023-2025 GlobalPlatform, Inc. All Rights Reserved.

This document (and the information herein) is subject to updates, revisions, and extensions by GlobalPlatform, and may be disseminated without restriction. Use of the information herein (whether or not obtained directly from GlobalPlatform) is subject to the terms of the corresponding GlobalPlatform license agreement on the GlobalPlatform website (the "License"). Any use (including but not limited to sublicensing) inconsistent with the License is strictly prohibited.

For further information, see section 3.3, Key Management.

Table 6-71: Key Operations

Operation	Details
key.initManage	<p>Initializes this service for <i>Import and Export Public Keys</i> by setting the keyMode from the available <i>Key Management</i> Modes and setting the public key resource to im- or export.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.KeyService.initManage(..) <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_CSP_ERROR
key.manage	<p>Depending on the selected keyMode, either writes the public key in plain (unencrypted) format to the provided output buffer or imports the provided data and stores it within the public key resource.</p> <p>For both, import and export, the CSP shall use the data formats defined in Table 6-58 <i>Key Types</i>.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.KeyService.manage(..) <p>CSP Protocol:</p> <ul style="list-style-type: none"> CSPSetValue <p>Reset Timers (IMPORT only):</p> <ul style="list-style-type: none"> TIMER_VALIDITY_PERIOD <p>Reset Counters (IMPORT only):</p> <ul style="list-style-type: none"> COUNT_USAGE, COUNT_USAGE_PER_BLOCK COUNT_USAGE_SUCCESS_ONLY, COUNT_USAGE_FAILURE_ONLY <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_PUBLIC_KEY_IMPORTED, EVENT_CSP_ERROR
key.updateManage	<p>Optional method to handle multi-part public key import or export by processing a chunk of data. The method can be invoked multiple times for sequential data processing and must be concluded with key.manage.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.KeyService.updateManage(..) <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_CSP_ERROR
key.getManagedLength	<p>Retrieve the size, in bytes, of the buffer required for importing or exporting a public key.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.KeyService.getManagedLength(..)

Operation	Details
key.generate	<p>Generates a cryptographic key value.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.KeyService.generate(..) <p>CSP Protocol:</p> <ul style="list-style-type: none"> CSPGenerateKey <p>Reset Timers:</p> <ul style="list-style-type: none"> TIMER_VALIDITY_PERIOD <p>Reset Counters:</p> <ul style="list-style-type: none"> COUNT_USAGE, COUNT_USAGE_PER_BLOCK COUNT_USAGE_SUCCESS_ONLY, COUNT_USAGE_FAILURE_ONLY <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_KEY_GENERATED, EVENT_CSP_ERROR
key.generateKeyPair	<p>Generates cryptographic key values for public-private key pairs.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.KeyService.generateKeyPair(..) <p>Reset Timers:</p> <ul style="list-style-type: none"> TIMER_VALIDITY_PERIOD <p>Reset Counters:</p> <ul style="list-style-type: none"> COUNT_USAGE, COUNT_USAGE_PER_BLOCK COUNT_USAGE_SUCCESS_ONLY, COUNT_USAGE_FAILURE_ONLY <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_CSP_ERROR
key.computePublicKey	<p>Calculates the cryptographic value of a public key from a given initialized private key.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.KeyService.computePublicKey(..) <p>CSP Protocol:</p> <ul style="list-style-type: none"> CSPComputePublicKey <p>Reset Counters (for the public key):</p> <ul style="list-style-type: none"> COUNT_USAGE, COUNT_USAGE_PER_BLOCK COUNT_USAGE_SUCCESS_ONLY, COUNT_USAGE_FAILURE_ONLY <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_CSP_ERROR

Operation	Details
key.derive	<p>Derives a new key value from source resource (e.g., shared secret, password). The algorithm used for <i>Key Derivation</i> must be configured to the source resource.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.KeyService.derive(..) <p>CSP Protocol:</p> <ul style="list-style-type: none"> CSPDeriveKey <p>Evaluate Timers (for the source key):</p> <ul style="list-style-type: none"> TIMER_VALIDITY_DATE TIMER_VALIDITY_PERIOD <p>Reset Timers (for the destination key):</p> <ul style="list-style-type: none"> TIMER_VALIDITY_PERIOD <p>Reset Counters (for the destination key):</p> <ul style="list-style-type: none"> COUNT_USAGE, COUNT_USAGE_PER_BLOCK COUNT_USAGE_SUCCESS_ONLY, COUNT_USAGE_FAILURE_ONLY <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_KEY_DERIVED, EVENT_CSP_ERROR
key.computeSharedSecret	<p>Performs a <i>Key Agreement</i>. The algorithm used must be configured to the source resources.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.KeyService.computeSharedSecret(..) <p>Evaluate Timers (for the source keys):</p> <ul style="list-style-type: none"> TIMER_VALIDITY_DATE TIMER_VALIDITY_PERIOD <p>Reset Timers (for the destination key):</p> <ul style="list-style-type: none"> TIMER_VALIDITY_PERIOD <p>Reset Counters (for the destination key):</p> <ul style="list-style-type: none"> COUNT_USAGE, COUNT_USAGE_PER_BLOCK COUNT_USAGE_SUCCESS_ONLY, COUNT_USAGE_FAILURE_ONLY <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_KEY_SHARED_SECRET_COMPUTED, EVENT_CSP_ERROR
key.isTransient	<p>Retrieve whether the key is marked as transient.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.KeyService.computeSharedSecret(..)
key.getType	<p>Retrieve the keyType of the key resource.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.KeyService.getType(..)

6.7.3.1 Access Restrictions on Key Operations

The KeyModule shall enforce the access restrictions for *Key Operations* listed in in Table 6-72 for all *Access Rights*, *Usage Types*, *Resource States*, and *Policy Types* supported by the platform. Unauthorized or illegal operation invocations shall be rejected and handled according to the errorMode configured.

Table 6-72: Key Operations Access Restrictions

Operation	Right Required	Usage Required	State Required	Other Restrictions
key.generate	SETUP	-	UNINITIALIZED	<ul style="list-style-type: none"> Client Authentication required Only symmetric or private keys
key.generateKeyPair (public & private key)	SETUP	-	UNINITIALIZED	<ul style="list-style-type: none"> Client Authentication required Only public-private key pairs POLICY_KEYPAIR
key.computePublicKey (public key)	SETUP	-	UNINITIALIZED	
key.computePublicKey (private key)	USE	-	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required Only public-private key pairs POLICY_KEYPAIR
key.derive (derived key)	SETUP	-	UNINITIALIZED	<ul style="list-style-type: none"> Client Authentication required Only consistent Key Derivation, Resource Type, Hash, Size, and Curve Combinations.
key.derive (source)	USE	KEYMGT	OPERATIONAL	
key.computeSharedSecret (destination shared secret)	SETUP	-	UNINITIALIZED	<ul style="list-style-type: none"> Client Authentication required Only consistent Key Agreement, Key Type, Curve, and Size Combinations.
key.computeSharedSecret (private key)	USE	KEYMGT	OPERATIONAL	
key.computeSharedSecret (public key)	USE	-	OPERATIONAL	
key.initManage (EXPORT)	USE	-	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required
key.initManage (IMPORT)	SETUP	-	UNINITIALIZED	<ul style="list-style-type: none"> Client Authentication required POLICY_SECCHANNEL_ESTABLISHED POLICY_PASSWORD
key.manage (EXPORT) key.updateManage (EXPORT)	-	-	OPERATIONAL	
key.manage (IMPORT) key.updateManage (IMPORT)	-	-	UNINITIALIZED	
key.getManagedLength	-	-	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required

6.7.3.2 Sensitive Results Computed by Key Operations

The KeyModule shall temporarily store the results of methods listed in Table 6-73 in transient memory and shall implement the assertSensitiveResult operation, enabling Client Applications to verify that these results were computed without any abnormality.

Table 6-73: Key Operations Requiring Sensitive Results Checks

Operation	Mode	Result type	Result Description
cert.manage cert.updateManage	MANAGE_MODE_PUBLIC_KEY_EXPORT	short	Length of the exported public key.

6.7.3.3 Sensitive Arrays Required for Key Operations

The KeyModule shall invoke `javacard.framework.SensitiveArrays.assertIntegrity(..)` ([JCAPI]) on the parameters listed in Table 6-74.

For further information, see section 5.1.3.4, Sensitive Arrays, and section 8.3, Byte Buffer Parameters.

Table 6-74: Key Operations Requiring Sensitive Arrays

Operation	Mode	Parameter	Description
key.manage key.updateManage	MANAGE_MODE_PUBLIC_KEY_IMPORT	Input buffer	New public key value.
key.manage key.updateManage	MANAGE_MODE_PUBLIC_KEY_EXPORT	Output buffer	The exported public key value.
key.derive	-	Input buffer	Key derivation data.

6.7.4 Key Lifecycle

Most operations provided by the key service, such as key generation, key derivation, and key agreement, are stateless. Only the import and export of public keys support multiple stateful update invocations for handling multipart data, in the same manner as the offloading service as illustrated in Figure 6-12: *Offloading Lifecycle*. A key service is an instance of the KeyModule.

The key service stores *Sensitive Results Computed by Key Operations*.

The CSP shall trigger lifecycle changes to key resources, as illustrated in Figure 6-7.

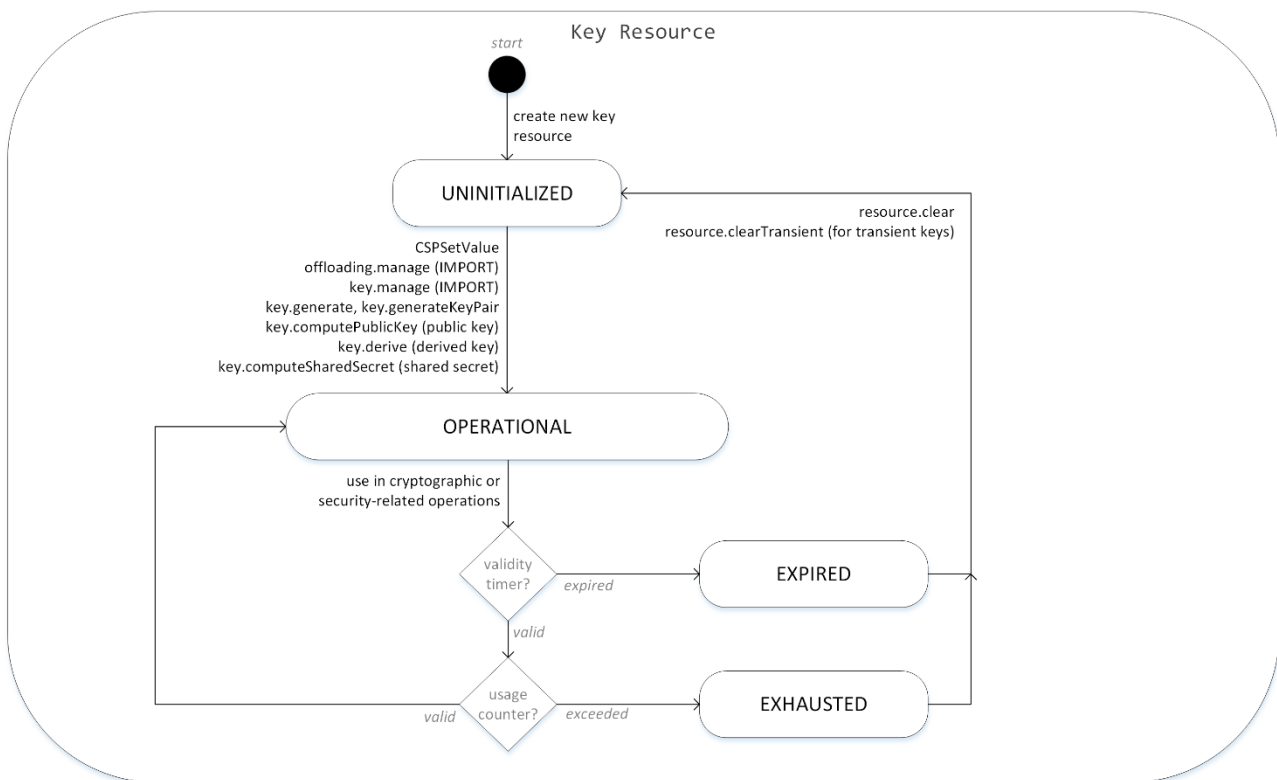
When a key's `usageCounter`, `blockUsageCounter`, `successCounter`, or `failureCounter` reaches its configured limit, the key resource transitions to `STATE_EXHAUSTED`, rendering it unusable for further cryptographic operations. Similarly, when a key's `validityDate` or `validityPeriod` surpasses the current estimated `systemTime`, the key resource changes to `STATE_EXPIRED`, making it unavailable for use.

These state changes of key resources are triggered by the key services and the following services:

- section 6.1.4, *Cipher Lifecycle*
- section 6.2.4, *Signature Lifecycle*
- section 6.3.4, *Transform Lifecycle*
- section 6.6.4, *Attestation Lifecycle*
- section 6.12.4, *Audit Lifecycle*
- section 6.13.4, *Offloading Lifecycle*
- section 6.4.4, *Secure Channel Lifecycle*
- section 6.5.4, *Confidential Data Transfer Lifecycle*

For further information, see section 3.9.1, Overview Timers, and section 3.10.1, Overview Counters.

Figure 6-7: Key Lifecycle



6.7.5 Key Structures

This section lists ASN.1 structures related to the configuration of the key service.

Note: A complete ASN.1 definition is available in [GP CSP ASN1].

6.7.5.1 CSPKeySupport

This data structure represents features, types, and algorithms of the *Key Module*.

It is included in the CSPEnforce command to detect their support by the platform.

ASN 6-22: Key: ASN.1 Definition for CSPKeySupport

```

-- ASN1START
-- Checks for key management support and supported algorithms.
CSPKeySupport ::= SEQUENCE {

    -- Key types, such as AES, RSA or ECC with key sizes.
    keySizes          SET OF CSPKeySize OPTIONAL,

    -- Curves / ECC domain parameter sets.
    curves            SET OF CSPCurve OPTIONAL,

    -- Key derivation algorithms.
    keyDerivationAlgorithms SET OF CSPKeyDerivationAlgorithm OPTIONAL,

```

```
-- Key agreement schemes.
keyAgreementSchemes      SET OF CSPKeyAgreementScheme OPTIONAL
}
-- ASN1STOP
```

6.7.5.2 CSPKey

This data structure represents a resource of type RESOURCE_KEY.

It is part of the CSPResource structure to configure *Key Configuration* parameters.

It can be modified using the CSPConfigureResource command.

For further information, see section 3.3.5, Transient Keys, and section 3.3, Key Management.

ASN 6-23: Key: ASN.1 Definition for CSPKey

```
-- ASN1START
-- Additional attributes required by key resources.
CSPKey ::= SEQUENCE {

    -- The cryptographic keyType.
    type          CSPKeyType,

    -- The cryptographic keySize; also mandatory for ECC keys.
    size          CSPKeySize,

    -- ECC curve parameters; only relevant for ECC keys.
    curve         CSPCurve OPTIONAL,

    -- Flag indicating the key is transient.
    transient     BOOLEAN DEFAULT FALSE
}
-- ASN1STOP
```

6.7.5.3 CSPKeyType

This data structure defines the list of available *Key Types*.

A key type can be selected using the CSPKey structure.

ASN 6-24: Key: ASN.1 Definition for CSPKeyType

```
-- ASN1START
-- Cryptographic key types.
CSPKeyType ::= ENUMERATED {

    -- KEY_AES:
    -- Symmetric key used with Advanced Encryption Standard (AES).
    key-aes          (1),

    -- KEY_HMAC:
```

```

-- Symmetric key used to create HMAC-based signatures.
key-hmac                (2),

-- KEY_ECC_PUBLIC:
-- Public key used for Elliptic Curve Cryptography (ECC).
key-ecc-public          (3),

-- KEY_ECC_PRIVATE:
-- Private key used for Elliptic Curve Cryptography (ECC).
key-ecc-private         (4),

-- KEY_RSA_PUBLIC:
-- Public key for Rivest-Shamir-Adleman (RSA) algorithms.
key-rsa-public          (5),

-- KEY_RSA_PRIVATE:
-- Private key for Rivest-Shamir-Adleman (RSA) algorithms.
key-rsa-private         (6),

-- KEY_MASTER_SECRET:
-- Secret generated within the CSP, used for key derivation.
master-secret           (7),

-- KEY_DERIVED_SECRET:
-- Secret result from key derivation, used for further key derivation.
derived-secret          (8),

-- KEY_SHARED_SECRET:
-- Secret from key agreement, used for key derivation.
key-shared-secret       (9)
}
-- ASN1STOP

```

6.7.5.4 CSPKeySize

This data structure defines the list of available *Key Sizes*.

A key size can be selected using the CSPKey structure.

The key sizes supported can be detected using the CSPKeySupport structure.

For further information, see section 6.7.1.2, Key Sizes.

ASN 6-25: Key: ASN.1 Definition for CSPKeySize

```

-- ASN1START
-- Cryptographic key sizes supported by the CSP.
CSPKeySize ::= ENUMERATED {

    -- KEY_AES_128:

```

```
-- KEY_AES 128 bit.  
key-aes-128                (1),  
  
-- KEY_AES_256:  
-- KEY_AES 256 bit.  
key-aes-256                (3),  
  
-- KEY_AES_2_128:  
-- KEY_AES 2x128 bit for CIPHER_AES_XTS.  
key-aes-2-128              (5),  
  
-- KEY_AES_2_256:  
-- KEY_AES 2x256 bit for CIPHER_AES_XTS.  
key-aes-2-256              (7),  
  
-- KEY_HMAC_256:  
-- KEY_HMAC 256 bit.  
key-hmac-256               (10),  
  
-- KEY_HMAC_384:  
-- KEY_HMAC 384 bit.  
key-hmac-384               (11),  
  
-- KEY_HMAC_512:  
-- KEY_HMAC 512 bit.  
key-hmac-512               (12),  
  
-- KEY_ECC_256:  
-- KEY_ECC_PUBLIC and KEY_ECC_PRIVATE 256 bit.  
key-ecc-256                (16),  
  
-- KEY_ECC_384:  
-- KEY_ECC_PUBLIC and KEY_ECC_PRIVATE 384 bit.  
key-ecc-384                (17),  
  
-- KEY_ECC_512:  
-- KEY_ECC_PUBLIC and KEY_ECC_PRIVATE 512 bit.  
key-ecc-512                (18),  
  
-- KEY_ECC_521:  
-- KEY_ECC_PUBLIC and KEY_ECC_PRIVATE 521 bit.  
key-ecc-521                (19),  
  
-- KEY_RSA_2048:  
-- KEY_RSA_PUBLIC and KEY_RSA_PRIVATE 2048 bit.  
key-rsa-2048               (26),
```

```

-- KEY_RSA_3072:
-- KEY_RSA_PUBLIC and KEY_RSA_PRIVATE 3072 bit.
key-rsa-3072          (27),

-- KEY_RSA_4096:
-- KEY_RSA_* 4096 bit; do not generate, only for import recommended.
key-rsa-4096          (28),

-- KEY_SECRET_128:
-- KEY_MASTER_SECRET, KEY_DERIVED_SECRET, and KEY_SHARED_SECRET 128 bit.
key-secret-128        (32),

-- KEY_SECRET_256:
-- KEY_MASTER_SECRET, KEY_DERIVED_SECRET, and KEY_SHARED_SECRET 256 bit.
key-secret-256        (33),

-- KEY_SECRET_384:
-- KEY_MASTER_SECRET, KEY_DERIVED_SECRET, and KEY_SHARED_SECRET 384 bit.
key-secret-384        (34),

-- KEY_SECRET_512:
-- KEY_MASTER_SECRET, KEY_DERIVED_SECRET and KEY_SHARED_SECRET 512 bit.
key-secret-512        (35),

-- KEY_SECRET_576:
-- KEY_MASTER_SECRET and KEY_DERIVED_SECRET.
key-secret-576        (36)
}
-- ASN1STOP

```

6.7.5.5 CSPCurve

This data structure defines the list of available *ECC Curves*.

A curve can be selected using the CSPKey structure.

The curves supported can be detected using the CSPKeySupport structure.

ASN 6-26: Key: ASN.1 Definition for CSPCurve

```

-- ASN1START
-- Elliptic Curve domain parameter sets for keys of type KEY_ECC_*.
CSPCurve ::= ENUMERATED {

    -- CURVE_BRAINPOOL_P256_R1:
    -- Brainpool P256 r1 [RFC 5639].
    curve-brainpool-p256-r1    (1),

```

```
-- CURVE_BRAINPOOL_P384_R1:
-- Brainpool P384 r1 [RFC 5639].
curve-brainpool-p384-r1      (2),

-- CURVE_BRAINPOOL_P512_R1:
-- Brainpool P512 r1 [RFC 5639].
curve-brainpool-p512-r1      (3),

-- CURVE_SEC_P256_R1:
-- NIST's P-256 curve [SP800-186].
curve-sec-p256-r1            (4),

-- CURVE_SEC_P384_R1:
-- NIST's P-384 curve [SP800-186].
curve-sec-p384-r1            (5),

-- CURVE_SEC_P521_R1:
-- NIST's P-521 curve [SP800-186].
curve-sec-p521-r1            (6),

-- CURVE_X25519:
-- 256-bit, only key agreement [RFC 7748].
curve-x25519                  (7)
}
-- ASN1STOP
```

6.7.5.6 CSPKeySizeOrCurve

This data structure specifies either an ECC curve for ECC keys or a key size for other key types.

It is included in attestation responses such as CSPDataAttestation and CSPKeyPoPAttestation.

For further information, see section 6.7.1.2, Key Sizes, and section 6.7.1.3, ECC Curves.

ASN 6-27: Key: ASN.1 Definition for CSPKeySizeOrCurve

```
-- ASN1START
-- Key parameter size or ECC curve.
CSPKeySizeOrCurve ::= CHOICE {

    -- The keySize in number of bits (not for ECC_*).
    size                CSPKeySize,

    -- ECC curve relevant for keys of type ECC_*.
    curve                CSPCurve
}
-- ASN1STOP
```

6.7.5.7 CSPKeyDerivationAlgorithms

This data structure represents a container for a resource-specific key derivation configuration.

It can be configured for a resource using the CSPAlgorithms structure.

ASN 6-28: Cipher: ASN.1 Definition for CSPKeyDerivationAlgorithms

```
-- ASN1START
-- Key derivation configuration for a specific resource.
CSPKeyDerivationAlgorithms ::= SEQUENCE {

    -- The key derivation algorithm.
    keyDerivationAlgorithm      CSPKeyDerivationAlgorithm,

    -- The hash algorithm used for the key derivation.
    keyDerivationHashAlgorithm  CSPMessageDigestAlgorithm DEFAULT alg-null
}
-- ASN1STOP
```

6.7.5.8 CSPKeyDerivationAlgorithm

This data structure defines the list of available *Key Derivation Algorithms*.

A key derivation algorithm can be selected using the CSPKeyDerivationAlgorithms structure.

The selected key derivation algorithm is utilized by the CSPDeriveKey command.

The key derivation algorithms supported can be detected using the CSPKeySupport structure.

ASN 6-29: Key: ASN.1 Definition for CSPKeyDerivationAlgorithm

```
-- ASN1START
-- List of available key derivation algorithms.
CSPKeyDerivationAlgorithm ::= ENUMERATED {

    -- KDF_AES_CMAC:
    -- Two-step key derivation [SP800-56C].
    kdf-aes-cmac      (1),

    -- KDF_ECC:
    -- ECC private key derivation from a secret [TR-03111].
    kdf-ecc           (2),

    -- KDF_HKDF:
    -- HMAC-based key derivation [RFC 5869].
    kdf-hkdf          (3),

    -- KDF_PBKDF2:
    -- Password-based key derivation [PKCS #5].
    kdf-pbkdf2        (4)
```

```
}
-- ASN1STOP
```

6.7.5.9 CSPKeyAgreementScheme

This data structure defines the list of available *Key Agreement Schemes*.

A key agreement scheme can be selected using the CSPAlgorithms structure.

The key agreement schemes supported can be detected using the CSPKeySupport structure.

ASN 6-30: Key: ASN.1 Definition for CSPKeyAgreementScheme

```
-- ASN1START
-- List of available key agreement schemes.
CSPKeyAgreementScheme ::= ENUMERATED {

    -- KAS_ECKA_DH:
    -- ECC Diffie-Hellman key agreement.
    kas-ecka-dh          (1),

    -- KAS_ECKA_EG:
    -- ECC ElGamal key agreement.
    kas-ecka-eg          (2)
}
-- ASN1STOP
```

6.8 Certificate Module

The CSP may implement the CertificateModule to offer *Certificate Management* services; the choice shall be subject to *Modularity*.

The CSP Admin shall be able to create resources of type RESOURCE_CERTIFICATE with the following configuration:

- certificateType from available *Certificate Types*.

The CSP Admin shall be able to import certificates using the CSPSetValue command.

The Client Application may

- Import or export certificates using the cert.manage operation.
- Extract the public key from certificates using the cert.extractPublicKey operation.

The CSP Admin may detect whether the platform supports this module using the CSPEnforce command.

For further information, see section 3.4, Certificate Management.

6.8.1 Certificate Definitions

6.8.1.1 Certificate Types

The CertificateModule may support the certificate types listed in Table 6-75; the exact list shall be subject to *Modularity*. The certificate type can be configured for resources of type RESOURCE_CERTIFICATE.

The CSP Admin may select the certificateType using the CSPCertificateType structure.

The CSP Admin may detect if a certificate type is supported using the CSPCertificateSupport structure. Each certificate type supported shall adhere to the algorithm combinations specified in:

- Table 6-4 for *Cipher, Padding, and Key Size Combinations*
- Table 6-16 for *Signature, Padding, Hash, Key Size, and Curve Combinations*
- Table 6-31 for *Protocol, Resource, Size, and Curve Combinations*
- Table 6-64 for *Key Derivation, Resource Type, Hash, Size, and Curve Combinations*
- Table 6-67 for *Key Agreement, Key Type, Curve, and Size Combinations*

For further information, see section 3.1.3, Load Resources, section 3.4.1, Import Certificates, and section 3.4.2, Export Certificates.

Table 6-75: Certificate Types

Certificate Type	Value (Hex)	Value (Int)	Description
CERT_CVC	0x01	1	Card Verifiable Certificate (CVC) according to section 2.3 of [TR-03110-3]. CVC contains an ECC or an RSA public key. Note: The CVC is optimized for environments with limited computational resources and storage capacity.
CERT_X509	0x02	2	X.509 version 3 certificate according to [ITU-T X.509]. The X.509 Certificate contains an ECC or an RSA public key, the certificate authority that issued the certificate and associated metadata.

6.8.1.2 Certificate Management Modes

The CertificateModule shall support the certificate management operation modes listed in Table 6-1. This mode defines if the certificate service operates in certificate import or export mode.

The Client Application may select the certificateMode using the cert.initManage operation.

Note: These modes function similarly to *Key Management Modes* and *Offloading Modes*.

Table 6-76: Certificate Management Modes

Mode	Value (Hex)	Value (Int)	Description
MANAGE_MODE_CERTIFICATE_IMPORT	0x03	3	Import certificate. The CertificateModule shall import the certificate provided within the input buffer via cert.manage or cert.updateManage.
MANAGE_MODE_CERTIFICATE_EXPORT	0x04	4	Export certificate. The CertificateModule shall write the certificate to the output buffer provided via cert.manage or cert.updateManage.

6.8.1.3 Certificate Events

The AuditModule may log the certificate-management-related *Audit Events* listed in Table 6-77; the exact list shall be subject to *Modularity*. The event-specific data listed in the table shall be stored according to the configured auditMode.

The CSP Admin may configure the resourceEvents to be logged using CSPResourceEvent.

The CSP Admin may detect if an event type is supported using the CSPAuditSupport structure.

For further information, see section 6.12.1.2, Audit Events.

Table 6-77: Certificate Events for Auditing

Event Type	Value (Hex)	Value (Int)	Description	Event Category
EVENT_CERTIFICATE_IMPORTED	0x1080	4224	Successfully set a new certificate value. This event is logged after successful certificate importation. Note: Importing a certificate via the administrative CSPSetValue command is not included in this event. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Operations: cert.manage (IMPORT) Event Data: CSPEventDataResource 	Resource Event
EVENT_CERTIFICATE_EXPORTED	0x1081	4225	Successfully exported a certificate value. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Operations: cert.manage (EXPORT) Event Data: CSPEventDataResource 	Resource Event

6.8.1.4 Certificate Error Codes

If ERROR_MODE_DETAILED is supported and activated, the CertificateModule shall use the error codes listed in Table 6-78 as reasons for exceptions.

For further information, see section 5.1.1.6, Error Types.

Table 6-78: Certificate Error Reason Codes

Reason	Description
0x2080	Illegal resource type: Only certificate resources can be used for certificate operations.
0x2081	Unknown certificate mode: The certificate management mode provided is not defined.
0x2082	Tag not found: The specified tag could not be found in the certificate structure.
0x4080	Certificate service not initialized: The certificate management has not been successfully initialized through the <code>cert.initManage</code> method.
0x5080	Certificate verification failed: The certificate could not be verified using the provided trust anchor.
0x6080	Illegal certificate import: The data provided is not compatible to the resource it shall be imported to.
0x6082	Illegal public key extraction: The public key within the certificate is not compatible to the resource it shall be imported to.
0x8080	Unsupported: The certificate module is not supported.
0x8081	Unsupported: The certificate type is not supported.

6.8.2 Certificate Configuration

The `CertificateModule` shall support the configuration parameters defined in Table 6-79.

For further information, see section 3.1.4, Configure Resources.

Table 6-79: Certificate Configuration Parameters

Parameter	Description	Parameter Type
<code>certificateType</code>	Type of the certificate, such as <code>CERT_X509</code> or <code>CERT_CVC</code> , selected from available <i>Certificate Types</i> . Technical Details <ul style="list-style-type: none"> Data type: <code>CSPCertificateType</code> Administration: <code>CSPCertificate</code> Operations: <code>cert.getType</code> 	Certificate Parameter
<code>certificateMode</code>	Operation mode of the <code>CertificateModule</code> , either certificate import or export mode, selected from the available <i>Certificate Management Modes</i> . Technical Details: <ul style="list-style-type: none"> Data type: <code>byte</code> Operations: <code>cert.initManage</code> 	Service Parameter

6.8.3 Certificate Operations

The `CertificateModule` shall implement the operations listed in Table 6-80.

For each operation, the CSP shall handle supported *Counter Types*, *Timer Types* and *Audit Events*.

The operations shall comply with the *Access Restrictions on Certificate Operations* specified in Table 6-81.

For further information, see section 3.4, Certificate Management.

Table 6-80: Certificate Operations

Operation	Details
cert.initManage	<p>Initializes this service for either <i>Import Certificates</i> or <i>Export Certificates</i> by setting the certificateMode from the available <i>Certificate Management Modes</i> and setting the certificate resource to im- or export.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.CertificateService.initManage(..) <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_CSP_ERROR
cert.manage	<p>Depending on the selected certificateMode, either writes the certificate in plain (unencrypted) format to the provided output buffer or imports the provided data and stores it within the certificate resource.</p> <p>For both, import and export, the CSP shall use the data formats defined in Table 6-75 <i>Certificate Types</i>.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.CertificateService.manage(..) <p>CSP Protocol:</p> <ul style="list-style-type: none"> CSPSetValue <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_CERTIFICATE_IMPORTED, EVENT_CERTIFICATE_EXPORTED, EVENT_CSP_ERROR
cert.updateManage	<p>Optional method to handle multi-part certificate import or export by processing a chunk of data. The method can be invoked multiple times for sequential data processing and must be concluded with cert.manage.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.CertificateService.updateManage(..) <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_CSP_ERROR
cert.getManagedLength	<p>Retrieve the size, in bytes, of the buffer required for importing or exporting a certificate.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.CertificateService.getManagedLength(..)
cert.extractPublicKey	<p><i>Extract Certificate Data:</i> Extracts the public key from the certificate and uses it to initialize the provided public key resource.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.CertificateService.extractPublicKeyValue(..)
cert.getPublicKeySize	<p><i>Extract Certificate Data:</i> Retrieve the size of the public key included in the certificate.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.CertificateService.getPublicKeySize(..)
cert.getPublicKeyCurve	<p><i>Extract Certificate Data:</i> Retrieve the <i>ECC Curves</i> of the public key included in the certificate.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.CertificateService.getPublicKeyCurve(..)
cert.getValidityDate	<p><i>Extract Certificate Data:</i> Retrieve the validity date included in the certificate.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.CertificateService.getValidityDate(..)

Operation	Details
cert.getType	Retrieve the certificateType of the certificate resource. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.CertificateService.getType(..)
cert.getTagLength	Returns the length of the value corresponding to a given TLV tag in the certificate. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.CertificateService.getTagLength(..)
cert.extractTag	Extracts a TLV-encoded tag value from the given certificate. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.CertificateService.extractTag(..)

6.8.3.1 Access Restrictions on Certificate Operations

The CertificateModule shall enforce the access restrictions for *Certificate Operations* listed in Table 6-81 for all *Access Rights*, *Usage Types*, *Resource States*, and *Policy Types* supported by the platform. Unauthorized or illegal operation invocations shall be rejected and handled according to the errorMode configured.

Table 6-81: Certificate Operations Access Restrictions

Operation	Right Required	Usage Required	State Required	Other restrictions
cert.initManage (EXPORT)	USE	-	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required
cert.initManage (IMPORT)	SETUP	-	UNINITIALIZED	<ul style="list-style-type: none"> Client Authentication required POLICY_SECCHANNEL_ESTABLISHED POLICY_PASSWORD
cert.manage (EXPORT) cert.updateManage (EXPORT)	-	-	OPERATIONAL	
cert.manage (IMPORT) cert.updateManage (IMPORT)	-	-	UNINITIALIZED	
cert.extractPublicKey (certificate)	USE	-	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required
cert.extractPublicKey (public key)	SETUP	-	UNINITIALIZED	
cert.getManagedLength cert.getType cert.extractTag cert.getTagLength cert.getPublicKeyCurve cert.getPublicKeySize cert.getValidityDate	USE	-	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required

6.8.3.2 Sensitive Results Computed by Certificate Operations

The CertificateModule shall temporarily store the results of methods listed in Table 6-82 in transient memory and shall implement the assertSensitiveResult operation, enabling Client Applications to verify that these results were computed without any abnormality.

Table 6-82: Certificate Operations Requiring Sensitive Results Checks

Operation	Mode	Result Type	Result Description
cert.manage cert.updateManage	MANAGE_MODE_CERTIFICATE_IMPORT	short	The length of the exported certificate.

6.8.3.3 Sensitive Arrays Required for Certificate Operations

The CertificateModule shall invoke javacard.framework.SensitiveArrays.assertIntegrity(..) ([JCAPI]) on the parameters listed in Table 6-83.

For further information, see section 5.1.3.4, Sensitive Arrays, and section 8.3, Byte Buffer Parameters.

Table 6-83: Certificate Operations Requiring Sensitive Arrays

Operation	Mode	Parameter	Description
cert.manage cert.updateManage	MANAGE_MODE_CERTIFICATE_IMPORT	Input buffer	New certificate.
cert.manage cert.updateManage	MANAGE_MODE_CERTIFICATE_EXPORT	Output buffer	The received certificate.
cert.getValidityDate	-	Output buffer	The returned validity date.

6.8.4 Certificate Lifecycle

Most of the operations provided by the certificate service, such as retrieving a certificate's validity date or extracting its public key, are stateless. Only the import and export of certificates support multiple stateful update invocations for handling multipart data, in the same manner as the offloading service as illustrated in Figure 6-12: *Offloading Lifecycle*. A certificate service is an instance of the CertificateModule.

The certificate service stores *Sensitive Results Computed by Certificate Operations*.

The CSP shall trigger lifecycle changes to certificate resources, as illustrated in Figure 6-8.

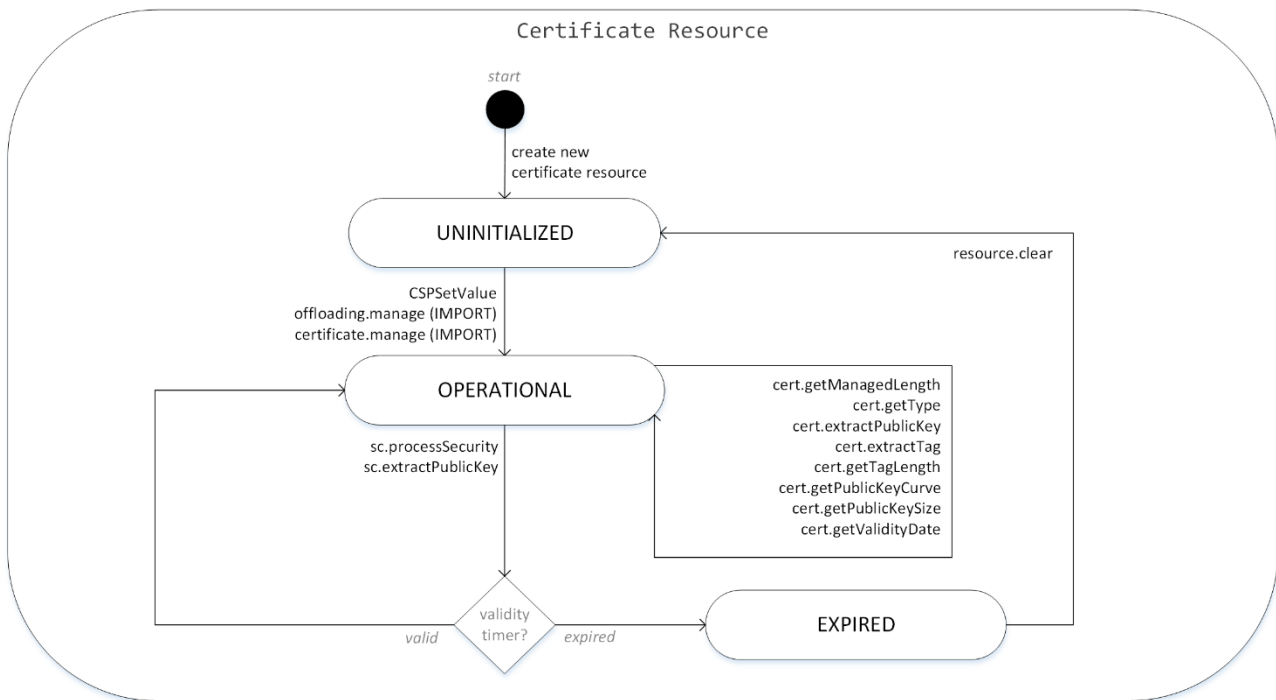
When the *Validity Date of Certificates* surpasses the current estimated systemTime (for certificates with an activated validityCertificate configuration), the certificate resource transitions to STATE_EXPIRED, making its public key unavailable for cryptographic operations.

These state changes of certificate resources are triggered by the certificate services and the following services:

- Section 6.4.4 *Secure Channel Lifecycle*

For further information, see section 3.9.1, Overview Counters, and section 3.10.1, Overview Timers.

Figure 6-8: Certificate Lifecycle



6.8.5 Certificate Structures

This section lists ASN.1 structures related to the configuration of the certificate service.

Note: A complete ASN.1 definition is available in [GP CSP ASN1].

6.8.5.1 CSPCertificateSupport

This data structure represents features, types, and algorithms of the *Certificate Module*.

It is included in the CSPEnforce command to detect their support by the platform.

ASN 6-31: Certificate: ASN.1 Definition for CSPCertificateSupport

```

-- ASN1START
-- Checks for certificate management support and supported certificate types.
CSPCertificateSupport ::= SEQUENCE {

    -- Certificate types, such as CVC or X.509.
    certificateTypes          SET OF CSPCertificateType OPTIONAL
}
-- ASN1STOP
  
```

6.8.5.2 CSPCertificate

This data structure represents a resource of type RESOURCE_CERTIFICATE.

It is part of the CSPResource structure to configure *Certificate Configuration* parameters.

It can be modified using the CSPConfigureResource command.

For further information, see section 3.4, Certificate Management.

ASN 6-32: Certificate: ASN.1 Definition for CSPCertificate

```
-- ASN1START
-- Additional attributes required by certificate resources.
CSPCertificate ::= SEQUENCE {

    -- The certificateType of the certificate.
    type          CSPCertificateType
}
-- ASN1STOP
```

6.8.5.3 CSPCertificateType

This data structure defines the list of available *Certificate Types*.

A certificate type can be selected using the CSPCertificate structure.

The certificate types supported can be detected using the CSPCertificateSupport structure.

ASN 6-33: Certificate: ASN.1 Definition for CSPCertificateType

```
-- ASN1START
-- List of available certificate types.
CSPCertificateType ::= ENUMERATED {

    -- CERT_CVC:
    -- Card Verifiable Certificate (CVC) according to [TR-03110-3].
    cert-cvc          (1),

    -- CERT_X509:
    -- X.509 Certificate according to [ITU-T X.509].
    cert-x509         (2)
}
-- ASN1STOP
```

6.9 Password Module

The CSP may implement the PasswordModule to offer *Password Management*; the choice shall be subject to *Modularity*.

The CSP Admin shall be able to create resources of type RESOURCE_PASSWORD with the following configurations:

- passwordType from available *Password Types*.
- minSize and maxSize for the allowed password character count.

The CSP Admin shall be able to initialize password resources using the *CSPSetValue* command.

The Client Application may trigger password operations by passing the resourceId of the password to the following operations:

- Verify the password via `pwd.check`.
- Change a password via `pwd.update`.
- Unblock a blocked password via `pwd.resetAndUnblock`.

The CSP Admin may detect whether the platform supports this module using the *CSPEnforce* command.

For further information, see section 3.5, Password Management.

6.9.1 Password Definitions

6.9.1.1 Password Types

The PasswordModule may support the password types listed in Table 6-84; the exact list shall be subject to *Modularity*. The password type can be configured for resources of type RESOURCE_PASSWORD.

The CSP Admin may select the passwordType using the *CSPPasswordType* structure.

The CSP Admin may detect if a password type is supported using the *CSPPasswordSupport* structure. Each password type supported shall adhere to the algorithm combinations specified in:

- Table 6-64 for *Key Derivation, Resource Type, Hash, Size, and Curve Combinations*
- Table 6-31 for *Protocol, Resource, Size, and Curve Combinations*

Note: The CSP does not verify the format or employ heuristics to assess password entropy during password creation or updates, as such checks would be too resource intensive.

Table 6-84: Password Types

Password Type	Value (Hex)	Value (Int)	Description
PWD_ANY	0x00	0	No specific rules are applied. The password is represented as a byte array. The underlying format for interpreting the byte values is undefined. This type can be used for custom password formats.
PWD_NUMERIC	0x01	1	The password consists only of numeric characters (0-9) and is encoded in ASCII format. This type is typically used for PINs and PUKs.

Password Type	Value (Hex)	Value (Int)	Description
PWD_ALPHANUMERIC	0x02	2	The password includes both numbers and alphanumeric characters (0-9, a-z, A-Z) and is encoded in ASCII format. This type is an option for basic passwords, extending the use beyond PINs and PUKs.
PWD_UTF8	0x03	3	The password can include a wide range of characters, encompassing international and special characters, encoded in the UTF-8 charset. This type is suitable for more complex passwords that go beyond alphanumeric passwords.
PWD_STRONG	0x04	4	Passwords contain at least one uppercase letter, one lowercase letter, one number, and one special character encoded in the UTF-8 charset based on at least N = 90 different characters. This type is used for password-based key derivation, see section 6.7.1.7, Key Derivation Combinations.

6.9.1.2 Password Events

The AuditModule may log the password-related *Audit Events* listed in Table 6-85; the exact list shall be subject to *Modularity*. The event-specific data listed in the table shall be stored according to the configured auditMode.

The CSP Admin may configure the resourceEvents to be logged using CSPResourceEvent.

The CSP Admin may detect if an event type is supported using the CSPAuditSupport structure.

For further information, see section 3.11.4, Selecting Audit Events, and section 6.12.1.2, Audit Events.

Table 6-85: Password Events for Auditing

Event Type	Value (Hex)	Value (Int)	Description	Event Category
EVENT_PASSWORD_UPDATED	0x1090	4240	Password changed successfully. This event is logged after successful setting an initial password or updating a password. Note: Setting a password via the administrative CSPSetValue command is not included in this event. Technical Details: <ul style="list-style-type: none"> • Message Format: CSPLogMessage • Operations: pwd.update • Event Data: CSPEventDataResource 	Resource Event
EVENT_PASSWORD_UPDATE_FAILED	0x1091	4241	Changing a password failed. This event is logged when setting or updating a password failed (e.g., password not authenticated). Note: The event is not logged if an exception occurs. Technical Details: <ul style="list-style-type: none"> • Message Format: CSPLogMessage • Operations: pwd.update • Event Data: CSPEventDataResource 	Resource Event

Event Type	Value (Hex)	Value (Int)	Description	Event Category
EVENT_PASSWORD_AUTHENTICATED	0x1092	4242	<p>Password verified successfully. This event is logged after successful password verification.</p> <p>Note: Password verifications within secure channel authentication are not included in this event.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Message Format: CSPLogMessage • Operations: pwd.check • Event Data: CSPEventDataResource 	Resource Event
EVENT_PASSWORD_CHECK_FAILED	0x1093	4243	<p>Password mismatch. This event is logged when a password verification has failed.</p> <p>Password verifications within secure channel authentication are not included in this event.</p> <p>Note: The event is not logged if an exception occurs.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Message Format: CSPLogMessage • Operations: pwd.check • Event Data: CSPEventDataPasswordFailure 	Resource Event
EVENT_PASSWORD_BLOCKED	0x1094	4244	<p>Password is blocked due to too many incorrect password attempts. This event is logged when a password changes to STATE_BLOCKED after the tryCounter reaches zero.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Message Format: CSPLogMessage • Operations: pwd.check • Event Data: CSPEventDataResource 	Resource Event
EVENT_PASSWORD_UNBLOCKED	0x1095	4245	<p>A blocked password was unblocked. This event is logged after a password in STATE_BLOCKED is reset and unblocked.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Message Format: CSPLogMessage • Operations: pwd.resetAndUnblock • Event Data: CSPEventDataResource 	Resource Event

6.9.1.3 Password Error Codes

If ERROR_MODE_DETAILED is supported and activated, the PasswordModule shall use the error codes listed in Table 6-86 as reasons for exceptions.

For further information, see section 5.1.1.6, Error Types.

Table 6-86: Password Error Reason Codes

Reason	Description
0x2090	Illegal resource type: Only password resources can be used for password operations.
0x5090	Wrong usage: The password to check is not configured for USAGE_PASSWORD.
0x5091	Password not authenticated: The operation requires the password to be authenticated.
0x5092	Password blocked: A password used is in or has changed to STATE_BLOCKED.

Reason	Description
0x6091	Illegal password: The new password is too short.
0x6092	Illegal password: The new password is too long.
0x8090	Unsupported: The password module is not supported.

6.9.2 Password Configuration

The PasswordModule shall support the configuration parameters defined in Table 6-87 for resources of type RESOURCE_PASSWORD.

For further information, see section 3.5 Password Management.

Table 6-87: Password Configuration Parameters

Parameter	Description	Parameter Type
passwordType	<p>Defines the specific type of the password, such as PWD_NUMERIC, PWD_STRONG, selectable from Password Types.</p> <p>Technical Details</p> <ul style="list-style-type: none"> Data type: CSPPasswordType Administration: CSPPassword Operations: pwd.getType 	Password Parameter
tryLimit	<p>Max number of incorrect password attempts before the password gets blocked.</p> <p>The tryCounter is set to this limit when a new password value is set via</p> <ul style="list-style-type: none"> Load Resource for Personalization Updating Passwords <p>Technical Details</p> <ul style="list-style-type: none"> Data type: byte Initial value: 0xFF (disabled) Administration: CSPPassword Operations: pwd.getTryLimit 	Password Parameter
minSize	<p>Min number of characters for the password.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Data type: byte Initial value: 4 Administration: CSPPassword Operations: pwd.getMinSize 	Password Parameter
maxSize	<p>Max number of characters for the password.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Data type: byte Initial value: 20 Administration: CSPPassword Operations: pwd.getMaxSize 	Password Parameter

Parameter	Description	Parameter Type
inTransport	<p>Represents the inTransport flag to mark <i>Passwords in Transport</i>, indicating an initial password value that requires change by the end-user.</p> <p>The flag can be used together with a transportUsageCounter to realize One-Time Passwords (OTP).</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPBoolean • Initial value: 0x8787 • Administration: CSPSetValue • Operations: pwd.isInTransport 	Password Parameter

6.9.3 Password Operations

The PasswordModule shall implement the operations listed in Table 6-88.

For each operation, the CSP shall handle supported *Counter Types*, *Timer Types* and *Audit Events*.

The operations shall solely be utilizable with resources configured for USAGE_PASSWORD for *Password Verification* and shall comply with the *Access Restrictions on Password Operations* specified in Table 6-89.

Note: Specifically, resources with USAGE_SECCHANNEL, dedicated for use in *Secure Channel Operations*, shall not be permitted for *Password Verification*.

For further information, see section 3.5, Password Management.

Table 6-88: Password Operations

Operation	Details
pwd.check	<p>Verifies a password against the value of the provided password resource. Upon successful <i>Password Verification</i>, the internal authenticated flag will be set. This method will fail for passwords in STATE_UNINITIALIZED, STATE_BLOCKED, STATE_EXHAUSTED, or STATE_EXPIRED.</p> <p>CSP API:</p> <ul style="list-style-type: none"> • org.globalplatform.csp.api.PasswordService.check(..) <p>Evaluate Timers:</p> <ul style="list-style-type: none"> • TIMER_VALIDITY_DATE, TIMER_VALIDITY_PERIOD <p>Increment Counters:</p> <ul style="list-style-type: none"> • COUNT_USAGE • COUNT_TRANSPORT_USAGE, COUNT_USAGE_SUCCESS_ONLY (if successful) • COUNT_USAGE_FAILURE_ONLY, tryCounter (failures only) <p>Reset Counters:</p> <ul style="list-style-type: none"> • COUNT_AUTH_USAGE <p>Fire Events:</p> <ul style="list-style-type: none"> • EVENT_PASSWORD_AUTHENTICATED • EVENT_PASSWORD_CHECK_FAILED • EVENT_PASSWORD_BLOCKED

Operation	Details
<code>pwd.isAuthenticated</code>	<p>Returns the authenticated flag of the password. This method will fail for passwords in <code>STATE_UNINITIALIZED</code>, <code>STATE_BLOCKED</code>, <code>STATE_EXHAUSTED</code>, or <code>STATE_EXPIRED</code>.</p> <p>CSP API:</p> <ul style="list-style-type: none"> <code>org.globalplatform.csp.api.PasswordService.isAuthenticated(..)</code> <p>Evaluate Timers:</p> <ul style="list-style-type: none"> <code>TIMER_AUTH_TIMEOUT</code> <p>Increment Counters:</p> <ul style="list-style-type: none"> <code>COUNT_AUTH_USAGE</code>
<code>pwd.isInTransport</code>	<p>Returns the <code>inTransport</code> flag of the password.</p> <p>CSP API:</p> <ul style="list-style-type: none"> <code>org.globalplatform.csp.api.PasswordService.isInTransport(..)</code>
<code>pwd.isBlocked</code>	<p>CSP API:</p> <ul style="list-style-type: none"> <code>org.globalplatform.csp.api.PasswordService.isBlocked(..)</code>
<code>pwd.getTryLimit</code>	<p>Returns the maximum number of failed password verification attempts allowed before the resource is blocked.</p> <p>CSP API:</p> <ul style="list-style-type: none"> <code>org.globalplatform.csp.api.PasswordService.getTryLimit(..)</code>
<code>pwd.getTryCounter</code>	<p>Returns the <code>tryCounter</code> containing the current number of failed password verification attempts.</p> <p>CSP API:</p> <ul style="list-style-type: none"> <code>org.globalplatform.csp.api.PasswordService.getTryCounter(..)</code>
<code>pwd.update</code>	<p><i>Updating Passwords</i> initially sets a password value or updates an existing password value. This method is resetting usage counter, try counter, validity period, state, and flags. This method will fail for passwords in <code>STATE_BLOCKED</code>, <code>STATE_EXHAUSTED</code>, or <code>STATE_EXPIRED</code>. If the password is already in <code>STATE_OPERATIONAL</code>, this method requires the password being authenticated through a former <code>pwd.check</code>.</p> <p>CSP API:</p> <ul style="list-style-type: none"> <code>org.globalplatform.csp.api.PasswordService.update(newPwd..)</code> <p>Evaluate Timers:</p> <ul style="list-style-type: none"> <code>TIMER_AUTH_TIMEOUT</code> <p>Increment Counters:</p> <ul style="list-style-type: none"> <code>COUNT_AUTH_USAGE</code> <p>Reset Timers (if successful):</p> <ul style="list-style-type: none"> <code>TIMER_VALIDITY_PERIOD</code> <p>Reset Counters (if successful):</p> <ul style="list-style-type: none"> <code>COUNT_USAGE</code>, <code>COUNT_USAGE_PER_BLOCK</code> <code>COUNT_USAGE_SUCCESS_ONLY</code>, <code>COUNT_USAGE_FAILURE_ONLY</code> <code>COUNT_TRANSPORT_USAGE</code> <p>Fire Events:</p> <ul style="list-style-type: none"> <code>EVENT_PASSWORD_UPDATED</code>, <code>EVENT_PASSWORD_UPDATE_FAILED</code>

Operation	Details
pwd.resetAndUnblock	<p>Unblock password and reset tryCounter. This method does not reset the usage counter or validity date. This <i>Unlocking Passwords</i> operation may be restricted by <i>Policies</i>.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.PasswordService.resetAndUnblock(..) <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_PASSWORD_UNBLOCKED
pwd.reset	<p>Logout by invalidating the authenticated flag of the password.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.PasswordService.reset(..)
pwd.getMaxSize	<p>Retrieve the maxSize, representing the maximum character size allowed for updating a password value.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.PasswordService.getMaxSize(..)
pwd.getMinSize	<p>Retrieve the minSize, representing the minimum character size allowed for updating a password value.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.PasswordService.getMinSize(..)
pwd.getType	<p>Retrieve the passwordType of the password.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.PasswordService.getType(..)

6.9.3.1 Access Restrictions on Password Operations

The PasswordModule shall enforce the access restrictions for *Password Operations* listed in Table 6-89 for all *Access Rights*, *Usage Types*, *Resource States*, and *Policy Types* supported by the platform. Unauthorized or illegal operation invocations shall be rejected and handled according to the errorMode configured.

Table 6-89: Password Operations Access Restrictions

Operation	Right Required	Usage Required	State Required	Other Restrictions
pwd.check	USE	PASSWORD	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required POLICY_SECCHANNEL_ESTABLISHED POLICY_PASSWORD POLICY_PRE_BLOCKED
pwd.isAuthenticated	USE	PASSWORD	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required POLICY_SECCHANNEL_ESTABLISHED POLICY_PASSWORD
pwd.isInTransport pwd.isBlocked	USE	-	-	<ul style="list-style-type: none"> Client Authentication required
pwd.reset	USE	PASSWORD	-	<ul style="list-style-type: none"> Client Authentication required
pwd.update	USE	-	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required authenticated: true POLICY_SECCHANNEL_ESTABLISHED POLICY_PASSWORD POLICY_TA2_ACCESS_FLAG

Operation	Right Required	Usage Required	State Required	Other Restrictions
pwd.update	SETUP	-	UNINITIALIZED	<ul style="list-style-type: none"> Client Authentication required POLICY_SECCHANNEL_ESTABLISHED POLICY_PASSWORD
pwd.resetAndUnblock	SETUP	-	BLOCKED	<ul style="list-style-type: none"> Client Authentication required POLICY_SECCHANNEL_ESTABLISHED POLICY_PASSWORD POLICY_UNBLOCK_PASSWORD

6.9.3.2 Sensitive Results computed by Password Operations

The PasswordModule shall temporarily store the results of methods listed in Table 6-90 in transient memory and shall implement the `assertSensitiveResult` operation, enabling Client Applications to verify that these results were computed without any abnormality.

Table 6-90: Password Operations Requiring Sensitive Results Checks

Operation	Result Type	Result Description
pwd.check	short (see CSPBoolean)	The password verification result.
pwd.isAuthenticated	short (see CSPBoolean)	The value of the authenticated flag.
pwd.isInTransport	short (see CSPBoolean)	The value of the <code>inTransport</code> flag.
pwd.isBlocked	short (see CSPBoolean)	Returns <code>CSP.RESULT_TRUE</code> or <code>CSP.RESULT_FALSE</code> .

6.9.3.3 Sensitive Arrays Required for Password Operations

The PasswordModule shall invoke `javacard.framework.SensitiveArrays.assertIntegrity(..)` ([JCAPI]) on the parameters listed in Table 6-91.

For further information, see section 5.1.3.4, Sensitive Arrays, and section 8.3, Byte Buffer Parameters.

Table 6-91: Password Operations Requiring Sensitive Arrays

Operation	Parameter	Parameter Description
pwd.check	Input buffer	The password to check.
pwd.update	Input buffer	New password value.
pwd.resetAndUnblock	Input buffer	New password value.

6.9.4 Password Lifecycle

Password management operations of the password service are stateless. A password service is an instance of the PasswordModule.

The password service stores *Sensitive Results computed by Password Operations*.

The CSP shall trigger lifecycle changes to password resources, as illustrated in Figure 6-9.

A password is set to authenticated after successful verification. The authenticated flag can be reset using the `pwd.reset` operation. Additionally, an optional `authUsageCounter` or `authTimeout` may also reset the authenticated flag.

During initialization using `CSPSetValue`, the CSP Admin can mark a password resource as `inTransport`, indicating that the password should be changed before it is used. A `transportUsageCounter` may be configured to enforce the password change.

If a `tryLimit` is configured for a password resource, the CSP decreases the `tryCounter` with each incorrect password attempt. Once the `tryCounter` reaches zero, the password transitions to `STATE_BLOCKED` and must be unblocked using the `pwd.resetAndUnblock` before it can be used again.

When a `usageCounter`, `successCounter`, or `failureCounter` configured for a password resource reaches its configured `counterLimit`, the password resource transitions to `STATE_EXHAUSTED`, rendering it unusable. Similarly, if the `validityDate` or `validityPeriod` of a password resource surpasses the current estimated `systemTime`, the password resource transitions to `STATE_EXPIRED`, making it unavailable for further use.

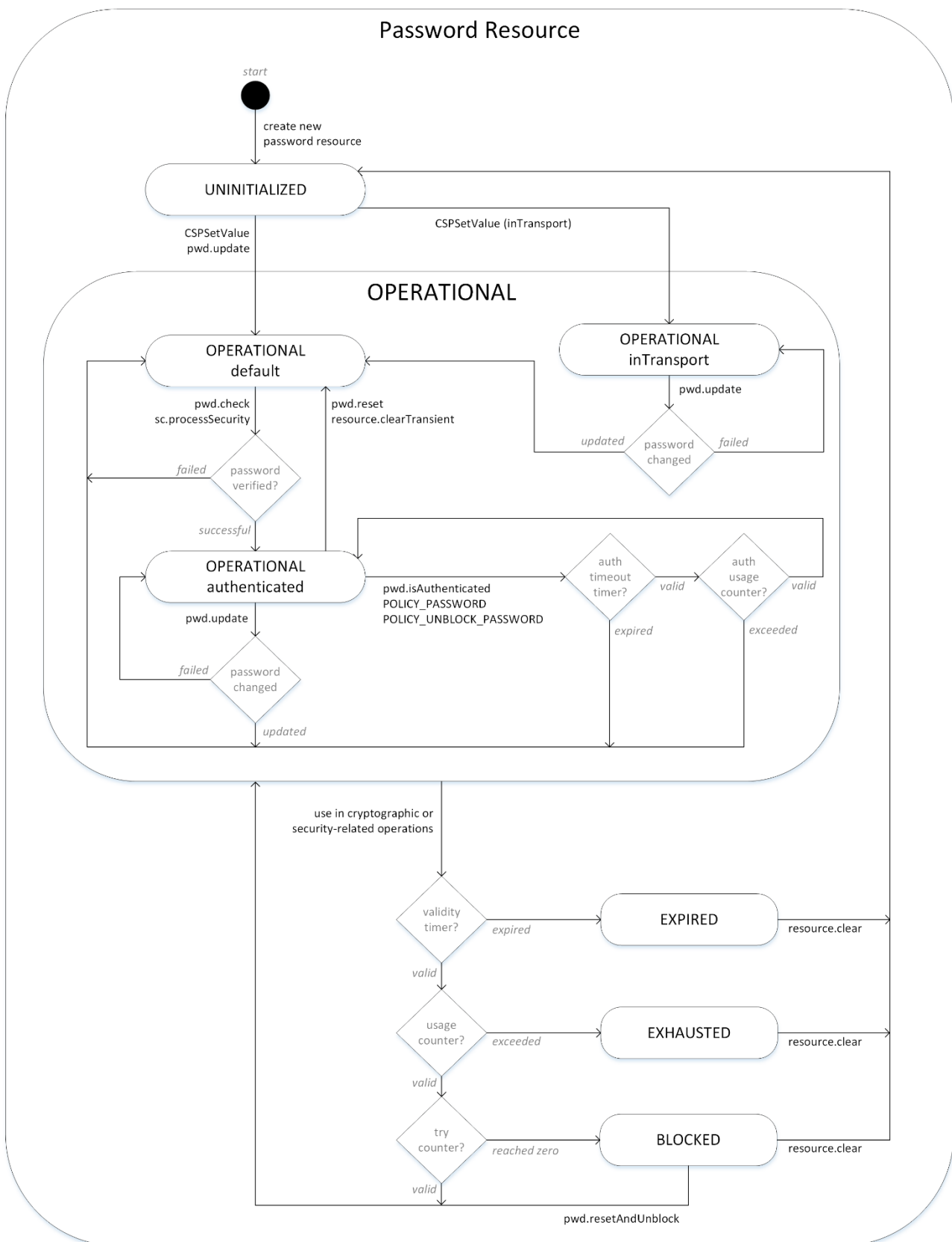
These state changes of password resources are triggered by the password services and the following services:

- Section 6.4.4, *Secure Channel Lifecycle*

Note: If a password update fails (e.g., if the new password does not meet the configured constraints), the CSP does not invalidate the password's authenticated flag. The CSP Admin should define an `authUsageCounter` with a `counterLimit` of 1 if authentication should be invalidated after each usage.

For further information, see section 3.5.1, Password Verification, section 3.9.1, Overview Timers, and section 3.10.1, Overview Counters.

Figure 6-9: Password Lifecycle



6.9.4.1 Internal Password Parameters

The PasswordModule shall maintain the internal parameters listed in Table 6-92.

Table 6-92: Password-internal Parameters

Parameter	Description	Parameter Type
authenticated	<p>Represents the authenticated flag of a password. This transient flag indicates a successful <i>Password Verification</i> and is set upon a successful <code>pwd.check</code> operation. The flag is cleared when the Client Application invokes either <code>pwd.reset</code> or <code>resource.clearTransient</code>, or when an <code>authTimeout</code> expires. It is stored temporarily in transient memory with <code>CLEAR_ON_RESET</code> and may be evaluated through <i>Policies</i> to enforce <i>Access Control</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Initial value: <code>false</code> Operations: <code>pwd.isAuthenticated</code> Events: <code>EVENT_PASSWORD_AUTHENTICATED</code>, <code>EVENT_PASSWORD_CHECK_FAILED</code> Policies: <code>POLICY_PASSWORD</code>, <code>POLICY_UNBLOCK_PASSWORD</code>, <code>POLICY_PRE_BLOCKED</code> Timer: <code>TIMER_AUTH_TIMEOUT</code> Counter: <code>COUNT_AUTH_USAGE</code> 	Password Parameter
tryCounter	<p>The current number of incorrect password attempts. Each unsuccessful authentication attempt reduces the counter by one in the following cases:</p> <ul style="list-style-type: none"> Password Verification (<code>pwd.check</code>) PACE (<code>sc.processSecurity</code>) PACE-CAM (<code>sc.processSecurity</code>) <p>It is reset to the <code>tryLimit</code> when a new password value is set through:</p> <ul style="list-style-type: none"> Load Resource for Personalization Updating Passwords Unblocking Passwords <p>The password transitions to <code>STATE_BLOCKED</code> when the <code>tryCounter</code> reaches zero.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Initial value: <code>tryLimit</code> Operations: <code>pwd.getTryCounter</code> Event: <code>EVENT_PASSWORD_CHECK_FAILED</code> Policies: <code>POLICY_UNBLOCK_PASSWORD</code>, <code>POLICY_PRE_BLOCKED</code> 	Password Parameter

6.9.5 Password Structures

This section lists ASN.1 structures related to the configuration of the password service.

Note: A complete ASN.1 definition is available in [GP CSP ASN1].

6.9.5.1 CSPPasswordSupport

This data structure represents features, types, and algorithms of the *Password Module*.

It is included in the CSPEnforce command to detect their support by the platform.

ASN 6-34: Password: ASN.1 Definition for CSPPasswordSupport

```
-- ASN1START
-- Checks for password support and supported password types.
```

```

CSPPasswordSupport ::= SEQUENCE {

    -- Password types, such as numbers only or strong with special characters.
    passwordTypes      SET OF CSPPasswordType OPTIONAL
}
-- ASN1STOP

```

6.9.5.2 CSPPassword

This data structure represents a resource of type RESOURCE_PASSWORD.

It is part of the CSPResource structure to configure *Password Configuration* parameters.

It can be modified using the CSPConfigureResource command.

For further information, see section 3.5, Password Management.

ASN 6-35: Password: ASN.1 Definition for CSPPassword

```

-- ASN1START
-- Additional attributes required by password resources.
CSPPassword ::= SEQUENCE {

    -- The passwordType parameter of the password.
    type          CSPPasswordType,

    -- Minimum number of password characters allowed.
    minSize       INTEGER (0..255) DEFAULT 4,

    -- Maximum number of password characters allowed.
    maxSize       INTEGER (0..255) DEFAULT 20,

    -- Incorrect password attempts allowed before blocked (disabled if 0xFF).
    tryLimit      INTEGER (0..255) DEFAULT 255
}
-- ASN1STOP

```

6.9.5.3 CSPPasswordType

This data structure defines the list of available *Password Types*.

A password type can be selected using the CSPPassword structure.

The curves supported can be detected using the CSPPasswordSupport structure.

ASN 6-36: Password: ASN.1 Definition for CSPPasswordType

```

-- ASN1START
-- List of available password types.
CSPPasswordType ::= ENUMERATED {

    -- PWD_ANY:

```

```
-- No rules apply on passwords.
pwd-any          (0),

-- PWD_NUMERIC:
-- Only ASCII numbers (e.g., PIN or PUK).
pwd-numeric      (1),

-- PWD_ALPHANUMERIC:
-- ASCII alphanumeric (0-9, a-z, A-Z).
pwd-alphanumeric (2),

-- PWD_UTF8:
-- UTF-8 charset.
pwd-utf8         (3),

-- PWD_STRONG:
-- UTF-8 charset, min. 1 uppercase, 1 lowercase, 1 number, 1 special.
pwd-strong       (4)
}
-- ASN1STOP
```

6.10 Counter Module

The CSP may implement the CounterModule to enforce *Counter and Limits* configurations on resources; the choice shall be subject to *Modularity*.

The CSP Admin shall be able to create resources of type RESOURCE_COUNTER with the following configurations:

- counterType from available *Counter Types*, distinguishing between manual and built-in counters.
- counterCapacity from available *Counter Capacities* to prevent memory wear.
- counterLimit, a threshold at which the counter is considered exceeded.

The CSP shall maintain built-in counters, such as usageCounter, blockUsageCounter, successCounter, failureCounter, transportUsageCounter, and authUsageCounter.

The Client Application may trigger counter operations by passing the resourceId of the counter to:

- Retrieve the current counter value via counter.getValue.
- Increment the counter via counter.increment; only available for counters of type COUNT_MANUAL.

The CSP Admin may detect whether the platform supports this module using the CSPEnforce command.

For further information, see section 3.9, Counter and Limits.

6.10.1 Counter Definitions

6.10.1.1 Counter Modes

The CounterModule may support the counter modes listed in Table 6-93; the exact list shall be subject to *Modularity*. The selected counter mode defines how the CSP shall handle scenarios where configured counter types or capacities are not supported by the platform.

The CSP Admin may select the counterMode using the CSPCounterMode structure.

The CSP Admin may detect if a counter mode is supported using the CSPCounterSupport structure.

For further information, see section 6.14.1.1, Fields.

Table 6-93: Counter Modes

Mode	Value (Hex)	Value (Int)	Description
COUNTER_MODE_OFF	0x00	0	<p>Counter functionality is disabled and not available.</p> <p>Regardless of whether any counter type is configured, the CounterModule shall not increment any counters and shall operate without errors. It shall not check any counter limits and shall use zeros for all counter values processed within the CSP, for example, for logging or retrieving active counter values.</p> <p>This is the standard mode when the platform does not support counters.</p> <p>Note: If the platform does support counters, this mode may be used to temporarily disable counter handling.</p>

Mode	Value (Hex)	Value (Int)	Description
COUNTER_MODE_IGNORE	0x01	1	<p>Ignore counter configurations not supported by the platform.</p> <p>The CounterModule shall handle counters for all counter configurations supported by the platform.</p> <p>For <i>Counter Types</i> not supported, the CounterModule shall ignore the specific counter configuration.</p> <p>For <i>Counter Capacities</i> not supported, the CounterModule shall fall back to the next suitable capacity supported by the platform.</p> <p>For all counter-related configurations ignored, the CounterModule shall continue operation without throwing any errors. It shall not evaluate the affected counterLimit and shall use zeros when using the unsupported counter, e.g., logging or returning data containing the current counterValue.</p>
COUNTER_MODE_STRICT	0x02	2	<p>Stop operation if a counter configuration is not supported by the platform.</p> <p>The CounterModule shall handle all counters configured by the CSP Admin.</p> <p>For unsupported <i>Counter Types</i> and <i>Counter Capacities</i>, the CounterModule shall throw an counterType exception when processing the affected counter.</p>

6.10.1.2 Counter Types

The CounterModule may support the counter types listed in Table 6-94; the exact list shall be subject to *Modularity*.

The counterType is not directly configured; instead, it is selected by configuring either a RESOURCE_COUNTER or one of the built-in counters: usageCounter, blockUsageCounter, successCounter, failureCounter, transportUsageCounter, and authUsageCounter.

The CSP Admin may detect if a CSPCounterType is supported using the CSPCounterSupport structure. Each counter type supported shall adhere to the combinations specified in:

- Table 6-96 for compatible *Counter, Resource Type and Capacity Combinations*

Note: The *Retry Counter with Maximum Try Limit* for passwords is handled within the PasswordModule and is not part of the CounterModule.

For further information, see section 3.9, Counter and Limits.

Table 6-94: Counter Types

Counter Type	Value (Hex)	Value (Int)	Description
COUNT_MANUAL	0x01	1	<p><i>Manual Counter.</i></p> <p>The CSP shall increment the counterValue by the provided delta each time the counter.increment operation is invoked.</p> <p>The CSP shall reset the counter to zero and STATE_OPERATIONAL when the counter.reset operation is invoked, provided the ACCESS_SETUP right is granted.</p> <p>If the optional counterLimit is not zero, the CSP shall compare the new counterValue against this limit. If the value exceeds the limit, the CSP shall set the counter resource to STATE_EXHAUSTED and shall throw an ERROR_NOT_ALLOWED [0x50A0] exception.</p> <p>The CSP shall store each counter update in persistent memory.</p> <p>Note: A <i>Manual Counter</i> is created via RESOURCE_COUNTER.</p>
COUNT_USAGE	0x02	2	<p><i>Usage Counter</i> to count resource usages.</p> <p>The CSP shall increment (by one) the counterValue of a usageCounter configured for resources used in the following methods, regardless of whether they have succeeded or failed:</p> <ul style="list-style-type: none"> • cipher.doFinal, sig.sign • transform.doFinal, sc.processSecurity • sc.confidentialWrap, sc.confidentialUnwrap • att.computeAttestation, pwd.check, offloading.manage <p>The CSP shall reset the counter to zero in the following methods:</p> <ul style="list-style-type: none"> • key.generate, key.generateKeyPair, key.manage (IMPORT) • key.computePublicKey (public key) • key.derive (destination key) • key.computeSharedSecret (destination key) • pwd.update • CSPSetValue • CSPComputePublicKey (only for the public key) • CSPDeriveKey (only for the destination key) <p>If the optional counterLimit is not zero, the CSP shall compare the new counterValue against this limit. If the value exceeds the limit, the CSP shall set the resource that owns the counter to STATE_EXHAUSTED and shall throw an ERROR_NOT_ALLOWED [0x50A0] exception.</p> <p>The CSP shall store each counter update in persistent memory.</p> <p>Note: This counter does not track multi-part processes.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Parameter: usageCounter • Administration: CSPCounters

Counter Type	Value (Hex)	Value (Int)	Description
COUNT_USAGE_PER_BLOCK	0x03	3	<p><i>Usage Counter per Block</i>, counting each cryptographic block.</p> <p>The CSP shall increment the counterValue of the blockUsageCounter by the number of block operations involving the resource within the following method invocations, regardless of whether they have succeeded or failed:</p> <ul style="list-style-type: none"> • cipher.update, cipher.doFinal • sig.update, sig.sign • transform.update, transform.doFinal • sc.processSecurity • sc.updateConfidentialWrap, sc.confidentialWrap • sc.updateConfidentialUnwrap, sc.confidentialUnwrap • att.update, att.computeAttestation • offloading.updateManage, offloading.manage <p>The CSP shall reset the counter to zero in the following methods:</p> <ul style="list-style-type: none"> • key.generate, key.generateKeyPair, key.manage (IMPORT) • key.computePublicKey (public key) • key.derive (destination key) • key.computeSharedSecret (destination key) • pwd.update • CSPSetValue • CSPComputePublicKey (only for the public key) • CSPDeriveKey (only for the destination key) <p>If the optional counterLimit is not zero, the CSP shall compare the new counterValue against this limit. If the value exceeds the limit, the CSP shall set the resource that owns the counter to STATE_EXHAUSTED and shall throw an ERROR_NOT_ALLOWED [0x50A0] exception.</p> <p>The CSP shall store each counter update in persistent memory.</p> <p>Note: This counter tracks individual computations, including multi-part data and block computations (e.g., increments after each cipher block).</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Parameter: blockUsageCounter • Administration: CSPCounters

Counter Type	Value (Hex)	Value (Int)	Description
COUNT_USAGE_SUCCESS_ONLY	0x04	4	<p><i>Success Counter</i>, counting successful completions.</p> <p>The CSP shall increment the counterValue of a successCounter configured for resources used in the following methods, but only when the method completes successfully without errors:</p> <ul style="list-style-type: none"> • cipher.doFinal, sig.sign, sig.verify • transform.doFinal, sc.processSecurity • sc.confidentialWrap, sc.confidentialUnwrap • att.computeAttestation, pwd.check, offloading.manage <p>The CSP shall reset the counter to zero in the following methods:</p> <ul style="list-style-type: none"> • key.generate, key.generateKeyPair, key.manage (IMPORT) • key.computePublicKey (public key) • key.derive (destination key) • key.computeSharedSecret (destination key) • pwd.update • CSPSetValue • CSPComputePublicKey (only for the public key) • CSPDeriveKey (only for the destination key) <p>If the optional counterLimit is not zero, the CSP shall compare the new counterValue against this limit. If the value exceeds the limit, the CSP shall set the resource that owns the counter to STATE_EXHAUSTED and shall throw an ERROR_NOT_ALLOWED [0x50A0] exception.</p> <p>The CSP shall store each counter update in persistent memory.</p> <p>Note: This counter does not track multi-part processes.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Parameter: successCounter • Administration: CSPCounters

Counter Type	Value (Hex)	Value (Int)	Description
COUNT_USAGE_FAILURE_ONLY	0x05	5	<p><i>Failure Counter</i> for keys and passwords.</p> <p>The CSP shall increment the counterValue of a failureCounter configured for resources used in the following methods, but only when the method fails (e.g., password verification failure) or an exception occurs (e.g., provided input buffer is null):</p> <ul style="list-style-type: none"> • cipher.doFinal, sig.sign, sig.verify • transform.doFinal, sc.processSecurity • sc.confidentialWrap, sc.confidentialUnwrap • att.computeAttestation, pwd.check, offloading.manage <p>The CSP shall reset the counter to zero in the following methods:</p> <ul style="list-style-type: none"> • key.generate, key.generateKeyPair, key.manage (IMPORT) • key.computePublicKey (public key) • key.derive (destination key) • key.computeSharedSecret (destination key) • pwd.update • CSPSetValue • CSPComputePublicKey (only for the public key) • CSPDeriveKey (only for the destination key) <p>If the optional counterLimit is not zero, the CSP shall compare the new counterValue against this limit. If the value exceeds the limit, the CSP shall set the resource that owns the counter to STATE_EXHAUSTED and shall throw an ERROR_NOT_ALLOWED [0x50A0] exception.</p> <p>The CSP shall store each counter update in persistent memory.</p> <p>Note: This counter does not track multi-part processes.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Parameter: failureCounter • Administration: CSPCounters
COUNT_AUTH_USAGE	0x06	6	<p><i>Timeout Counter</i> for passwords.</p> <p>The CSP shall increment the counterValue of an authUsageCounter configured for resources used in the following methods and policies:</p> <ul style="list-style-type: none"> • pwd.isAuthenticated • pwd.update • POLICY_PASSWORD • POLICY_UNBLOCK_PASSWORD • POLICY_PRE_BLOCKED <p>The CSP shall reset the counter to zero in the following methods:</p> <ul style="list-style-type: none"> • pwd.check (if successful) <p>The CSP shall compare the current counterValue against the counterLimit. If the value exceeds the limit, the authenticated flag of the password shall be reset to false.</p> <p>The CSP shall store each counter update in transient memory with CLEAR_ON_RESET.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Parameter: authUsageCounter • Administration: CSPCounters

Counter Type	Value (Hex)	Value (Int)	Description
COUNT_TRANSPORT_USAGE	0x07	7	<p><i>Transport Counter (e.g., OTP) for Passwords in Transport.</i></p> <p>The CSP shall increment the counterValue of an transportUsageCounter configured for passwords with activated inTransport flag in the following methods:</p> <ul style="list-style-type: none"> pwd.check (if successful) sc.processSecurity (for successful PACE authentication) <p>The CSP shall reset the counter to zero in the following methods:</p> <ul style="list-style-type: none"> pwd.update (if successful) CSPSetValue <p>The CSP shall set the inTransport to false in the following methods:</p> <ul style="list-style-type: none"> pwd.update (if successful) <p>If the inTransport flag is activated and the counterLimit is not zero, the CSP shall compare the new counterValue against this limit. If the value exceeds the limit, the CSP shall set the resource that owns the counter to STATE_EXHAUSTED and shall throw an ERROR_NOT_ALLOWED [0x50A0].</p> <p>The CSP shall store each counter update in persistent memory.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Parameter: transportUsageCounter Administration: CSPCounters

6.10.1.3 Counter Capacities

The CounterModule may support the counter capacities listed in Table 6-95; the exact list shall be subject to *Modularity*. The counter capability can be configured for resources of type RESOURCE_COUNTER or to built-in counters. Each counter capacity specifies a specific range and a minimal change limit, designed to prevent Flash memory wear. While certain platforms may support a higher number of modifications, each CSP Implementation shall at least support the defined limit. Exceeding this limit shall trigger an ERROR_ILLEGAL_USE [0x60A2] exception.

The CSP Admin may select the counterCapacity using the CSPCounterCapacity structure.

The CSP Admin may detect if a capacity is supported using the CSPCounterSupport structure. Each counter type supported shall adhere to the counter combinations specified in:

- Table 6-96 for compatible *Counter, Resource Type and Capacity Combinations*

For further information, see section 3.9.2, Maximum Counter Increments.

Table 6-95: Counter Capacities

Counter Capacity	Value (Hex)	Value (Int)	Description
COUNTER_TINY	0x01	1	<p>Simple counter for small ranges.</p> <p>Technical Details</p> <ul style="list-style-type: none"> Range: 0-255 (1 byte) Minimum changes supported: 10,000

Counter Capacity	Value (Hex)	Value (Int)	Description
COUNTER_SMALL	0x02	2	Counter used for medium ranges with high precision. Note: This is the default counter capacity if no other is specified. Technical Details <ul style="list-style-type: none"> Range: 0-65,535 (2 byte) Minimum changes supported: 10,000
COUNTER_MEDIUM	0x03	3	Counter used for big ranges with high precision. Note: This counter distributes writes across multiple blocks to prevent memory wear if the Flash memory's supported write cycles are insufficient to handle the specified minimum number of increments. Technical Details <ul style="list-style-type: none"> Range: 0-4,294,967,295 (4 byte) Minimum changes supported: 100,000
COUNTER_LARGE	0x04	4	Counter supporting many increments. Note: This counter uses special memory types or distributes writes across multiple Flash memory blocks to prevent memory wear. Technical Details <ul style="list-style-type: none"> Range: 0-4,294,967,295 (4 byte) Minimum changes supported: 5,000,000

6.10.1.4 Counter, Resource Type and Capacity Combinations

The CounterModule shall support the compatible combinations of *Counter Types* listed in Table 6-96 for all *Resource Types* and *Counter Capacities* supported by the platform. Unsupported or illegal combinations shall be rejected and handled according to the errorMode configured.

Table 6-96: Counter, Resource Type, and Capacity Combinations

Counter Type	Resource Type	Counter Capacities	Description
COUNT_MANUAL	RESOURCE_COUNTER	all	
COUNT_USAGE	RESOURCE_KEY, RESOURCE_PASSWORD	all	
COUNT_USAGE_FAILURE_ONLY		all	
COUNT_USAGE_SUCCESS_ONLY		all	
COUNT_USAGE_PER_BLOCK	RESOURCE_KEY	all	
COUNT_AUTH_USAGE	RESOURCE_PASSWORD	all	
COUNT_TRANSPORT_USAGE	RESOURCE_PASSWORD	all	

6.10.1.5 Counter Events

The AuditModule may log the counter-related *Audit Events* listed in Table 6-97; the exact list shall be subject to *Modularity*. The event-specific data listed in the table shall be stored according to the configured auditMode.

The CSP Admin may configure the resourceEvents to be logged using the CSPResourceEvent.

The CSP Admin may detect if an event type is supported using the CSPAuditSupport structure.

For further information, see section 3.11.4, Selecting Audit Events, and section 6.12.1.2, Audit Events.

Table 6-97: Counter Events for Auditing

Event Type	Value (Hex)	Value (Int)	Description	Event Category
EVENT_COUNTER_EXHAUSTED	0x10A0	4256	Resource counter exhausted. This event is logged when a resource state changes to STATE_EXHAUSTED. Technical Details: <ul style="list-style-type: none"> • Message Format: CSPLogMessage • Operations: all that involve counters • Event Data: CSPEventDataResource 	Resource Event

6.10.1.6 Counter Error Codes

If ERROR_MODE_DETAILED is supported and activated, the CounterModule shall use the error codes listed in Table 6-98 as reasons for exceptions.

For further information, see section 5.1.1.6, Error Types.

Table 6-98: Counter Error Reason Codes

Reason	Description
0x20A0	Illegal counter type: Counter type is not supported by the operation.
0x50A0	Counter exhausted: A resource used is in or has changed to STATE_EXHAUSTED.
0x60A0	Capacity exceeds short: counter.getValueAsShort cannot be used.
0x60A1	No limit configured: counter.getLimit and counter.getRemaining cannot be used.
0x60A2	Counter Capacity Reached: The counter reached its modification limit to prevent memory wear.
0x80A0	Unsupported: The counter module is not supported.
0x80A1	Unsupported: The counter mode is not supported.
0x80A2	Unsupported: The counter type is not supported and COUNTER_MODE_STRICT is set.
0x80A3	Unsupported: The counter capacity is not supported and COUNTER_MODE_STRICT is set.

6.10.2 Counter Configuration

The CounterModule shall support the configuration parameters defined in Table 6-99.

Table 6-99: Counter Configuration Parameters

Parameter	Description	Parameter Type
counterMode	The counter mode defines how the CSP shall handle scenarios where a configured counter is not supported by the platform. It is selected from the available <i>Counter Modes</i> . Technical Details: <ul style="list-style-type: none"> • Data type: CSPCounterMode • Initial value: 0 (COUNTER_MODE_OFF) • Administration: CSPTIMESETTINGS 	CSP Instance Parameter

Parameter	Description	Parameter Type
counterCapacity	<p>Defines the number of increments supported by the counter, ranging from a small capacity of up to 10,000 to a large capacity of up to 5,000,000 increments, selectable from the available <i>Counter Capacities</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPCounterCapacity • Initial value: 2 (COUNTER_SMALL) • Administration: CSPCounter 	Counter Parameter
counterLimit	<p>Maximum limit of a counter. It is used to limit the counterValue.</p> <p>It is evaluated for the following counters:</p> <ul style="list-style-type: none"> • COUNT_MANUAL, COUNT_USAGE, COUNT_USAGE_PER_BLOCK • COUNT_USAGE_SUCCESS_ONLY, COUNT_USAGE_FAILURE_ONLY • COUNT_AUTH_USAGE, COUNT_TRANSPORT_USAGE <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: byte[] (the length varies, see Counter Capacities) • Initial value: 0x0 (no limit set) • Administration: CSPCounter • Operations: counter.getLimit 	Counter Parameter
counterType	<p>Defines how the counter is counted, as specified by the available <i>Counter Types</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPCounterType • Administration: - (indirect, see Counter Types) 	-
usageCounter	<p>A built-in counter that counts resource usages, as specified by COUNT_USAGE. Its counterValue is reset to zero when a new resource value is set via</p> <ul style="list-style-type: none"> • <i>Load Resource for Personalization</i> • <i>Import and Export Public Keys</i> • <i>Updating Passwords</i> <p>Note: It may be utilized as audit log counter for <i>Creating Log Messages</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPCounter • Administration: CSPCounters • Operations: counter.getLimit 	Key or Password Parameter
blockUsageCounter	<p>A built-in counter that counts each block operation using a specific resource, as specified by COUNT_USAGE_PER_BLOCK.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPCounter • Administration: CSPCounters • Operations: counter.getLimit 	Key or Password Parameter
successCounter	<p>A built-in counter that counts only successfully completed operations using a specific resource, as specified by COUNT_USAGE_SUCCESS_ONLY.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPCounter • Administration: CSPCounters • Operations: counter.getLimit 	Key or Password Parameter

Parameter	Description	Parameter Type
failureCounter	A built-in counter that counts failed operations using a specific resource, as specified by COUNT_USAGE_FAILURE_ONLY. Technical Details: <ul style="list-style-type: none"> Data type: CSPCounter Administration: CSPCounters Operations: counter.getLimit 	Key or Password Parameter
authUsageCounter	A built-in counter that functions as a timeout for authenticated passwords, as specified by COUNT_AUTH_USAGE. Technical Details: <ul style="list-style-type: none"> Data type: CSPCounter Administration: CSPCounters Operations: counter.getValueAsShort 	Key or Password Parameter
transportUsageCounter	A built-in counter to limit the number of uses for passwords marked as inTransport, as specified by COUNT_TRANSPORT_USAGE. For example, it may function as One-Time Password (OTP) for <i>Passwords in Transport</i> . Technical Details: <ul style="list-style-type: none"> Data type: CSPCounter Administration: CSPCounters Operations: counter.getValueAsShort 	Password Parameter

6.10.3 Counter Operations

The CounterModule shall implement the operations listed in Table 6-100.

For each operation, the CSP shall handle supported *Audit Events*.

These operations shall comply with the *Access Restrictions on Counter Operations* specified in Table 6-101.

For further information, see section 3.9, Counter and Limits.

Table 6-100: Counter Operations

Operation	Description
counter.hasCounter	Retrieve whether a specific timer type is activated for the resource. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.CounterService.hasCounter(..)
counter.getValue	Retrieve the current counterValue of a counter. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.CounterService.getValue(..)
counter.getValueAsShort	Retrieve the current counterValue as short value. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.CounterService.getValueAsShort(..)
counter.hasLimit	Check if a counterLimit is configured for a counter. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.CounterService.hasLimit(..)
counter.getLimit	Retrieve the maximum counterLimit configured for a counter. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.CounterService.getLimit(..)

Operation	Description
counter.getRemaining	Compute and returns the difference of counterLimit and counterValue. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.CounterService.getRemaining(..)
counter.isExhausted	Check if the current counterValue has exceeded the configured counterLimit. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.CounterService.isExhausted(..)
counter.increment	Increments the counterValue and triggers state change to STATE_EXPIRED if the counterLimit exceeds, as specified by COUNT_MANUAL. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.CounterService.increment(..) Fire Events: <ul style="list-style-type: none"> EVENT_COUNTER_EXHAUSTED
counter.reset	Resets manual counters to zero and state to STATE_OPERATIONAL, as specified by COUNT_MANUAL. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.CounterService.reset(..)

6.10.3.1 Access Restrictions on Counter Operations

The CounterModule shall enforce the access restrictions for *Counter Operations* listed in Table 6-101 for all *Access Rights*, *Usage Types*, *Resource States*, and *Resource Types* supported by the platform. Unauthorized or illegal operation invocations shall be rejected and handled according to the errorMode configured.

Table 6-101: Counter Operations Access Restrictions

Operation	Right Required	Usage Required	State Required	Other Restrictions
counter.increment	USE	-	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required Only COUNT_MANUAL
counter.reset	SETUP	-	-	<ul style="list-style-type: none"> Client Authentication required Only COUNT_MANUAL
counter.hasCounter	USE	-	-	<ul style="list-style-type: none"> Client Authentication required Not COUNT_MANUAL
counter.getValue counter.getValueAsShort counter.getRemaining	USE	-	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required
counter.hasLimit counter.getLimit counter.isExhausted	-	-	-	<ul style="list-style-type: none"> Client Authentication required

6.10.3.2 Sensitive Results computed by Counter Operations

The CounterModule shall temporarily store the results of methods listed in Table 6-102 in transient memory and shall implement the assertSensitiveResult operation, enabling Client Applications to verify that these results were computed without any abnormality.

Table 6-102: Counter Operations Requiring Sensitive Results Checks

Operation	Result Type	Result Description
counter.hasCounter	short (CSPBoolean)	The CSPBoolean returned.
counter.getValue	short	The buffer length returned.
counter.getValueAsShort	short	The counterValue returned.
counter.getRemaining	short	The remaining counterValue returned.
counter.hasLimit	short (CSPBoolean)	The CSPBoolean returned.
counter.isExhausted	short (CSPBoolean)	The CSPBoolean returned.

6.10.3.3 Sensitive Arrays Required for Counter Operations

The CounterModule shall invoke `javacard.framework.SensitiveArrays.assertIntegrity(..)` ([JCAPI]) on the parameters listed in Table 6-103.

For further information, see section 5.1.3.4, Sensitive Arrays, and section 8.3, Byte Buffer Parameters.

Table 6-103: Counter Operations Requiring Sensitive Arrays

Operation	Parameter	Parameter Description
counter.getValue	Output buffer	Retrieved counter value.
counter.getRemaining	Output buffer	Retrieved remaining counter value.
counter.getLimit	Output buffer	Retrieved counter limit.

6.10.4 Counter Lifecycle

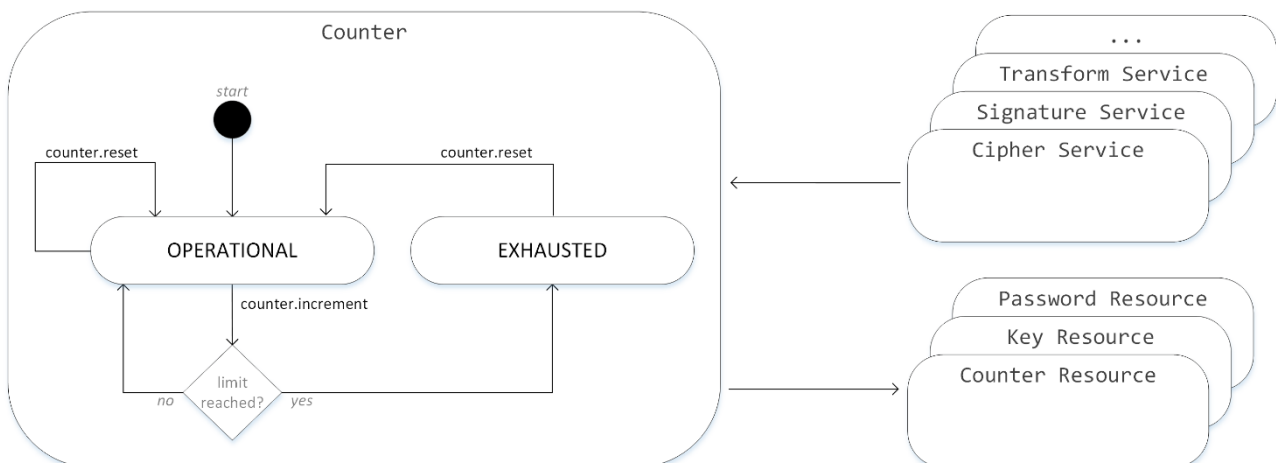
Operations of the counter service are stateless. A counter service is an instance of the CounterModule.

The counter service stores *Sensitive Results computed by Counter Operations*.

The CSP shall trigger lifecycle changes to counters, as illustrated in Figure 6-10. A counter represents either a counter resource or a CSP-internal data structure used to maintain counters for keys and passwords.

A counter resource is incremented manually and transitions to STATE_EXHAUSTED when the counter limit is reached. The CSP-internal usageCounter, blockUsageCounter, successCounter, failureCounter, and transportUsageCounter are incremented by the CSP and transition the resource owning the counter to STATE_EXHAUSTED once any respective limit is reached. The CSP-internal authUsageCounter resets the authenticated flag of the password that owns the counter.

Figure 6-10: Counter Lifecycle



6.10.4.1 Internal Counter Parameters

The CounterModule shall maintain the internal parameters listed in Table 6-104.

Table 6-104: Counter-internal Parameters

Parameter	Description	Parameter Type
counterValue	<p>Number of operations performed. It may be limited by the counterValue.</p> <p>It is reset to zero when a new resource value is set through</p> <ul style="list-style-type: none"> Load Resource for Personalization Import and Export Public Keys (only import) Updating Passwords. <p>It is increment and evaluated for the following counters:</p> <ul style="list-style-type: none"> COUNT_MANUAL, COUNT_USAGE, COUNT_USAGE_PER_BLOCK COUNT_USAGE_SUCCESS_ONLY, COUNT_USAGE_FAILURE_ONLY COUNT_AUTH_USAGE <p>Technical Details:</p> <ul style="list-style-type: none"> Data type: byte[] (the length varies, see <i>Counter Capacities</i>) Initial value: 0x0 Fields: FIELD_USAGE_COUNTER Operations: counter.getValue 	Counter Parameter

6.10.5 Counter Structures

This section lists ASN.1 structures related to the configuration of the counter service.

Note: A complete ASN.1 definition is available in [GP CSP ASN1].

6.10.5.1 CSPCounterSupport

This data structure represents features, types, and algorithms of the *Counter Module*.

It is included in the CSPEnforce command to detect their support by the platform.

ASN 6-37: Counter: ASN.1 Definition for CSPCounterSupport

```
-- ASN1START
-- Checks for counter functionality support and supported counter types.
```

```
CSPCounterSupport ::= SEQUENCE {

    -- Counter operation modes to specify the handling of unsupported counters.
    counterModes          SET OF CSPCounterMode OPTIONAL,

    -- Counter types.
    counterTypes          SET OF CSPCounterType OPTIONAL,

    -- Counter capacities.
    counterCapacities     SET OF CSPCounterCapacity OPTIONAL,

    -- Maximum number of counters with large capacity.
    largeCounterLimit     INTEGER (0..255) OPTIONAL
}
-- ASN1STOP
```

6.10.5.2 CSPCounterSettings

This data structure represents a general Counter Configuration for the CSP Instance.

It can be configured using the CSPSetup structure.

It can be retrieved using the CSPConfiguration structure.

For further information, see section 3.9, Counter and Limits.

ASN 6-38: Counter: ASN.1 Definition for CSPCounterSettings

```
-- ASN1START
-- General counter settings for the CSP Instance.
CSPCounterSettings ::= SEQUENCE {

    -- Select counter mode specify the handling of unavailable counter types.
    counterMode          CSPCounterMode

}
-- ASN1STOP
```

6.10.5.3 CSPCounterMode

This data structure defines the list of available of *Counter Modes*.

A counter mode can be selected using the CSPCounterSettings structure.

The counter modes supported can be detected using the CSPCounterSupport structure.

ASN 6-39: Counter: ASN.1 Definition for CSPCounterMode

```
-- ASN1START
-- List of available counter operation modes.
CSPCounterMode ::= ENUMERATED {

    -- COUNTER_MODE_OFF:
```

```
-- Counter functionality is disabled or not available.
counter-mode-off          (0),

-- COUNTER_MODE_IGNORE:
-- Ignore counters configured if not supported by the platform.
counter-mode-ignore       (1),

-- COUNTER_MODE_STRICT:
-- Stop operation if a configured counter is not supported.
counter-mode-strict       (2)
}
-- ASN1STOP
```

6.10.5.4 CSPCounterType

This data structure defines the list of available of *Counter Types*.

A counter type can be selected using the CSPCounterSettings structure.

The counter types supported can be detected using the CSPCounterSupport structure.

ASN 6-40: Counter: ASN.1 Definition for CSPCounterType

```
-- ASN1START
-- List of available counter types.
CSPCounterType ::= ENUMERATED {

    -- COUNT_MANUAL:
    -- Manual counter invoked by the Client Application.
    count-manual          (1),

    -- COUNT_USAGE_PER_BLOCK:
    -- usage counter counting each computation, including each cipher block.
    count-usage-per-block (2),

    -- COUNT_USAGE:
    -- usage counter counting complete processes (e.g., only doFinal calls).
    count-usage-completions (3),

    -- COUNT_USAGE_SUCCESS_ONLY:
    -- Usage counter counting only successful processes (excluding updates).
    count-usage-success-only (4),

    -- COUNT_USAGE_FAILURE_ONLY:
    -- Usage counter counting only failed processes (excluding updates).
    count-usage-failure-only (5),

    -- COUNT_AUTH_USAGE:
    -- Password authentication timeout realized as usage counter.
```

```

count-auth-usage          (6),

-- COUNT_TRANSPORT_USAGE:
-- Transport counter (e.g., OTP) for passwords in transport.
count-transport-usage     (7)
}
-- ASN1STOP

```

6.10.5.5 CSPCounters

This data structure represents a container for a resource-specific *Counter Configuration*.

It is part of the CSPResource structure.

It can be modified using the CSPConfigureResource command.

The counter types supported can be detected using the CSPCounterSupport structure.

For further information, see section 3.9, Counter and Limits.

ASN 6-41: Counter: ASN.1 Definition for CSPCounters

```

-- ASN1START
-- Container for built-in counters that can be configured to a resource.
CSPCounters ::= SEQUENCE {

    -- Counter of type COUNT_USAGE.
    usageCounter          CSPCounter OPTIONAL,

    -- Counter of type COUNT_USAGE_PER_BLOCK.
    blockUsageCounter     CSPCounter OPTIONAL,

    -- Counter of type COUNT_USAGE_SUCCESS_ONLY.
    successCounter        CSPCounter OPTIONAL,

    -- Counter of type COUNT_USAGE_FAILURE_ONLY.
    failureCounter        CSPCounter OPTIONAL,

    -- Counter of type COUNT_AUTH_USAGE.
    authUsageCounter      CSPCounter OPTIONAL,

    -- Counter of type COUNT_TRANSPORT_USAGE.
    transportUsageCounter CSPCounter OPTIONAL
}
-- ASN1STOP

```

6.10.5.6 CSPCounter

This data structure represents a resource of type RESOURCE_COUNTER or one of the built-in counters.

It is part of the CSPResource, CSPCounters and CSPSecureChannelSettings structure.

It can be modified using the CSPConfigureResource command.

For further information, see section 3.9, Counter and Limits.

ASN 6-42: Counter: ASN.1 Definition for CSPCounter

```
-- ASN1START
-- Data structure for a counter configuration.
CSPCounter ::= SEQUENCE {

    -- The structure of the counter defines the max increments supported.
    counterCapacity      CSPCounterCapacity DEFAULT counter-medium,

    -- Maximum counter limit before the counter exceeds; is 0 when disabled.
    counterLimit         OCTET STRING (SIZE(4)) DEFAULT '00000000'H
}
-- ASN1STOP
```

6.10.5.7 CSPCounterCapacity

This data structure defines the list of available of *Counter Capacities*.

A counter capacity can be selected using the CSPCounterSettings structure.

The counter capacities supported can be detected using the CSPCounterSupport structure.

ASN 6-43: Counter: ASN.1 Definition for CSPCounterCapacity

```
-- ASN1START
-- List of available counter capacity types.
CSPCounterCapacity ::= ENUMERATED {

    -- COUNTER_TINY:
    -- Tiny: 1-byte counter supporting 10,000 increments.
    counter-tiny      (1),

    -- COUNTER_SMALL:
    -- Small: 2-byte counter supporting 10,000 increments.
    counter-small     (2),

    -- COUNTER_MEDIUM:
    -- Medium: 4-byte counter supporting 100,000 increments.
    counter-medium    (3),

    -- COUNTER_LARGE:
    -- Large: 4-byte counter supporting 5,000,000 increments.
    counter-large     (4)
}
-- ASN1STOP
```

6.11 Time Module

The CSP may implement the TimeModule to provide *Timer and Time Management*; the choice shall be subject to *Modularity*.

The CSP Admin shall be able to create resources of type RESOURCE_TIMER with the following configurations:

- timerType from available *Timer Types*, distinguishing between manual and built-in timers.
- timeLimit, the timer configuration at which the timer is considered expired.

The CSP Admin shall be able to set a reference time using the CSPSetTime command.

The CSP shall maintain built-in timers for resources, such as validityPeriod, validityDate, validityCertificate, authTimeout, and securityTimeout.

The Client Application may trigger timer operations by passing the resourceId of the timer to:

- Start a timer via time.reset; only available for manual timers.

The CSP Admin may detect whether the platform supports this module using the CSPEnforce command.

For further information, see section 3.10, Timer and Time Management.

6.11.1 Time Definitions

6.11.1.1 Time Modes

The TimeModule may support the time management modes listed in Table 6-105; the exact list shall be subject to *Modularity*. The selected time mode defines how the CSP shall handle scenarios where time is not available.

The CSP Admin may select the timeMode using the CSPTimeMode structure.

The CSP Admin may detect if a time mode is supported using the CSPTimeSupport structure.

For further information, see section 3.10.3, Time Not Available, and section 6.14.1.1, Fields.

Table 6-105: Time Modes

Time Mode	Value (Hex)	Value (Int)	Description
TIME_MODE_OFF	0x00	0	<p>Time functionality is disabled and not available.</p> <p>The TimeModule shall ignore all time-based configurations and shall operate without errors. It shall not evaluate any timer limits and shall use 0xFF for all time values processed within the CSP, for example, when logging or returning timestamps (0xFFFFFFFF), validity periods (0xFFFF), or timeouts (0xFF).</p> <p>This is the standard mode when time is not supported by the platform.</p> <p>Note: This mode may be used to temporarily disable timer evaluation even if time is supported by the platform.</p>

Time Mode	Value (Hex)	Value (Int)	Description
TIME_MODE_IGNORE	0x01	1	<p>Ignore time-related configurations not supported by the platform.</p> <p>The TimeModule shall manage time and evaluate timers for all time-related configurations supported by the platform.</p> <p>For <i>Timer Types</i> not supported, the TimeModule shall ignore the specific timer configuration.</p> <p>The TimeModule shall use <code>systemTime</code> when available. If <code>systemTime</code> is unavailable for the current session, it shall fall back to <code>timeSinceBoot</code>. If neither is available, the TimeModule shall ignore the time-related configuration.</p> <p>For all configurations ignored, the TimeModule shall continue operation without throwing any errors. It shall not evaluate the affected <code>timeLimit</code> and shall use <code>0xFF</code> for unsupported time values processed within the CSP, for example, when logging or returning timestamps (<code>0xFFFFFFFF</code>), validity periods (<code>0xFFFF</code>) or timeouts (<code>0xFF</code>). If only <code>systemTime</code> is unavailable, it shall use <code>0xFFFF</code> concatenated with <code>timeSinceBoot</code>.</p>
TIME_MODE_STRICT	0x02	2	<p>Stop operation if a timer configuration is not supported by the platform or if it requires a currently unsynchronized <code>systemTime</code>.</p> <p>The TimeModule shall manage time and evaluate all timers as configured by the CSP Admin.</p> <p>For unsupported <i>Timer Types</i>, the TimeModule shall throw an <code>0x80B4</code> exception when processing the affected timer.</p> <p>The TimeModule shall throw <code>ERROR_NOT_SUPPORTED (0x80B1)</code> if <code>timeSinceBoot</code> is utilized but not supported, or if <code>systemTime</code> is utilized but not supported. Additionally, it shall throw <code>ERROR_INVALID_INIT (0x40B0)</code> if <code>systemTime</code> is utilized but no <code>referenceTime</code> is set for the current session.</p>

6.11.1.2 Time Synchronization Methods

The TimeModule may support the time synchronization methods listed in Table 6-107; the exact list shall be subject to *Modularity*. The TimeModule shall perform the time synchronization according to the bit values specified in Table 6-106.

The CSP Admin may configure the `timeSynchronization` using the `CSPTimeSynchronization` structure.

The CSP Admin may detect if a synchronization method is supported using the `CSPTimeSupport` structure.

For further information, see section 3.10.2, Time Synchronization.

Table 6-106: Time Synchronization Methods Bit Positions

b8	b7	b6	b5	b4	b3	b2	b1	Time Synchronization Method
-	-	-	-	X	X	-	1	TIME_SYNC_FROM_TA
-	-	-	-	X	X	1	-	TIME_SYNC_FROM_CLIENT
X	X	X	X	X	X	X	X	RFU
X	X	X	X	X	X	X	X	RFU
-	-	-	1	X	X	-	-	TIME_SYNC_ENFORCE_NEWER
-	-	1	-	X	X	-	-	TIME_SYNC_PERSIST

b8	b7	b6	b5	b4	b3	b2	b1	Time Synchronization Method
-	1	-	-	X	X	-	-	TIME_SYNC_VERIFY_SIG
1	-	-	-	X	X	-	-	TIME_SYNC_VERIFY_SIG_WITH_CHALLENGE

X = Bit is not evaluated. - = Any bit can be used. 0 = Bit cannot be set. 1 = Bit determines the action.

Table 6-107: Time Synchronization Methods

Time Sync Method	Value (Hex)	Value (Int)	Description
TIME_SYNC_FROM_TA	0x01	1	If enabled, the CSP uses the creation dates from <i>TA Certificates as Time Source</i> during processing of the channel protocols PROTOCOL_EAC_ID, PROTOCOL_EAC_MRTD, and PROTOCOL_PACE_CAM to update the referenceTime.
TIME_SYNC_FROM_CLIENT	0x02	2	If enabled, the CSP permits Client Applications to set the referenceTime as <i>External Timekeeper</i> using the time.setReferenceTime operation.
TIME_SYNC_ENFORCE_NEWER	0x10	16	If enabled, the CSP ensures that any new referenceTime is always later than the previously time. If the incoming timestamp is not later, the update is simply ignored without any exceptions. Note: This check can only be disabled for timestamps provided by Client Applications. Reference times derived from Terminal Authentication (TA) certificates are always only considered when they contain newer times.
TIME_SYNC_PERSIST	0x20	32	If enabled, the new referenceTime is stored to persistent memory. Note 1: This option applies to referenceTime obtained from TA certificates and timestamps set by Client Applications. Note 2: Activating this option requires thoughtful consideration, as frequent updates of the reference time may endanger the secure element's non-volatile memory.
TIME_SYNC_VERIFY_SIG	0x40	64	If enabled, the CSP checks the signature of the incoming timestamp in the time.setReferenceTime operation and the CSPSetTime command before accepting it as new referenceTime. If the signature verification fails, the update is ignored without triggering an exception. The CSP Admin must configure the key used for verifying the signature through the timeVerificationKey option. Note: Signature verification applies only to new referenceTime provided by Client Applications, and not to new referenceTime derived from certificate creation dates.
TIME_SYNC_VERIFY_SIG_WITH_CHALLENGE	0x80	128	Functions same as TIME_SYNC_VERIFY_SIG with the difference that the signature must include the current timeVerificationChallenge generated by the CSP.

6.11.1.3 Timer Types

The TimeModule may support the timer types listed in Table 6-110; the exact list shall be subject to *Modularity*.

The timer type is not directly configured; instead, it is selected by creating a manual counter resource using the `RESOURCE_TIMER` type, or by activating any of the built-in timers `validityPeriod`, `validityDate`, `validityCertificate`, `authTimeout`, and `securityTimeout`.

The CSP Admin may detect if a `CSPTimerType` is supported using the `CSPTimeSupport` structure. Each timer type supported shall adhere to the timer combinations specified in:

- Table 6-111 for compatible *Timer Type, Format, Resource Type, and Timeout Type Combinations*

Note: The evaluation of configured timers depends on the supported *Time Modes*. The CSP may strictly fail with `ERROR_INVALID_INIT [0x40B0]` if the `systemTime` is not synchronized, or it may ignore the timer depending on the `timeMode` configured.

For further information, see section 3.10.1, Overview Timers.

Table 6-108: Timer Types

Timer Type	Value (Hex)	Value (Int)	Description
<code>TIMER_MANUAL_DATE</code>	<code>0x01</code>	1	<p><i>Manual Timer</i> available as <code>FORMAT_TIMESTAMP</code>; only available for resources of type <code>RESOURCE_COUNTER</code>.</p> <p>The CSP shall evaluate the <code>timerValue</code> each time the <code>time.isExpired</code> operation is invoked.</p> <p>When the timer of a resource in <code>STATE_OPERATIONAL</code> expires, the CSP shall set the resource to <code>STATE_EXPIRED</code>.</p> <p>The CSP shall store the configured timestamp in persistent memory.</p>
<code>TIMER_MANUAL_PERIOD</code>	<code>0x02</code>	2	<p><i>Manual Timer</i> computed from a given duration in <code>FORMAT_DURATION</code>; only available for resources of type <code>RESOURCE_COUNTER</code>.</p> <p>The CSP shall evaluate the <code>timerValue</code> each time the <code>time.isExpired</code> operation is invoked.</p> <p>The CSP shall compute the <code>timerValue</code> and set the timer resource to <code>STATE_OPERATIONAL</code> each time the <code>time.reset</code> operation is invoked.</p> <p>When the timer of a resource in <code>STATE_OPERATIONAL</code> expires, the CSP shall set the resource to <code>STATE_EXPIRED</code>.</p> <p>The CSP shall store the computed timestamp in persistent memory.</p>

Timer Type	Value (Hex)	Value (Int)	Description
TIMER_VALIDITY_PERIOD	0x03	3	<p><i>Validity Period</i> to compute the validity date from a given duration in <code>FORMAT_DURATION</code> for key and password resources.</p> <p>The CSP shall evaluate the <code>timerValue</code> of a <code>validityPeriod</code> configured for resources used in the following methods:</p> <ul style="list-style-type: none"> • <code>resource.getState</code> • <code>cipher.init</code>, <code>sig.init</code>, <code>transform.init</code> • <code>sc.processSecurity</code>, <code>att.computeAttestation</code> • <code>key.derive</code> (only for the source key) • <code>key.computeSharedSecret</code> (only for the source keys) • <code>pwd.check</code> • <code>time.setReferenceTime</code> (for signature verification key) • <code>audit.dequeueEvent</code> (for the audit signing key) • <code>offloading.initManage</code> (for the offloading key) <p>The CSP shall compute the <code>timerValue</code> (i.e., the validity date) each time the following methods are invoked:</p> <ul style="list-style-type: none"> • <code>key.generate</code>, <code>key.generateKeyPair</code>, <code>key.manage</code> (IMPORT) • <code>key.computePublicKey</code> (public key) • <code>key.derive</code> (destination key) • <code>key.computeSharedSecret</code> (shared secret) • <code>pwd.update</code> • <code>CSPSetValue</code>, <code>CSPGenerateKey</code> • <code>CSPComputePublicKey</code> (public key) • <code>CSPDeriveKey</code> (destination key) <p>When the timer expires, the CSP shall set the resource, for which the <code>validityPeriod</code> is configured, to <code>STATE_EXPIRED</code>.</p> <p>The CSP shall store the computed timestamp in persistent memory.</p>
TIMER_VALIDITY_DATE	0x04	4	<p><i>Validity Date for Keys and Passwords</i> given as specific Unix timestamp in <code>FORMAT_TIMESTAMP</code> for key and password resources.</p> <p>The CSP shall evaluate the <code>timeLimit</code> of a <code>validityDate</code> configured for resources used in the following methods:</p> <ul style="list-style-type: none"> • <code>resource.getState</code> • <code>cipher.init</code>, <code>sig.init</code>, <code>transform.init</code> • <code>sc.processSecurity</code>, <code>att.computeAttestation</code> • <code>key.derive</code> (only for the source key) • <code>key.computeSharedSecret</code> (only for the source keys) • <code>pwd.check</code> • <code>time.setReferenceTime</code> (for signature verification key) • <code>audit.dequeueEvent</code> (for the audit signing key) • <code>offloading.initManage</code> (for the offloading key) <p>When the timer expires, the CSP shall set the resource, for which the <code>validityDate</code> is configured, to <code>STATE_EXPIRED</code>.</p> <p>The CSP shall store the configured timestamp in persistent memory.</p>

Timer Type	Value (Hex)	Value (Int)	Description
TIMER_VALIDITY_CERTIFICATE	0x05	5	<p><i>Validity Date of Certificates</i> extracted from certificates.</p> <p>The CSP shall extract the validity data from the 'Certificate Validity Period' of certificates during their import via:</p> <ul style="list-style-type: none"> cert.manage (IMPORT) CSPSetValue <p>The CSP shall evaluate this validityCertificate, stored as timerValue, when the certificate is used in the following methods:</p> <ul style="list-style-type: none"> cipher.init, sig.init sc.processSecurity time.setReferenceTime (for signature verification key) <p>When the timer expires, the CSP shall set the certificate resource to STATE_EXPIRED.</p> <p>The CSP shall store the extracted timestamp in persistent memory.</p>
TIMER_AUTH_TIMEOUT	0x06	6	<p><i>Authentication Timeout</i> in FORMAT_TIMEOUT for password resources.</p> <p>The CSP shall evaluate a timerValue of an authTimeout configured for resources used in the following methods and policies:</p> <ul style="list-style-type: none"> pwd.isAuthenticated pwd.update POLICY_PASSWORD POLICY_UNBLOCK_PASSWORD POLICY_PRE_BLOCKED <p>The CSP shall compute the timerValue (i.e., the timeout) each time the following methods are invoked:</p> <ul style="list-style-type: none"> pwd.check (if successful) <p>When the timer expires, the CSP shall reset the authenticated flag of the password to false.</p> <p>The CSP shall store the computed timeout in transient memory with CLEAR_ON_RESET.</p> <p>Note: Depending on the timeoutType configured, the timeout may be refreshed when the password is used, as specified for FORMAT_TIMEOUT.</p>

Timer Type	Value (Hex)	Value (Int)	Description
TIMER_SECURITY_TIMEOUT	0x07	7	<p><i>Security Timeout</i> in FORMAT_TIMEOUT for secure channels.</p> <p>The CSP shall evaluate the <code>timerValue</code> of the <code>securityTimeout</code> configured for secure channels in the following methods and policies:</p> <ul style="list-style-type: none"> • <code>sc.initWrap</code> • <code>sc.initUnwrap</code> • <code>POLICY_SECCHANNEL_ESTABLISHED</code> <p>The CSP shall compute the <code>timerValue</code> (i.e., the timeout) each time the following methods are invoked:</p> <ul style="list-style-type: none"> • <code>sc.processSecurity</code> (if changes to <code>SECURITY_ESTABLISHED</code>) <p>When the timer expires, the CSP shall reset the <code>securityState</code> of the secure channel.</p> <p>The CSP shall store the computed timeout in transient memory with <code>CLEAR_ON_RESET</code>.</p> <p>Note: Depending on the <code>timeoutType</code> configured, the timeout may be refreshed when the secure channel is used, as specified for <code>FORMAT_TIMEOUT</code>.</p>

6.11.1.4 Time Formats

The `TimeModule` may support the time formats listed in Table 6-109; the exact list depends on the *Timer Types* supported.

The CSP Admin indirectly specifies the time format using the `CSPTimers` and the `CSPManualTimer` structures.

Table 6-109: Time Formats

Time Format	Value (Hex)	Value (Int)	Description
FORMAT_TIMEOUT	0x01	1	<p>A timeout in seconds as 2-byte signed integer (short), covering up to 9 hours.</p> <p>The encoding uses big-endian byte order and right justification for the number of requested bytes, with leading zeros.</p> <p>The <code>timerValue</code> is computed by adding the estimated <code>timeSinceBoot</code> to the <code>timeLimit</code> configured for the timer.</p> <p>If the current <code>timeSinceBoot</code> exceeds the computed <code>timerValue</code>, the timer is considered expired.</p> <p>Timers:</p> <ul style="list-style-type: none"> • <code>TIMER_AUTH_TIMEOUT</code>, <code>TIMER_SECURITY_TIMEOUT</code> <p>Data Type:</p> <ul style="list-style-type: none"> • <code>CSPTimeout</code>

Time Format	Value (Hex)	Value (Int)	Description
FORMAT_DURATION	0x02	2	<p>A duration in seconds as 4-byte signed integer, covering up to 68 years.</p> <p>The encoding uses big-endian byte order and right justification for the number of requested bytes, with leading zeros or sign extension as appropriate.</p> <p>The <code>timerValue</code> is computed by adding the <code>timeLimit</code> to either the current <code>timeSinceBoot</code> or the estimated <code>systemTime</code>, depending on the specific CSP implementation, capabilities, and the configured <code>timeMode</code>.</p> <p>If the current <code>timeSinceBoot</code> or estimated <code>systemTime</code> (depending on how the <code>timerValue</code> is calculated) exceeds the computed <code>timerValue</code>, the timer is considered expired.</p> <p>Timers:</p> <ul style="list-style-type: none"> TIMER_MANUAL_PERIOD, TIMER_VALIDITY_PERIOD <p>Data Type:</p> <ul style="list-style-type: none"> CSPDuration
FORMAT_TIMESTAMP	0x03	3	<p>Unix timestamp in seconds in 64-bit format as 8-byte signed integer, covering dates until 292 billion years from 1970.</p> <p>The encoding uses big-endian byte order and right justification for the number of requested bytes, with leading zeros or sign extension as appropriate.</p> <p>The <code>timerValue</code> is not computed; instead, the <code>timeLimit</code> is used directly as <code>timerValue</code>.</p> <p>If the estimated <code>systemTime</code> exceeds the configured <code>timeLimit</code>, the timer is considered expired.</p> <p>Timers:</p> <ul style="list-style-type: none"> TIMER_MANUAL_DATE, TIMER_VALIDITY_DATE, TIMER_VALIDITY_CERTIFICATE <p>Data Type:</p> <ul style="list-style-type: none"> CSPTimestamp

6.11.1.5 Timeout Types

The TimeModule may support the timeout types listed in Table 6-110; the exact list shall be subject to *Modularity*.

The CSP Admin may configure the `timeoutType` using the CSPTimeType structure.

The CSP Admin may detect if a timeout type is supported using the CSPTimeSupport structure.

For further information, see section 3.10.1.5, Authentication Timeout.

Table 6-110: Timeout Types

Timeout Type	Value (Hex)	Value (Int)	Description
TIMEOUT_OFF	0x00	0	Timeout functionality is disabled and not available.
TIMEOUT_HARD	0x01	1	Time limit applied after the authentication of a resource: Timeout in seconds until an authenticated password or secure channel gets invalidated.

Timeout Type	Value (Hex)	Value (Int)	Description
TIMEOUT_SOFT	0x02	2	Time limit applied after the authentication of a resource which gets refreshed (extended) after each use of the resource: Timeout in seconds until an authenticated password or secure channel gets invalidated when it was not used in between.

6.11.1.6 Timer Type, Format, Resource Type, and Timeout Type Combinations

The TimeModule shall support the compatible combinations of *Timer Types* listed in Table 6-111 for all *Time Formats*, *Resource Types*, and *Timeout Types* supported by the platform. Unsupported or illegal combinations shall be rejected and handled according to the errorMode configured.

Table 6-111: Timer Type, Format, Resource Type, and Timeout Type Combinations

Timer Type	Time Format	Resource Type	Timeout Type
TIMER_MANUAL_DATE	FORMAT_TIMESTAMP	RESOURCE_TIMER	-
TIMER_MANUAL_PERIOD	FORMAT_DURATION	RESOURCE_TIMER	-
TIMER_VALIDITY_PERIOD	FORMAT_DURATION	RESOURCE_KEY, RESOURCE_PASSWORD	-
TIMER_VALIDITY_DATE	FORMAT_TIMESTAMP	RESOURCE_KEY, RESOURCE_PASSWORD	-
TIMER_VALIDITY_CERTIFICATE	FORMAT_TIMESTAMP	RESOURCE_CERTIFICATE	-
TIMER_AUTH_TIMEOUT	FORMAT_TIMEOUT	RESOURCE_PASSWORD	TIMEOUT_HARD, TIMEOUT_SOFT
TIMER_SECURITY_TIMEOUT	FORMAT_TIMEOUT	-	TIMEOUT_HARD, TIMEOUT_SOFT

6.11.1.7 Time Events

The AuditModule may log the time-related *Audit Events* listed in Table 6-112; the exact list shall be subject to *Modularity*. The event-specific data listed in the table shall be stored according to the configured auditMode.

The CSP Admin may configure resourceEvents to be logged using the CSPResourceEvent structure and systemEvents to be logged using the CSPSystemEvent structure.

The CSP Admin may detect if an event type is supported using the CSPAuditSupport structure.

For further information, see section 3.11.4, Selecting Audit Events, and section 6.12.1.2, Audit Events.

Table 6-112: Time Events for Auditing

Event Type	Value (Hex)	Value (Int)	Description	Event Category
EVENT_TIMER_EXPIRED	0x10B0	4272	Resource validity date expired. This event is logged when a resource state changes to STATE_EXPIRED. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Operations: all that involve timers Event Data: CSPEventDataResource 	Resource Event

Event Type	Value (Hex)	Value (Int)	Description	Event Category
EVENT_CSP_TIME_SET	0x00B0	176	<p>Successful update of system clock. This event logs every successful change to the referenceTime required for <i>Time Synchronization</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Message Format: CSPLogMessage Event Data: CSPEventDataSetTime <p>Operations:</p> <ul style="list-style-type: none"> time.setReferenceTime sc.processSecurity 	System Event

6.11.1.8 Time Error Codes

If ERROR_MODE_DETAILED is supported and activated, the TimeModule shall use the error codes listed in Table 6-113 as reasons for exceptions.

For further information, see section 5.1.1.6, Error Types.

Table 6-113: Time Error Reason Codes

Reason	Description
0x20B0	Illegal timer type: Timer type is not supported by the operation.
0x30B0	Missing time verification configuration: No CSP Instance-specific time verification key configured.
0x40B0	Time not synchronized: No reference time set for the current session and time mode is set to TIME_MODE_STRICT.
0x50B0	Wrong usage: Resource not configured for USAGE_SIGNATURE.
0x50B1	Timer expired: A resource used is in or has changed to STATE_EXPIRED.
0x50B2	Not a time admin: The Client Application AID is not granted to update the reference time.
0x50B3	Time signature verification failed: Signature verification was enabled but did not succeed for the provided timestamp.
0x80B0	Unsupported: The time module is not supported.
0x80B1	Unsupported: Time since boot not supported and TIME_MODE_STRICT is set.
0x80B2	Unsupported: System time is not supported and TIME_MODE_STRICT is set.
0x80B3	Unsupported: The time mode is not supported.
0x80B4	Unsupported: The timer type is not supported and TIME_MODE_STRICT is set.

6.11.2 Time Configuration

The TimeModule shall support the configuration parameters defined in Table 6-114.

Table 6-114: Time Configuration Parameters

Parameter	Description	Parameter Type
timeMode	<p>The time mode defines how the CSP shall handle scenarios where a configured timer is not supported by the platform. It is selected from the available <i>Time Modes</i>.</p> <p>Technical Details</p> <ul style="list-style-type: none"> • Data type: CSPTIME_MODE • Initial value: 0 (TIME_MODE_OFF) • Administration: CSPTIME_SETTINGS 	CSP Instance Parameter
timeSynchronization	<p>The time synchronization is a bitmask defining how the CSP shall synchronize the referenceTime. It is selected from the available <i>Time Synchronization Methods</i>.</p> <p>Technical Details</p> <ul style="list-style-type: none"> • Data type: CSPTIME_SYNCHRONIZATION • Initial value: 0 (time not synchronized) • Administration: CSPTIME_SETTINGS 	CSP Instance Parameter
validityDate	<p>A <i>Validity Date for Keys and Passwords</i> configured as a Unix timestamp, which makes the resource unusable once the specified date has expired, as specified by TIMER_VALIDITY_DATE.</p> <p>Technical Details</p> <ul style="list-style-type: none"> • Data type: CSPTIME_TIMESTAMP • Administration: CSPTIME_TIMERS • Operations: time.getValue 	Resource Parameter
validityPeriod	<p>A <i>Validity Period</i> used to calculate a validity date for a resource, making the resource unusable once the computed date has expired, as specified by TIMER_VALIDITY_PERIOD.</p> <p>Technical Details</p> <ul style="list-style-type: none"> • Data type: CSP_DURATION • Administration: CSPTIME_TIMERS • Operations: time.getValue 	Resource Parameter
validityCertificate	<p>Extract the <i>Validity Date of Certificates</i>, making the resource unusable once the extracted date has expired, as specified by TIMER_VALIDITY_CERTIFICATE.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: BOOLEAN • Administration: CSPTIME_TIMERS • Operations: time.getValue 	Resource Parameter
authTimeout	<p>An <i>Authentication Timeout</i> that resets the authenticated flag of a password when the timeout expires, as specified by TIMER_AUTH_TIMEOUT.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPTIME_TIMEOUT • Administration: CSPTIME_TIMERS • Operations: time.getValue 	Password Parameter

Parameter	Description	Parameter Type
timeLimit	<p>Specific date or period in seconds before the resource changes to STATE_EXPIRED. It is used to assign the timerValue, as specified in Table 6-109 <i>Time Formats</i>, e.g. upon <i>Load Resource for Personalization, Import and Export Public Keys</i>, or <i>Updating Passwords</i>.</p> <p>Technical Details</p> <ul style="list-style-type: none"> • Data type: varies, either CSPTimestamp, CSPDuration, or CSPTimeout • Initial value: 0x0 (disabled) • Administration: CSPManualTimer 	Timer Parameter
timerType	<p>Defines how the timer is evaluated, as specified by the available <i>Timer Types</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPTimerType • Administration: - (indirect, see <i>Timer Types</i>) 	-
timeoutType	<p>Timeout type, e.g., distinguishing if a timeout can be refreshed, selected from available <i>Timeout Types</i>.</p> <p>Technical Details</p> <ul style="list-style-type: none"> • Data type: CSPTimeoutType • Initial value: 0x01 (TIMEOUT_HARD - no refresh) • Administration: CSPManualTimer 	Timer Parameter
referenceTime	<p>Reference time set by the CSP Admin, retrieved from TA certificates or set by an authorized Client Application depending on the <i>Time Synchronization</i> mechanism used.</p> <p>Technical Details</p> <ul style="list-style-type: none"> • Data type: CSPTimestamp • Initial value: 0xFFFFFFFF • Administration: CSPTimeSettings • Operations: time.setReferenceTime • Events: CSPEventData 	CSP Instance Parameter
timeVerificationKey	<p>This is the public key value of the <i>Time Verification Key</i>. This key must have USAGE_SIGNATURE and is used to verify the signature of new reference times provided through the time.setReferenceTime operation or the CSPSetTime command.</p> <p>Note: Signature verification applies only to new reference times provided by Client Applications or CSP Admins, and not to new reference times derived from TA certificates.</p> <p>Technical Details</p> <ul style="list-style-type: none"> • Data type: CSPResourceId • Administration: CSPTimeSettings • Initial value: null • Operations: time.setReferenceTime 	CSP Instance Parameter
timeVerificationChallenge	<p>If TIME_SYNC_VERIFY_SIG_WITH_CHALLENGE is enabled, the CSP generates and stores a challenge in transient memory (CLEAR_ON_RESET) to verify the signature of newly provided referenceTime. The CSP creates this challenge at system start and whenever a new referenceTime is set.</p> <ul style="list-style-type: none"> • Data type: CSPChallenge • Administration: - • Initial value: random data • Operations: time.getChallenge 	CSP Instance Parameter

6.11.3 Time Operations

The TimeModule shall implement the operations listed in Table 6-115.

These operations shall comply with the *Access Restrictions on Time Operations* specified in Table 6-116.

For further information, see section 3.10, Timer and Time Management.

Table 6-115: Time Operations

Operation	Description
time.hasTimer	Retrieve whether a specific timer type is activated for the resource. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.TimeService.hasTimer(..)
time.getValue	Retrieve the current timerValue of a timer. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.TimeService.getValue(..)
time.isExpired	Check if the current counterValue of a timer has expired. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.TimeService.isExpired(..)
time.setReferenceTime	Sets a new referenceTime to the CSP Instance. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.TimeService.setReferenceTime(..) CSP Protocol: <ul style="list-style-type: none"> CSPSetTime Evaluate Timers: <ul style="list-style-type: none"> TIMER_VALIDITY_DATE (for the signature verification key) TIMER_VALIDITY_PERIOD (for the signature verification key) Fire Events: <ul style="list-style-type: none"> EVENT_CSP_TIME_SET
time.reset	Reset a timer of type TIMER_MANUAL_PERIOD. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.TimeService.reset(..)
time.getChallenge	Returns the timeVerificationChallenge that is used to verify the signature of a new referenceTime. CSP API: <ul style="list-style-type: none"> org.globalplatform.csp.api.TimeService.getChallenge(..)

6.11.3.1 Access Restrictions on Time Operations

The TimeModule shall enforce the access restrictions for *Time Operations* listed in Table 6-116 for all *Access Rights*, *Usage Types*, *Resource States*, and *Policy Types* supported by the platform. Unauthorized or illegal operation invocations shall be rejected and handled according to the errorMode configured.

Table 6-116: Time Operations Access Restrictions

Operation	Right Required	Usage Required	State Required	Other Restrictions
time.setReferenceTime (time verification key)	USE	SIGNATURE	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required Only Client Applications granted for TIME_SYNC_FROM_CLIENT

Operation	Right Required	Usage Required	State Required	Other Restrictions
time.reset	SETUP	-	-	<ul style="list-style-type: none"> Client Authentication required Only TIMER_MANUAL_PERIOD
time.hasTimer	USE	-	-	<ul style="list-style-type: none"> Client Authentication required Not TIMER_MANUAL_PERIOD and not TIMER_MANUAL_DATE
time.getValue	USE	-	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required
time.isExpired	-	-	-	<ul style="list-style-type: none"> Client Authentication required
time.getChallenge	-	-	-	<ul style="list-style-type: none"> Client Authentication required

6.11.3.2 Sensitive Results computed by Time Operations

The TimeModule shall temporarily store the results of methods listed in Table 6-117 in transient memory and shall implement the assertSensitiveResult operation, enabling Client Applications to verify that these results were computed without any abnormality.

Table 6-117: Time Operations Requiring Sensitive Results Checks

Operation	Result Type	Result Description
time.hasTimer	short (CSPBoolean)	The CSPBoolean returned.
time.getValue	short	The length of the timerValue returned.
time.isExpired	short (CSPBoolean)	The CSPBoolean returned.
time.getChallenge	short	The length of the timeVerificationChallenge returned.

6.11.3.3 Sensitive Arrays Required for Time Operations

The TimeModule shall invoke javacard.framework.SensitiveArrays.assertIntegrity(..) ([JCAPI]) on the parameters listed in Table 6-118.

For further information, see section 5.1.3.4, Sensitive Arrays, and section 8.3, Byte Buffer Parameters.

Table 6-118: Time Operations Requiring Sensitive Arrays

Operation	Parameter	Parameter Description
time.getValue	Output buffer	The retrieved timer value.
time.setReferenceTime	Input buffer	New reference time.
time.setReferenceTime	Signature buffer	Signature of the reference time.
time.getChallenge	Output buffer	The challenge used for timestamp signature verification.

6.11.4 Time Lifecycle

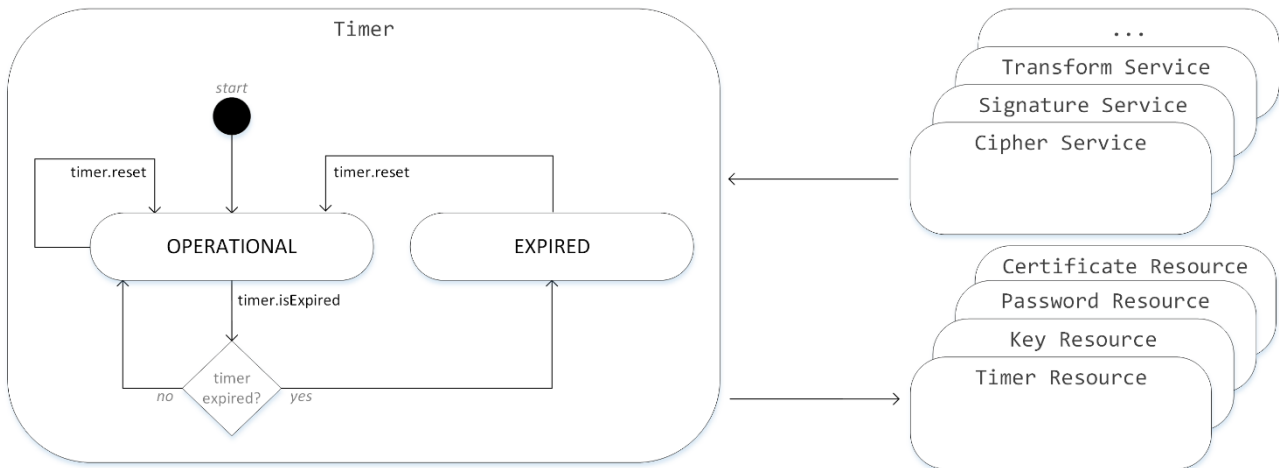
The operations of the time service are stateless. A time service is an instance of the TimeModule.

The time service stores *Sensitive Results computed by Time Operations*.

The CSP shall trigger lifecycle changes to timers, as illustrated in Figure 6-11. A timer represents either a timer resource or a CSP-internal data structure used to maintain timers for keys, certificates and passwords.

A timer resource is started manually and transitions to STATE_EXPIRED when the timer exceeds the current estimated systemTime. The CSP-internal validityDate, validityPeriod, validityCertificate are managed by the CSP and transition the resource owning the timer to STATE_EXPIRED when the timer exceeds the current estimated systemTime. The CSP-internal authTimeout and securityTimeout reset the authenticated flag or securityState that owns the timer.

Figure 6-11: Timer Lifecycle



6.11.4.1 Internal Time Parameters

The TimeModule shall maintain the internal parameters listed in Table 6-119.

For further information, see section 3.10, Timer and Time Management.

Table 6-119: Time-internal Parameters

Parameter	Description	Parameter Type
timerValue	<p>Contains either a specific timestamp or the specific time when the timer is considered expired. It is computed from the <code>timeLimit</code>, as specified in Table 6-109 <i>Time Formats</i>, e.g., upon <i>Load Resource for Personalization</i>, <i>Import and Export Public Keys</i>, or <i>Updating Passwords</i></p> <p>Technical Details:</p> <ul style="list-style-type: none"> Data type: varies, CSPTimestamp, CSPDuration, or CSPTIMEOUT Initial value: 0x0 (not active) Administration: - Operations: <code>time.getValue</code> 	Timer Parameter
systemTime	<p>Current system time Unix timestamp in seconds in <code>FORMAT_TIMESTAMP</code>. It is estimated by the CSP from the <code>referenceTime</code> using <i>Time Synchronization</i> mechanisms. It is computed from the <code>referenceTime</code> and not stored in memory.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Data type: CSPTimestamp Initial value: 0xFFFFFFFF Fields: <code>FIELD_SYSTEM_TIME</code> 	-

Parameter	Description	Parameter Type
timeSinceBoot	<p>The elapsed since boot in seconds in FORMAT_DURATION. Only available, if the target platform is equipped with a timer.</p> <p>It is stored in persistent memory.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPDuration • Initial value: 0xFFFF • Fields: FIELD_TIME_SINCE_BOOT 	CSP Instance Parameter

6.11.5 Time Structures

This section lists ASN.1 structures related to the configuration of the CSP's time and timer functionality.

Note: A complete ASN.1 definition is available in [GP CSP ASN1].

6.11.5.1 CSPTIMEsupport

This data structure represents features, types, and algorithms of the *Time Module*.

It is included in the CSPEnforce command to detect their support by the platform.

ASN 6-44: Time: ASN.1 Definition for CSPTIMEsupport

```
-- ASN1START
-- Checks for time management support and supported timer types.
CSPTIMEsupport ::= SEQUENCE {

    -- Time operation modes to specify the handling for unsupported time.
    timeModes          SET OF CSPTIMEmode OPTIONAL,

    -- Time synchronization strategies.
    timeSynchronization CSPTIMEsynchronization OPTIONAL,

    -- Timer types, such as validity date or authentication timeout.
    timerTypes         SET OF CSPTIMEtimerType OPTIONAL,

    -- Timeout types to specify if a timeout timer is refreshed.
    timeoutTypes       SET OF CSPTIMEtimeoutType OPTIONAL
}
-- ASN1STOP
```

6.11.5.2 CSPTIMEsettings

This data structure represents a general *Time Configuration* for the CSP Instance.

It can be configured using the CSPSetup structure.

It can be retrieved using the CSPConfiguration structure.

ASN 6-45: Time: ASN.1 Definition for CSPTIMEsettings

```
-- ASN1START
```

```
-- General time configuration settings for the CSP Instance.
CSPTIMESETTINGS ::= SEQUENCE {

    -- Select time mode to specify the handling of unavailable system time.
    timeMode                CSPTIMEMODE DEFAULT time-mode-off,

    -- Configure a strategy to synchronize the reference time.
    timeSynchronization      CSPTIME SYNCHRONIZATION DEFAULT {},

    -- A public key to verify signatures of new timestamps.
    timeVerificationKey      CSPRESOURCEID OPTIONAL
}
-- ASN1STOP
```

6.11.5.3 CSPTIMEMODE

This data structure defines the list of available of *Time Modes*.

A time mode can be selected using the CSPTIMESETTINGS structure.

The time modes supported can be detected using the CSPTIMESUPPORT structure.

ASN 6-46: Time: ASN.1 Definition for CSPTIMEMODE

```
-- ASN1START
-- List of available time management modes.
CSPTIMEMODE ::= ENUMERATED {

    -- TIME_MODE_OFF:
    -- Time functionality is disabled or not available.
    time-mode-off           (0),

    -- TIME_MODE_IGNORE:
    -- Ignore time-related configurations if time is not supported.
    time-mode-ignore        (1),

    -- TIME_MODE_STRICT:
    -- Stop operation if time is not synchronized.
    time-mode-strict        (2)
}
-- ASN1STOP
```

6.11.5.4 CSPTIMESYNCHRONIZATION

This data structure represents the *Time Synchronization Methods* bitmask.

The timeSynchronization can be configured using the CSPTIMESettings structure.

The synchronization methods supported can be detected using the CSPTIMESupport structure.

For further information, see section 3.2.4, Access Control Rules Bitmask.

ASN 6-47: Time: ASN.1 Definition for CSPTIMEsynchronization

```
-- ASN1START
-- Bitmask to configure the time synchronization strategy.
CSPTIMEsynchronization ::= BIT STRING {

    -- TIME_SYNC_FROM_TA:
    -- Update the reference time from TA2 certificates during EAC v2.
    time-sync-from-ta                (0),

    -- TIME_SYNC_FROM_CLIENT:
    -- Permit Client Applications to set the reference time.
    time-sync-from-client            (1),

    -- TIME_SYNC_ENFORCE_NEWER:
    -- Accept only newer timestamps.
    time-sync-enforce-newer          (4),

    -- TIME_SYNC_PERSIST:
    -- Persist the reference time.
    time-sync-persist                (5),

    -- TIME_SYNC_VERIFY_SIG:
    -- Verify the timestamp signature.
    time-sync-verify-sig             (6),

    -- TIME_SYNC_VERIFY_SIG_WITH_CHALLENGE:
    -- Verify the timestamp signature using a challenge generated by CSP.
    time-sync-verify-sig-with-challenge (7)
}
-- ASN1STOP
```

6.11.5.5 CSPTimerType

This data structure defines the list of available of *Timer Types*.

A timer format can be selected using the CSPManualTimer structure.

The time types supported can be detected using the CSPTIMESupport structure.

ASN 6-48: Time: ASN.1 Definition for CSPTimerType

```
-- ASN1START
-- List of available timer types.
CSPTimerType ::= ENUMERATED {
```

```

-- TIMER_MANUAL_DATE:
-- Manual timer invoked by the Client Application.
timer-manual-date          (1),

-- TIMER_MANUAL_PERIOD:
-- Manual timer invoked by the Client Application.
timer-manual-period        (2),

-- TIMER_VALIDITY_PERIOD:
-- Validity period used to compute (and refresh) the validity date.
timer-validity-period      (3),

-- TIMER_VALIDITY_DATE:
-- Validity date as specific Unix timestamp.
timer-validity-date        (4),

-- TIMER_VALIDITY_CERTIFICATE:
-- Validity date extracted from certificates.
timer-validity-certificate (5),

-- TIMER_AUTH_TIMEOUT:
-- Timeout for authenticated passwords.
timer-auth-timeout         (6),

-- TIMER_SECURITY_TIMEOUT:
-- Timeout for secure channel service.
timer-security-timeout     (7)
}
-- ASN1STOP

```

6.11.5.6 CSPTimers

This data structure represents a container for a resource-specific *Time Configuration*.

It is part of the CSPResource structure.

It can be modified using the CSPConfigureResource command.

The timer types supported can be detected using the CSPTimeSupport structure.

ASN 6-49: Time: ASN.1 Definition for CSPTimers

```

-- ASN1START
-- Container for built-in timers that can be configured to a resource.
CSPTimers ::= SEQUENCE {

    -- Validity period for key or password (TIMER_VALIDITY_PERIOD).
    validityPeriod          CSPDuration OPTIONAL,

```

```
-- Validity date for key or password (TIMER_VALIDITY_DATE).
validityDate          CSPTimestamp OPTIONAL,

-- Validity date from certificate (TIMER_VALIDITY_CERTIFICATE).
validityCertificate    BOOLEAN OPTIONAL,

-- Authentication timeout for passwords (TIMER_AUTH_TIMEOUT).
authTimeout           CSPTimestamp OPTIONAL
}
-- ASN1STOP
```

6.11.5.7 CSPTimestamp

This data structure represents a Unix timestamp in seconds, as defined for FORMAT_TIMESTAMP. It is part of the CSPTimers and CSPSetTime structures.

ASN 6-50: Time: ASN.1 Definition for CSPTimestamp

```
-- ASN1START
-- Unix timestamp in seconds as 8-byte signed int (292 billion years >1970).
CSPTimestamp ::= OCTET STRING (SIZE (8))
-- ASN1STOP
```

6.11.5.8 CSPDuration

This data structure represents time duration in seconds, as defined for FORMAT_DURATION. It is part of the CSPTimers structure and used to compute a new validity date each time a key is generated or imported.

ASN 6-51: Time: ASN.1 Definition for CSPDuration

```
-- ASN1START
-- Duration in seconds, represented as 4-byte signed integer (up to 68 years).
CSPDuration ::= OCTET STRING (SIZE (4))
-- ASN1STOP
```

6.11.5.9 CSPTimeout

This data structure represents a timeout in seconds, as specified for FORMAT_TIMEOUT. It is part of the CSPTimers and CSPSecureChannelSettings structures.

ASN 6-52: Time: ASN.1 Definition for CSPTimeout

```
-- ASN1START
-- Timeout in seconds, represented as 2-byte signed integer (up to 9 hours).
CSPTimeout ::= SEQUENCE {

    -- Maximum time limit before the timer expires.
    timeoutValue      OCTET STRING (SIZE (2)),
```

```
-- Specifies if the timeout value is re-computed on each resource usage.
timeoutType          CSPTimeoutType DEFAULT timeout-hard
}
-- ASN1STOP
```

6.11.5.10 CSPTimeoutType

This data structure defines the list of available of *Timeout Types*.

A timeout type can be selected using the CSPTimeout structure.

The timeout types supported can be detected using the CSPTimeSupport structure.

ASN 6-53: Time: ASN.1 Definition for CSPTimeoutType

```
-- ASN1START
-- List of available authentication timeout types.
CSPTimeoutType ::= ENUMERATED {

    -- Timeout functionality is disabled and/or not available.
    timeout-off          (0),

    -- TIMEOUT_HARD:
    -- Fixed timeout after which the authenticated state is invalidated.
    timeout-hard         (1),

    -- TIMEOUT_SOFT:
    -- Dynamic timeout that is refreshed when using the resource.
    timeout-soft         (2)
}
-- ASN1STOP
```

6.11.5.11 CSPManualTimer

This data structure represents a manual counter resource of type RESOURCE_TIMER.

It is part of the CSPResource structures.

It can be modified using the CSPConfigureResource command.

ASN 6-54: Time: ASN.1 Definition for CSPManualTimer

```
-- ASN1START
-- Manual timer configuration.
CSPManualTimer ::= CHOICE {

    -- The timer has a fixed expiration date.
    expirationDate      CSPTimestamp,

    -- Maximum time limit before the timer expires.
    expirationPerid     CSPDuration,
}
```

```
-- Specifies if the timer is re-computing the time value on each usage.  
timeoutValue      CSPTimeout  
}  
-- ASN1STOP
```

6.12 Audit Module

The CSP may implement the AuditModule to offer *Secure Auditing*; the choice shall be subject to *Modularity*.

The CSP Admin shall be able to configure the events to be logged and define log message format by:

- Configuring resourceEvents to be logged the CSPResourceEvent structure.
- Configuring systemEvents to be logged the CSPSystemEvent structure.
- Creating the auditSigningKey as a CSPResource and setting it via the CSPAuditSettings structure.
- Specifying the paddingAlgorithm, signatureAlgorithm, and messageDigestAlgorithm.
- Configuring fieldsAddedAsPrefix and fieldsAddedAsSuffix to the auditSigningKey from available *Fields*.

The CSP shall log events and store them in internal audit event queue according to the supported *Audit Modes* and shall generate audit log messages when Client Applications fetch events via the audit.dequeueEvent operation.

The CSP Admin may detect whether the platform supports this module using the CSPEnforce command.

For further information, see section 3.11, Secure Auditing.

6.12.1 Audit Definitions

6.12.1.1 Audit Modes

The AuditModule may support the audit operation modes listed in Table 6-120; the exact list shall be subject to *Modularity*. The selected audit mode defines how the CSP shall handle scenarios when the audit event queue reaches its maximum capacity or configured event types are not supported by the platform.

The CSP Admin may select the auditMode using the CSPAuditMode structure.

The CSP Admin may detect if an audit mode is supported using the CSPAuditSupport structure.

For further information, see section 3.11.2, Processing Log Messages, and section 3.11.5, Limitations of Audit.

Table 6-120: Audit Operation Modes

Mode	Value (Hex)	Value (Int)	Description
AUDIT_MODE_OFF	0x00	0	<p>Audit functionality is disabled and not available.</p> <p>Regardless of whether any <i>Audit Events</i> are configured, the AuditModule shall not log any events and shall operate without errors.</p> <p>This is the standard mode when audit is not supported by the platform.</p> <p>Note: This mode may be used to temporarily disable audit event logging even if audit is supported by the platform.</p>

Mode	Value (Hex)	Value (Int)	Description
AUDIT_MODE_OVERWRITE	0x01	1	<p>Overwrite events when the audit event queue is full.</p> <p>The AuditModule shall temporarily store events and all event-related data (including dedicated <i>Fields</i>) required to create the log message in persistent memory until fetched using a first-in, first-out (FIFO) queue specific to each CSP Instance.</p> <p>The AuditModule shall discard the oldest event to make space for new events when the audit event queue reaches capacity. For example, if a queue with size three already contains events E1, E2, and E3, and a new event E4 arrives, the CSP updates to E2, E3, and E4, removing E1.</p> <p>Fetching log messages by Client Applications follows the same FIFO principles. Thus, for a queue containing E2, E3, and E4, calling <code>audit.dequeueEvent</code> will return E2, and E1 is lost.</p> <p>For unsupported <i>Audit Events</i>, the AuditModule shall ignore the specific event configuration.</p> <p>The AuditModule shall temporarily store events and all event-specific data required to create the log message in persistent memory until fetched.</p>
AUDIT_MODE_STRICT	0x02	2	<p>Stop operation when the audit event queue is full.</p> <p>The AuditModule shall temporarily store events and all event-related data (including dedicated <i>Fields</i>) required to create the log message in persistent memory until fetched using a first-in, first-out (FIFO) queue specific to each CSP Instance.</p> <p>The AuditModule shall throw <code>ERROR_INVALID_INIT [0x40C0]</code> when attempting to add a new log entry after the audit event queue has reached its limit.</p> <p>This restriction remains until at least one log message is fetched by the Client Application through <code>audit.dequeueEvent</code>, thereby creating space in the queue.</p> <p>Fetching log messages by Client Applications adheres to FIFO principles, ensuring that in this mode, no events are lost.</p> <p>For unsupported <i>Audit Events</i>, the AuditModule shall throw an <code>0x80C2</code> exception when trying to log the affected event.</p>

6.12.1.2 Audit Events

The CSP may support the following events for auditing:

- Table 5-4 *Core System Events*
- Table 5-18 *Core Resource Events*
- Table 6-6 *Cipher Events*
- Table 6-17 *Signature Events*
- Table 6-38 *Secure Channel Events*
- Table 6-68 *Key Events*
- Table 6-77 *Certificate Events*
- Table 6-85 *Password Events*
- Table 6-97 *Counter Events*

- Table 6-112 *Time Events*
- Table 6-129 *Offloading Events*

For further information, see section 3.11.4, Selecting Audit Events.

6.12.1.3 Audit Error Codes

If `ERROR_MODE_DETAILED` is supported and activated, the `AuditModule` shall use the error codes listed in Table 6-121 as reasons for exceptions.

For further information, see section 5.1.1.6, Error Types.

Table 6-121: Audit Error Reason Codes

Reason	Description
0x30C0	Missing configuration: No CSP Instance-specific audit signing key configured.
0x40C0	Events not fetched: The audit event queue is full and <code>AUDIT_MODE_STRICT</code> is set.
0x50C0	Wrong usage: Resource not configured for <code>USAGE_AUDIT</code> .
0x80C0	Unsupported: The audit module is not supported.
0x80C1	Unsupported: The audit mode is not supported.
0x80C2	Unsupported: The event type is not supported and <code>AUDIT_MODE_STRICT</code> is set.

6.12.2 Audit Configuration

The `AuditModule` shall support the configuration parameters defined in Table 6-122.

For further information, see section 3.11.2, Processing Log Messages.

Table 6-122: Audit Configuration Parameters

Parameter	Description	Parameter Type
<code>auditMode</code>	<p>The audit mode defines how the CSP shall handle scenarios when the audit queue reaches its maximum capacity or an event type is not supported by the platform. It is selected from the available <i>Audit Modes</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: <code>CSPAuditMode</code> • Initial value: <code>0</code> (<code>AUDIT_MODE_OFF</code>) • Administration: <code>CSPAuditSettings</code> 	CSP Instance Parameter
<code>resourceEvents</code>	<p>Resource-specific <i>Audit Events</i> to be logged. The CSP Admin may <i>Configure Resources</i> to select the events that shall be logged.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: <code>SET OF CSPResourceEvent</code> • Initial value: empty list • Administration: <code>CSPResource</code>, <code>CSPConfigureResource</code> 	Resource Parameter
<code>systemEvents</code>	<p>CSP Instance-specific <i>Audit Events</i> to be logged, such as changes of the reference time.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: <code>SET OF CSPSystemEvent</code> • Initial value: empty list • Administration: <code>CSPAuditSettings</code> 	CSP Instance Parameter

Parameter	Description	Parameter Type
auditSigningKey	<p>A CSP Instance-specific resource ID referring to for the audit signing key used for <i>Creating Log Messages</i> by adding signature fields to the log data and then signing it.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> • Data type: CSPResourceId • Algorithm: CSPAttestationAlgorithms • Administration: CSPAuditSettings • Initial value: null 	CSP Instance Parameter

6.12.3 Audit Operations

The AuditModule shall implement the operations listed in Table 6-116.

For each operation, the CSP shall handle supported *Counter Types*, *Timer Types* and *Audit Events*.

The operations shall exclusively utilize resources configured with USAGE_CONFIDENTIAL as audit signing key and shall comply with the *Access Restrictions on Audit Operations* specified in Table 6-124.

Note: Specifically, the resources with USAGE_CONFIDENTIAL, shall not be permitted for signature creation using *Signature Operations* or *Attestation Operations*.

For further information, see section 3.11, Secure Auditing.

Table 6-123: Audit Operations

Operation	Description
audit.notifyPendingEvents	<p>Returns TRUE if there are pending events, covering resourceEvents and systemEvents. If the queue is not empty, the CSP notifies the Client Application via the <i>Audit Listener</i> callback.</p> <p>CSP API:</p> <ul style="list-style-type: none"> • org.globalplatform.csp.api.AuditService.notifyPendingEvents(..)
audit.getRemainingCapacity	<p>Retrieve the remaining capacity of the audit event queue to quantify the <i>Limited Event Buffering Capacity</i>.</p> <p>CSP API:</p> <ul style="list-style-type: none"> • org.globalplatform.csp.api.AuditService.getRemainingCapacity(..)
audit.getNumberOfPendingEvents	<p>Retrieve the current number of pending events in the audit event queue.</p> <p>CSP API:</p> <ul style="list-style-type: none"> • org.globalplatform.csp.api.AuditService.getNumberOfPendingEvents(..)
audit.dequeueEvent	<p>Generates the log message for the oldest event and removes it from the queue, enabling Client Applications for <i>Processing Log Messages</i>.</p> <p>CSP API:</p> <ul style="list-style-type: none"> • org.globalplatform.csp.api.AuditService.dequeueEvent(..) <p>Evaluate Timers (for the audit signing key):</p> <ul style="list-style-type: none"> • TIMER_VALIDITY_DATE, TIMER_VALIDITY_PERIOD <p>Increment Counters:</p> <ul style="list-style-type: none"> • No counters incremented for the audit key. <p>Fire Events:</p> <ul style="list-style-type: none"> • EVENT_TIMER_EXPIRED, EVENT_CSP_ERROR • No other events fired, not even signature events.

6.12.3.1 Access Restrictions on Audit Operations

The AuditModule shall enforce the access restrictions for *Audit Operations* listed in in Table 6-124 for all *Access Rights*, *Usage Types*, *Resource States*, and *Policy Types* supported by the platform. Unauthorized or illegal operation invocations shall be rejected and handled according to the errorMode configured.

Table 6-124: Audit Operations Access Restrictions

Operation	Right Required	Usage Required	State Required	Other Restrictions
audit.dequeueEvent (audit signing key)	USE	AUDIT	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required

6.12.3.2 Sensitive Results Computed by Audit Operations

The AuditModule shall temporarily store the results of methods listed in Table 6-125 in transient memory and shall implement the assertSensitiveResult operation, enabling Client Applications to verify that these results were computed without any abnormality.

Table 6-125: Audit Operations Requiring Sensitive Results Checks

Operation	Result Type	Result Description
audit.dequeueEvent	short	The length of the resulting signed log message.

6.12.3.3 Sensitive Arrays Required for Audit Operations

The AuditModule shall invoke javacard.framework.SensitiveArrays.assertIntegrity(..) ([JCAP]) on the parameters listed in Table 6-126.

For further information, see section 5.1.3.4, Sensitive Arrays, and section 8.3, Byte Buffer Parameters.

Table 6-126: Audit Operations Requiring Sensitive Arrays

Operation	Parameter	Parameter Description
audit.dequeueEvent	Input data	Additional input data for the log message.
audit.dequeueEvent	Output buffer	The generated log message.

6.12.3.4 Audit Listener

The AuditModule shall actively notify the Client Application about event occurrences via the *Listener Mechanism* of the CSP API using the callback methods listed in Table 6-127.

Note: If multiple Client Applications are registered to the same CSP Instance, each providing a listener instance, all listeners will be notified of the same event. However, audit.dequeueEvent can only be called once per event, as each call will remove the event from the audit event queue.

For further information, see section 3.11.2, Processing Log Messages, and section 8.4, Listener Mechanism.

Table 6-127: Audit Listener

Operation	Description
listener.auditEventOccurred	<p>Called by the CSP when a configured event occurs, informing Client Applications for further <i>Processing Log Messages</i>. Only events configured by the CSP Admin will be logged.</p> <ul style="list-style-type: none"> CSP API: org.globalplatform.csp.api.AuditListener.auditEventOccurred(..)

Operation	Description
listener.auditEventsPending	Called by the CSP after the Client Application calls audit.notifyPendingEvents, but only if there are pending events. <ul style="list-style-type: none"> CSP API: org.globalplatform.csp.api.AuditListener.auditEventsPending(..)

6.12.4 Audit Lifecycle

The operations of the audit service are stateless. The audit service is an instance of the AuditModule.

The audit service stores *Sensitive Results Computed by Audit Operations*.

It stores events in an internal event queue. Client Applications can retrieve events by popping from this queue.

6.12.5 Audit Structures

This section lists ASN.1 structures related to the configuration of the audit service.

Note: A complete ASN.1 definition is available in [GP CSP ASN1].

6.12.5.1 CSPAuditSupport

This data structure represents features, types, and algorithms of the *Audit Module*.

It is included in the CSPEnforce command to detect their support by the platform.

ASN 6-55: Audit: ASN.1 Definition for CSPAuditSupport

```
-- ASN1START
-- Checks for secure auditing support and supported events.
CSPAuditSupport ::= SEQUENCE {

    -- Audit operation modes to specify the handling for unsupported events.
    auditModes          SET OF CSPAuditMode OPTIONAL,

    -- Event types of category "system event".
    systemEvents        SET OF CSPSystemEvent OPTIONAL,

    -- Event types of category "resource event".
    resourceEvents      SET OF CSPResourceEvent OPTIONAL
}
-- ASN1STOP
```

6.12.5.2 CSPAuditSettings

This data structure represents a general *Audit Configuration* for the CSP Instance.

It can be configured using the CSPSetup structure.

It can be retrieved using the CSPConfiguration structure.

For further information, see section 3.11, Secure Auditing, and section 3.11.1, Creating Log Messages.

ASN 6-56: Audit: ASN.1 Definition for CSPAuditSettings

```
-- ASN1START
```

```
-- Audit configuration for this CSP Instance.
CSPAuditSettings ::= SEQUENCE {

    -- Select audit mode to specify the handling of a full audit event queue.
    auditMode                CSAuditMode DEFAULT audit-mode-off,

    -- The system events that shall be audited.
    systemEvents              SET OF CSPSystemEvent OPTIONAL,

    -- The key that shall be used to sign the audit log messages.
    auditSigningKey           CSPResourceId OPTIONAL
}
-- ASN1STOP
```

6.12.5.3 CSAuditMode

This data structure defines the list of available of *Audit Modes*.

An audit mode can be selected using the CSPAuditSettings structure.

The audit types supported can be detected using the CSPAuditSupport structure.

ASN 6-57: Audit: ASN.1 Definition for CSAuditMode

```
-- ASN1START
-- List of available audit operation modes.
CSAuditMode ::= ENUMERATED {

    -- AUDIT_MODE_OFF:
    -- Audit event logging is disabled or not available.
    audit-mode-off           (0),

    -- AUDIT_MODE_OVERWRITE:
    -- events that are not fetched will be overwritte if audit queue is full.
    audit-mode-overwrite     (1),

    -- AUDIT_MODE_STRICT:
    -- CSP will throw an exception if audit event queue is full.
    audit-mode-strict        (2)
}
-- ASN1STOP
```

6.12.5.4 CSPSystemEvent

This data structure defines the list of available *Audit Events* applicable to the CSP Instance.

A system event type can be selected using the CSPAuditSettings structure.

The event types supported can be detected using the CSPAuditSupport structure.

For further information, see section 3.11, Secure Auditing.

ASN 6-58: Audit: ASN.1 Definition for CSPSystemEvent

```
-- ASN1START
-- List of available CSP Instance-specific event types.
CSPSystemEvent ::= ENUMERATED {

    -- EVENT_CSP_START:
    -- Startup of the CSP (0x0001).
    event-csp-start          (1),

    -- EVENT_CSP_UPDATE_STARTED:
    -- Start of a CSP software update (0x0002).
    event-csp-update-started (2),

    -- EVENT_CSP_UPDATE_FINISHED:
    -- CSP software update is finished (0x0003).
    event-csp-update-finished (3),

    -- EVENT_CSP_CONFIG_UPDATED:
    -- The CSP configuration is modified (0x0004).
    event-csp-config-updated (4),

    -- EVENT_CSP_ERROR:
    -- An error occurred that was not yet covered by another event (0x0005).
    event-csp-error          (5),

    -- EVENT_CSP_TIME_SET:
    -- A new reference time is set (0x00B0).
    event-csp-time-set       (176)
}
-- ASN1STOP
```

6.12.5.5 CSPResourceEvent

This data structure defines the list of available *Audit Events* applicable to a resource.

A resource-specific event type can be selected using the CSPResource structure.

The event types supported can be detected using the CSPAuditSupport structure.

For further information, see section 3.11, Secure Auditing.

ASN 6-59: Audit: ASN.1 Definition for CSPResourceEvent

```
-- ASN1START
-- List of available resource-specific event types.
CSPResourceEvent ::= ENUMERATED {

    -- EVENT_RESOURCE_CLEARED:
    -- Resource cleared successfully (0x1000).
```

```
event-resource-cleared                                (4096),

-- EVENT_RESOURCE_VALUE_SET:
-- Resource modified (0x1001).
event-resource-value-set                              (4097),

-- EVENT_CIPHER_ENCRYPTED:
-- Data encrypted successfully (0x1010).
event-cipher-encrypted                               (4112),

-- EVENT_CIPHER_DECRYPTED:
-- Data decrypted successfully (0x1011).
event-cipher-decrypted                               (4113),

-- EVENT_SIGNATURE_CREATED:
-- Signature created successfully (0x1020).
event-signature-created                              (4128),

-- EVENT_SIGNATURE_VERIFIED:
-- Signature verified successfully (0x1021).
event-signature-verified                             (4129),

-- EVENT_SIGNATURE_VERIFICATION_FAILED:
-- Signature verification failed (0x1022).
event-signature-verification-failed                  (4130),

-- EVENT_SECCHANNEL_ESTABLISHED:
-- Secure messaging successfully established (0x1040).
event-secure-channel-established                     (4160),

-- EVENT_SECCHANNEL_AUTHENTICATION_FAILED:
-- Authentication for secure messaging failed (0x1041).
event-secure-channel-authentication-failed           (4161),

-- EVENT_KEY_GENERATED:
-- Key successfully generated (0x1070).
event-key-generated                                  (4208),

-- EVENT_KEY_DERIVED:
-- Key derived successfully (0x1071).
event-key-derived                                    (4209),

-- EVENT_KEY_SHARED_SECRET_COMPUTED:
-- Successful key agreement (0x1072).
event-key-shared-secret-computed                     (4210),
```

```
-- EVENT_PUBLIC_KEY_IMPORTED:
-- Successfully imported a new public key value (0x1073).
event-public-key-imported                (4211),

-- EVENT_CERTIFICATE_IMPORTED:
-- Successfully imported a new certificate (0x1080).
event-certificate-imported                (4224),

-- EVENT_CERTIFICATE_EXPORTED:
-- Successfully exported a certificate (0x1081).
event-certificate-exported                (4225),

-- EVENT_PASSWORD_UPDATED:
-- Password changed successfully (0x1090).
event-password-updated                    (4240),

-- EVENT_PASSWORD_UPDATE_FAILED:
-- Changing a password failed (0x1091).
event-password-update-failed              (4241),

-- EVENT_PASSWORD_AUTHENTICATED:
-- Password verified successfully (0x1092).
event-password-authenticated              (4242),

-- EVENT_PASSWORD_CHECK_FAILED:
-- Password mismatch (0x1093).
event-password-check-failed               (4243),

-- EVENT_PASSWORD_BLOCKED:
-- Password is blocked due to too many incorrect password attempts (0x1094)
event-password-blocked                    (4244),

-- EVENT_PASSWORD_UNBLOCKED:
-- A blocked password was unblocked (0x1095).
event-password-unblocked                  (4245),

-- EVENT_COUNTER_EXHAUSTED:
-- Resource counter exhausted (0x10B0).
event-counter-exhausted                   (4256),

-- EVENT_TIMER_EXPIRED:
-- Resource validity date expired (0x10C0).
event-timer-expired                       (4272),

-- EVENT_OFFLOAD_IMPORTED:
-- Resource imported for offloading (0x10D0).
```

```

event-offload-imported                (4304),

-- EVENT_OFFLOAD_EXPORTED:
-- Resource exported for offloading (0x10D1).
event-offload-exported                (4305)
}
-- ASN1STOP

```

6.12.5.6 CSPLogMessage

This data structure is the result of the `audit.dequeueEvent` operation.

For further information, see section 6.12.1.2, Audit Events.

ASN 6-60: Audit: ASN.1 Definition for CSPLogMessage

```

-- ASN1START
-- Log message format, computed by the audit.dequeueEvent operation.
CSPLogMessage ::= SEQUENCE {

    -- Event type; value can be taken from CSPSystemEvent or CSPResourceEvent.
    eventType                INTEGER (0..65535),

    -- Additional Fields to be included; configured by the CSP Admin.
    fieldsAddedAsPrefix      SET OF CSPField OPTIONAL,

    -- Event-specific log data.
    eventData                CSPEventData OPTIONAL,

    -- Additional input data provided by the Client through audit.dequeueEvent.
    inputData                OCTET STRING (SIZE(0..65536)) OPTIONAL,

    -- Additional Fields to be included; configured by the CSP Admin.
    fieldsAddedAsSuffix      SET OF CSPField OPTIONAL
}
-- ASN1STOP

```

6.12.5.7 CSPEventData

This data structure represents event-specific log data.

It is part of the CSPLogMessage structure.

ASN 6-61: Audit: ASN.1 Definition for CSPEventData

```

-- ASN1START
-- Event-specific log data; this structure is part of a log message.
CSPEventData ::= CHOICE {

    -- Event-specific data for: CSP software update started or has finished.

```

```

updateCSPEvent      CSPEventDataUpdateCSP,

-- Event-specific data for: Config updated successful.
updateConfigEvent   CSPEventDataUpdateConfig,

-- Event-specific data for: Reference time updated.
setTimeEvent        CSPEventDataSetTime,

-- Event-specific data for: Reference time updated.
generalErrorEvent   CSPEventDataGeneralError,

-- Event-specific data for: General resource-specific events.
generalResourceEvent CSPEventDataResource,

-- Event-specific data for: Successful key derivation.
keyDerivationEvent  CSPEventDataKeyDerivation,

-- Event-specific data for: Shared secret computed successfully.
keyAgreementEvent   CSPEventDataKeyAgreement,

-- Event-specific data for: Password verification failed.
passwordCheckFailedEvent CSPEventDataPasswordFailure
}
-- ASN1STOP

```

6.12.5.8 CSPEventDataUpdateCSP

This event data is generated for the following events:

- EVENT_CSP_UPDATE_STARTED
- EVENT_CSP_UPDATE_FINISHED

It is part of the CSPEventData structure.

ASN 6-62: Audit: ASN.1 Definition for CSPEventDataUpdateCSP

```

-- ASN1START
-- Event-specific data for 'CSP software update started or has finished'.
CSPEventDataUpdateCSP ::= SEQUENCE {

    -- The cspELFVersion of the CSP ELF of the before the SW update.
    oldCSPELFVersion      CSPELFVersion,

    -- The cspELFVersion of the CSP ELF after the SW update.
    newCSPELFVersion      CSPELFVersion,

    -- The version of the CSP Protocol before the SW update.
    cspProtocolVersion     CSPProtocolVersion,

```

```
-- The cspProtocolVersion of the CSP Protocol after the SW update.
newProtocolVersion    CSPProtocolVersion
}
-- ASN1STOP
```

6.12.5.9 CSPEventDataUpdateConfig

This event data is generated for the following events:

- EVENT_CSP_CONFIG_UPDATED

It is part of the CSPEventData structure.

For further information, see section 5.1.1.4, System Events.

ASN 6-63: Audit: ASN.1 Definition for CSPEventDataUpdateConfig

```
-- ASN1START
-- Event-specific data for EVENT_CSP_CONFIG_UPDATED: Config updated.
CSPEventDataUpdateConfig ::= SEQUENCE {

    -- Custom name or identifier set by the CSP Admin via CSPSetup.
    configName          CSPConfigName,

    -- The old configVersion of the CSP Configuration before the update.
    oldConfigVersion    CSPConfigVersion,

    -- The new configVersion of the CSP Configuration after the update.
    newConfigVersion    CSPConfigVersion
}
-- ASN1STOP
```

6.12.5.10 CSPEventDataSetTime

This event data is generated for the following events:

- EVENT_CSP_TIME_SET

It is part of the CSPEventData structure.

For further information, see section 6.11.1.7, Time Events.

ASN 6-64: Audit: ASN.1 Definition for CSPEventDataSetTime

```
-- ASN1START
-- Event-specific data for EVENT_CSP_TIME_SET: Reference time updated.
CSPEventDataSetTime ::= SEQUENCE {

    -- The old referenceTime as it was before the update.
    oldReferenceTime    CSPTimestamp,

    -- The new referenceTime after the update.
    newReferenceTime    CSPTimestamp
}
```

```
}
-- ASN1STOP
```

6.12.5.11 CSPEventDataGeneralError

This event data is generated for the following events:

- EVENT_CSP_ERROR

It is part of the CSPEventData structure.

For further information, see section 5.1.1.4, System Events, and section 5.1.1.6, Error Types.

ASN 6-65: Audit: ASN.1 Definition for CSPEventDataGeneralError

```
-- ASN1START
-- Event-specific data for EVENT_CSP_ERROR: An exception occurred.
CSPEventDataGeneralError ::= SEQUENCE {

    -- Contains the reason of the exception that occurred.
    reason                OCTET STRING (SIZE(2)),

    -- A resource ID involved to the error (if available).
    resourceId            CSPResourceId OPTIONAL
}
-- ASN1STOP
```

6.12.5.12 CSPEventDataResource

This event data is generated for the following events:

- EVENT_CIPHER_DECRYPTED, EVENT_CIPHER_ENCRYPTED
- EVENT_SIGNATURE_CREATED, EVENT_SIGNATURE_VERIFIED
- EVENT_SIGNATURE_VERIFICATION_FAILED
- EVENT_SECCHANNEL_ESTABLISHED, EVENT_SECCHANNEL_AUTHENTICATION_FAILED
- EVENT_KEY_GENERATED, EVENT_PUBLIC_KEY_IMPORTED, EVENT_CERTIFICATE_IMPORTED
- EVENT_CERTIFICATE_IMPORTED, EVENT_CERTIFICATE_EXPORTED
- EVENT_PASSWORD_UPDATED, EVENT_PASSWORD_UPDATE_FAILED, EVENT_PASSWORD_AUTHENTICATED
- EVENT_PASSWORD_BLOCKED, EVENT_PASSWORD_UNBLOCKED
- EVENT_COUNTER_EXHAUSTED, EVENT_TIMER_EXPIRED
- EVENT_RESOURCE_CLEARED, EVENT_RESOURCE_VALUE_SET
- EVENT_OFFLOAD_IMPORTED, EVENT_OFFLOAD_EXPORTED

It is part of the CSPEventData structure.

ASN 6-66: Audit: ASN.1 Definition for CSPEventDataResource

```
-- ASN1START
-- Event-specific data for general resource-specific events.
```

```

CSPEventDataResource ::= SEQUENCE {

    -- The identifier of the resource that triggered this event.
    resourceId          CSPResourceId
}
-- ASN1STOP

```

6.12.5.13 CSPEventDataKeyDerivation

This event data is generated for the following events:

- EVENT_KEY_DERIVED

It is part of the CSPEventData structure.

ASN 6-67: Audit: ASN.1 Definition for CSPEventDataKeyDerivation

```

-- ASN1START
-- Event-specific data for EVENT_KEY_DERIVED: Successful key derivation.
CSPEventDataKeyDerivation ::= SEQUENCE {

    -- The source resource for key derivation.
    sourceResourceId      CSPResourceId,

    -- The target resource where the result is stored.
    destResourceId        CSPResourceId
}
-- ASN1STOP

```

6.12.5.14 CSPEventDataKeyAgreement

This event data is generated for the following events:

- EVENT_KEY_SHARED_SECRET_COMPUTED

It is part of the CSPEventData structure.

ASN 6-68: Audit: ASN.1 Definition for CSPEventDataKeyAgreement

```

-- ASN1START
-- Event-specific data for EVENT_KEY_SHARED_SECRET_COMPUTED: Key agreement.
CSPEventDataKeyAgreement ::= SEQUENCE {

    -- The local or remote private key.
    privateKeyId          CSPResourceId,

    -- The local or remote public key.
    publicKeyId           CSPResourceId,

    -- The resource ID of the destination shared secret.
    sharedSecretId        CSPResourceId
}

```

```
-- ASN1STOP
```

6.12.5.15 CSPEventDataPasswordFailure

This event data is generated for the following events:

- EVENT_PASSWORD_CHECK_FAILED

It is part of the CSPEventData structure.

ASN 6-69: Audit: ASN.1 Definition for CSPEventDataPasswordFailure

```
-- ASN1START
-- Event-specific data for EVENT_PASSWORD_CHECK_FAILED: Verification failed.
CSPEventDataPasswordFailure ::= SEQUENCE {

    -- The identifier of the resource that triggered this event.
    resourceId          CSPResourceId,

    -- The remaining try counter value.
    tryCounter          INTEGER (0..255)
}
-- ASN1STOP
```

6.13 Offloading Module

The CSP may implement the OffloadingModule to support *Streaming Resources for Offloading*; the choice shall be subject to *Modularity*.

The CSP Admin shall be able to create the resources specific for offloading by:

- Creating the resources to be exported or imported as CSPResource with ACCESS_MOVE permission.
- Creating the offloading key as CSPResource with USAGE_CIPHER.
- Configuring the paddingAlgorithm and cipherAlgorithm to the offloading key.

The Client Application shall be able to pass the resourceId referring to these resources to the offloading.initManage operation and initiate:

- Export or import streaming processes using the offloading.manage operation.
- Multi-part streaming processes using the offloading.updateManage operation.

The CSP Admin may detect whether the platform supports this module using the CSPEnforce command.

For further information, see section 3.1.6, Offloading Resources.

6.13.1 Offloading Definitions

6.13.1.1 Offloading Modes

The OffloadingModule shall support resource streaming modes listed in Table 6-1. This mode defines if the offloading service operates in import or export mode.

The Client Application may select the offloadingMode using the offloading.initManage operation.

Note: These modes function similarly to *Key Management Modes* and *Certificate Management Modes*.

Table 6-128: Key Management Modes

Mode	Value (Hex)	Value (Int)	Description
MANAGE_MODE_OFFLOAD_IMPORT	0x05	5	Streamed resource import. The OffloadingModule shall import the complete resource, including its value and parameters, using data from the input buffer via offloading.manage OR offloading.updateManage.
MANAGE_MODE_OFFLOAD_EXPORT	0x06	6	Streamed resource export. The OffloadingModule shall write the entire resource as a stream to the output buffer via offloading.manage OR offloading.updateManage.

6.13.1.2 Offloading Events

The AuditModule may log the offloading-related *Audit Events* listed in Table 6-112; the exact list shall be subject to *Modularity*. The event-specific data listed in the table shall be stored according to the configured auditMode.

The CSP Admin may configure resourceEvents to be logged using the CSPResourceEvent structure and systemEvents to be logged using the CSPSystemEvent structure.

The CSP Admin may detect if an event type is supported using the CSPAuditSupport structure.

For further information, see section 3.11.4, Selecting Audit Events, and section 6.12.1.2, Audit Events.

Table 6-129: Offloading Events for Auditing

Event Type	Value (Hex)	Value (Int)	Description	Event Category
EVENT_OFFLOAD_IMPORTED	0x10D0	4304	Resource imported successfully. This event is logged when a resource has been successfully imported. Technical Details: <ul style="list-style-type: none"> Message Format: CSPLogMessage Operations: offloading.manage (IMPORT) Event Data: CSPEventDataResource 	Resource Event
EVENT_OFFLOAD_EXPORTED	0x10D1	4305	Resource exported successfully. This event is logged when a resource has been successfully exported. Technical Details: <ul style="list-style-type: none"> Message Format: CSPEventDataResource Operations: offloading.manage (EXPORT) Event Data: CSPEventDataResource 	Resource Event

6.13.1.3 Offloading Error Codes

If ERROR_MODE_DETAILED is supported and activated, the OffloadingModule shall use the error codes listed in Table 6-130 as reasons for exceptions.

For further information, see section 5.1.1.6, Error types.

Table 6-130: Offloading Error Reason Codes

Reason	Description
0x50D0	Wrong usage: Resource not configured for USAGE_OFFLOADING.
0x50D1	Export not allowed: Resource has an active usage counter.
0x60D0	Illegal import data: The data provided is not compatible to the resource it shall be imported to.
0x60D1	Missing validity date: Resource has a validity period, but no validity date is provided in the import data.
0x80D0	Unsupported: The offloading module is not supported.

6.13.2 Offloading Configuration

The OffloadingModule shall support the configuration parameters defined in Table 6-133.

Table 6-131: Offloading Configuration Parameters

Parameter	Description	Parameter Type
offloadingMode	Operation mode of the OffloadingModule, either resource streaming import or export mode, selected from the available <i>Offloading Modes</i> . Technical Details: <ul style="list-style-type: none"> Data type: byte Operations: offloading.initManage 	Service Parameter

6.13.3 Offloading Operations

The OffloadingModule shall implement the operations listed in Table 6-132.

For each operation, the CSP shall handle supported *Counter Types*, *Timer Types*, and *Audit Events*.

These operations shall solely be utilizable with resources configured for USAGE_OFFLOADING as encryption keys and shall comply with the *Access Restrictions on Offloading Operations* specified in Table 6-133.

Note: Specifically, resources with USAGE_OFFLOADING, shall not be permitted for decryption using *Cipher Operations*, *Transform Operations*, or *Confidential Data Transfer Operations*.

For further information, see section 3.1.6, Offloading Resources.

Table 6-132: Offloading Operations

Operation	Description
offloading.initManage	<p>Initializes this service for <i>Streaming Resources</i> for Offloading by setting the offloadingMode from the available <i>Offloading Modes</i> and setting the resource to im- or export.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.OffloadingService.initManage(..) <p>Evaluate Timers (for the offloading key):</p> <ul style="list-style-type: none"> TIMER_VALIDITY_DATE TIMER_VALIDITY_PERIOD <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_TIMER_EXPIRED, EVENT_CSP_ERROR
offloading.manage	<p>Depending on the selected offloadingMode, either writes the resource as encrypted stream to the provided output buffer or imports the provided data to the resource.</p> <p>Note: Timer values are part of the stream without being reset.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.OffloadingService.manage(..) <p>CSP Protocol:</p> <ul style="list-style-type: none"> CSPSetValue <p>Increment Counters (for the oflloading key only):</p> <ul style="list-style-type: none"> COUNT_USAGE, COUNT_USAGE_PER_BLOCK COUNT_USAGE_SUCCESS_ONLY, COUNT_USAGE_FAILURE_ONLY <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_OFFLOAD_IMPORTED, EVENT_OFFLOAD_EXPORTED EVENT_COUNTER_EXHAUSTED, EVENT_CSP_ERROR
offloading.updateManage	<p>Optional method to handle multi-part resource stream import or export by processing a chunk of data. The method can be invoked multiple times for sequential data processing and must be concluded with offloading.manage.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.OffloadingService.updateManage(..) <p>Increment Counters:</p> <ul style="list-style-type: none"> COUNT_USAGE_PER_BLOCK <p>Fire Events:</p> <ul style="list-style-type: none"> EVENT_COUNTER_EXHAUSTED, EVENT_CSP_ERROR
offloading.getManagedLength	<p>Retrieve the size, in bytes, of the buffer required for importing or exporting the resource.</p> <p>CSP API:</p> <ul style="list-style-type: none"> org.globalplatform.csp.api.OffloadingService.getManagedLength(..)

6.13.3.1 Access Restrictions on Offloading Operations

The OffloadingModule shall enforce the access restrictions for *Offloading Operations* listed in Table 6-133 for all *Access Rights*, *Usage Types*, *Resource States*, and *Policy Types* supported by the platform. Unauthorized or illegal operation invocations shall be rejected and handled according to the *errorMode* configured.

Table 6-133: Offloading Operations Access Restrictions

Operation	Right Required	Usage Required	State Required	Other Restrictions
offloading.initManage (resource to EXPORT)	MOVE	-	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required Resources with active usage counter cannot be used for offloading. POLICY_PASSWORD POLICY_SECCHANNEL_ESTABLISHED
offloading.initManage (resource to IMPORT)	MOVE	-	UNINITIALIZED	
offloading.initManage (offloading key)	USE	-	OPERATIONAL	<ul style="list-style-type: none"> Client Authentication required
offloading.updateManage offloading.manage (resource to EXPORT)	-	-	OPERATIONAL	
offloading.updateManage offloading.manage (resource to IMPORT)	-	-	UNINITIALIZED	
offloading.updateManage offloading.manage (offloading key)	-	-	OPERATIONAL	
offloading.getManagedLength	-	-	OPERATIONAL	

6.13.3.2 Sensitive Results Computed by Offloading Operations

The OffloadingModule shall temporarily store the results of methods listed in Table 5-22 in transient memory and shall implement the `assertSensitiveResult` operation, enabling Client Applications to verify that these results were computed without any abnormality.

Table 6-134: Offloading Operations Requiring Sensitive Results Checks

Operation	Mode	Result Type	Result Description
offloading.manage offloading.updateManage	MANAGE_MODE_OFFLOAD_EXPORT	short	The length of the encrypted exported resource data.

6.13.3.3 Sensitive Arrays Required for Offloading Operations

The OffloadingModule shall invoke `javacard.framework.SensitiveArrays.assertIntegrity(..)` ([JCAPI]) on the parameters listed in Table 6-135.

For further information, see section 5.1.3.4, Sensitive Arrays, and section 8.3, Byte Buffer Parameters.

Table 6-135: Offloading Operations Requiring Sensitive Arrays

Operation	Parameter	Parameter Description
offloading.initManage	Input buffer	Algorithm-specific initialization data.

6.13.4 Offloading Lifecycle

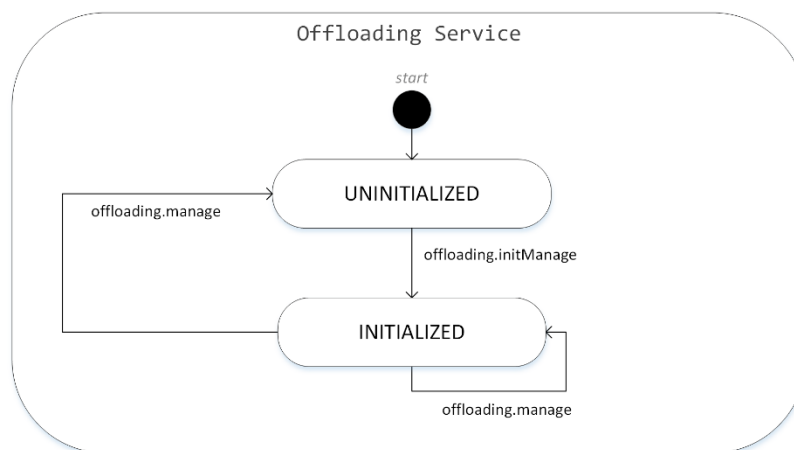
Figure 6-6 shows lifecycle changes of an offloading service triggered by *Offloading Operations*. An offloading service is an instance of the OffloadingModule. When a new instance is created, it starts in the UNINITIALIZED state. The Client Application needs to invoke the offloading.initManage operation to configure the offloadingMode along with the resources required. After completing the offloading.manage operation, the offloading service will return to the UNINITIALIZED state.

The offloading service stores *Sensitive Results Computed by Offloading Operations*.

The offloading service triggers lifecycle changes to the encryption key resource used, as specified in

- Section 6.7.4 Key Lifecycle

Figure 6-12: Offloading Lifecycle



6.14 Field Module

The CSP may implement the FieldModule to offer *Signatures with Counters & Timestamps*; the choice shall be subject to *Modularity*.

The CSP Admin shall be able to configure the fields that are included in attestation results and log messages, using the CSPAttestationAlgorithms structure to:

- Configure *Fields* to the auditSigningKey with USAGE_AUDIT.
- Configure *Fields* to attestation keys with USAGE_ATTESTATION.

The CSP shall include the data referenced by the fields, such as the current timestamp, resource counter, or resource state, in the attestation results provided to Client Applications through att.computeAttestation and in log messages generated via audit.dequeueEvent.

The CSP Admin may detect whether the platform supports this module using the CSPEnforce command.

For further information, see section 3.6.7, Signatures with Counters & Timestamps.

6.14.1 Field Definitions

6.14.1.1 Fields

The FieldModule may support the fields listed in Table 6-136; the exact list shall be subject to *Modularity*. The FieldModule shall include fields in attestation data and log messages before they are signed by the CSP.

The CSP Admin may configure fieldsAddedAsPrefix and fieldsAddedAsSuffix as ordered lists of CSPFieldType using the CSPAttestationAlgorithms structure. Optionally, a resourceId can be assigned to each CSPFieldType to source its data from a different resource.

The CSP Admin may detect if a field is supported using the CSPFieldSupport structure.

Note: Fields configured to the auditSigningKey shall be stored in persistent memory within the audit event queue each time a new event occurs, unless otherwise specified in Table 6-136.

For further information, see section 3.6.7, Signatures with Counters & Timestamps, and section 3.11.1, Create Log Messages.

Table 6-136: Fields

Field	Value (Hex)	Value (Int)	Description
FIELD_SYSTEM_TIME	0x01	1	<p>Adds the estimated <code>systemTime</code> as Unix timestamp in seconds. Shall be stored within the <i>Audit Event Queue</i>.</p> <p>Note: Availability of this field depends on the <i>Time Modes</i> supported.</p> <ul style="list-style-type: none"> • Data Format: CSPTimestamp • Size: 8 bytes
FIELD_TIME_SINCE_BOOT	0x02	2	<p>Adds the elapsed <code>timeSinceBoot</code> boot as Unix duration in seconds. Shall be stored within the <i>Audit Event Queue</i>.</p> <p>Note: Availability of this field depends on the <i>Time Modes</i> supported.</p> <ul style="list-style-type: none"> • Data Format: CSPDuration • Size: 4 bytes

Field	Value (Hex)	Value (Int)	Description
FIELD_REFERENCE_TIME	0x03	3	<p>Adds the referenceTime as Unix timestamp in seconds.</p> <p>Shall be stored within the <i>Audit Event Queue</i>.</p> <p>Note: Availability of this field depends on the <i>Time Modes</i> supported.</p> <ul style="list-style-type: none"> Data Format: CSPTimestamp Size: 8 bytes
FIELD_USAGE_COUNTER	0x04	4	<p>Adds the counterValue of a counter of type COUNT_USAGE to the data before the counter is incremented.</p> <p>Shall not be stored within the <i>Audit Event Queue</i> when referring to the usage counter of the auditSigningKey. Other usage counters shall be stored in the <i>Audit Event Queue</i>.</p> <p>Note: Availability of this field depends on the <i>Counter Modes</i> supported.</p> <ul style="list-style-type: none"> Data Format: byte[] Size: 2-4 byte (dependent on the counterCapacity)
FIELD_CSP_CONFIG_VERSION	0x05	5	<p>Adds the current configVersion of the CSP configuration as signed short to the message before it is signed. This version can be set by the CSP Admin through CSPSetup.</p> <p>Shall be stored within the <i>Audit Event Queue</i>.</p> <ul style="list-style-type: none"> Data Format: CSPConfigVersion Size: 2 byte
FIELD_CSP_PROTOCOL_VERSION	0x06	6	<p>Adds the current cspProtocolVersion of the CSP platform to the data before it is signed.</p> <p>May be stored within the <i>Audit Event Queue</i>.</p> <ul style="list-style-type: none"> Data Format: CSPProtocolVersion Size: 1 byte
FIELD_CSP_ELF_VERSION	0x07	7	<p>Adds the current cspELFVersion of the CSP Application to the data before it is signed.</p> <p>May be stored within the <i>Audit Event Queue</i>.</p> <ul style="list-style-type: none"> Data Format: CSPELFVersion Size: 2 byte
FIELD_RESOURCE_STATE	0x08	8	<p>Adds the resourceState the data before it is signed.</p> <p>Shall be stored within the <i>Audit Event Queue</i>.</p> <ul style="list-style-type: none"> Data Format: CSPResourceState Size: 1 byte
FIELD_PUBLIC_KEY	0x09	9	<p>Adds a public key value to the data before it is signed.</p> <p>May be stored within the <i>Audit Event Queue</i>.</p> <ul style="list-style-type: none"> Data Format: byte[] Size: n byte
FIELD_MANUAL_COUNTER	0x0A	10	<p>Adds the counterValue of a resource type RESOURCE_COUNTER to the data before it is signed.</p> <p>Shall be stored within the <i>Audit Event Queue</i>.</p> <p>Note: Availability of this field depends on the <i>Counter Modes</i> supported.</p> <ul style="list-style-type: none"> Data Format: byte[] Size: 2-4 byte

Field	Value (Hex)	Value (Int)	Description
FIELD_MANUAL_COUNTER_LIMIT	0x0B	11	<p>Adds the counterLimit of a resource of type RESOURCE_COUNTER to the data before it is signed.</p> <p>May be stored within the <i>Audit Event Queue</i>.</p> <p>Note: Availability of this field depends on the <i>Counter Modes</i> supported.</p> <ul style="list-style-type: none"> Data Format: byte[] Size: 2-4 byte
FIELD_MANUAL_TIMER	0x0C	12	<p>Adds the timerValue of a resource of type RESOURCE_TIMER to the data before it is signed.</p> <p>Shall be stored within the <i>Audit Event Queue</i>.</p> <p>Note: Availability of this field depends on the <i>Time Modes</i> supported.</p> <ul style="list-style-type: none"> Data Format: byte[] Size: 2-8 byte
FIELD_MANUAL_TIMER_LIMIT	0x0D	13	<p>Adds the timeLimit of a resource of type RESOURCE_TIMER to the data before it is signed or the counter is incremented.</p> <p>May be stored within the <i>Audit Event Queue</i>.</p> <p>Note: Availability of this field depends on the <i>Time Modes</i> supported.</p> <ul style="list-style-type: none"> Data Format: byte[] Size: 2-8 byte

6.14.1.2 Field Sources

The FieldModule shall support the field sources listed in Table 6-137. The field source specifies the source from which a field retrieves its data.

The CSP Admin may select the fieldSource using the CSPFieldMode structure.

Table 6-137: Field Sources

Field Source	Value (Hex)	Value (Int)	Description
FIELD_SOURCE_DEFAULT	0x00	0	<p>For this source, fields requiring data from a CSP Resource (e.g., FIELD_RESOURCE_STATE) shall use the signing key to which the field is configured through the CSPAttestationAlgorithms.</p> <p>Fields not dependent on resources shall use CSP Instance-specific system data.</p> <p>This is the standard source if no field source is specified.</p>
FIELD_SOURCE_EVENT	0x01	1	<p>For this source, fields requiring data from a CSP Resource (e.g., FIELD_RESOURCE_STATE) shall use the resource that triggered the event.</p> <p>This source can only be used for fields configured to the auditSigningKey.</p>

6.14.1.3 Field Modes

The FieldModule may support the field modes listed in Table 6-138; the exact list shall be subject to *Modularity*. This mode defines how the CSP shall handle scenarios where configured *Fields* are not supported by the platform.

The CSP Admin may select the fieldMode using the CSPFieldMode structure.

The CSP Admin may detect if a field mode is supported using the CSPCoreSupport structure.

Table 6-138: Field Modes

Field Mode	Value (Hex)	Value (Int)	Description
FIELD_MODE_OFF	0x00	0	Fields are disabled and not available. Regardless of whether any field is configured, the <code>FieldModule</code> shall not add them to any attestation data or log message and shall operate without errors. This is the standard mode when fields are not supported by the platform. Note: This mode may be used to temporarily disable fields handling even if fields are supported by the platform.
FIELD_MODE_IGNORE	0x01	1	Ignore fields configurations not supported by the platform. The <code>FieldModule</code> shall add fields to the data before it is signed for all fields supported by the platform. For <i>Fields</i> not supported, the CSP shall ignore the specific field.
FIELD_MODE_STRICT	0x02	2	Stop operation if a field is not supported by the platform. The <code>FieldModule</code> shall handle all fields configured by the CSP Admin. For unsupported <i>Fields</i> , the CSP shall throw an <code>ERROR_NOT_SUPPORTED</code> [0x80E0] exception when processing the affected field.

6.14.1.4 Field Error Codes

If `ERROR_MODE_DETAILED` is supported and activated, the `FieldModule` shall use the error codes listed in Table 6-139 as reasons for exceptions.

For further information, see section 5.1.1.6, Error Types.

Table 6-139: Field Error Reason Codes

Reason	Description
0x30E0	Invalid field config: A resource ID is either missing (but required by a field) or provided (but not required).
0x80E0	Unsupported: The field is not supported and <code>FIELD_MODE_STRICT</code> is set.

6.14.2 Field Configuration

The `FieldModule` shall support the configuration parameters defined in Table 6-140.

The CSP Admin may retrieve the `CSPConfiguration` using the `CSPGetConfiguration` command.

Table 6-140: CSP Configuration Parameters

Parameter	Description	Parameter Type
fieldsAddedAsPrefix	<p>Fields added to the beginning of the attestation data or log messages before signed. These fields are configured for an attestation key or for the auditSigningKey, selected from available <i>Fields</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Data type: CSPField Administration: CSPAttestationAlgorithms <p>Included In:</p> <ul style="list-style-type: none"> CSPLogMessage CSPConfigAttestation CSPDataAttestation CSPKeyPoPAttestation 	Resource Parameter
fieldsAddedAsSuffix	<p>Fields added to the beginning of the attestation data or log messages before signed. These fields are configured for an attestation key or for the auditSigningKey, selected from available <i>Fields</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Data type: CSPField Administration: CSPAttestationAlgorithms <p>Included In:</p> <ul style="list-style-type: none"> CSPLogMessage CSPConfigAttestation CSPDataAttestation CSPKeyPoPAttestation 	Resource Parameter
fieldSource	<p>Specifies the source from which a field retrieves its data (e.g., the signing key or the resource that triggered the event), selected from available <i>Field Sources</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Data type: CSPFieldSource Administration: CSPField 	Field Parameter
fieldMode	<p>Defines how the CSP shall handle scenarios where a configured <i>Fields</i> is not supported by the platform. It is selected from the available <i>Field Modes</i>.</p> <p>Technical Details:</p> <ul style="list-style-type: none"> Data type: CSPFieldMode Administration: CSPFieldSettings 	CSP Instance Parameter

6.14.3 Field Operations

The FieldModule does not offer operations. It is integrated into the *Attestation Module* and *Audit Module*.

6.14.4 Field Lifecycle

Fields do not have a lifecycle state.

6.14.5 Field Structures

This section lists ASN.1 structures related to the configuration of signature fields.

Note: A complete ASN.1 definition is available in [GP CSP ASN1].

6.14.5.1 CSPFieldSupport

This data structure represents features and types of the *Field Module*.

It is included in the CSPEnforce command to detect their support by the platform.

ASN 6-70: Access: ASN.1 Definition for CSPFieldSupport

```
-- ASN1START
-- Checks for signing field functionality support.
CSPFieldSupport ::= SEQUENCE {

    -- CSP handling modes of how to handle signature fields.
    fieldModes          SET OF CSPFieldMode OPTIONAL,

    -- Signature fields to be added to log messages and attestation data.
    fieldTypes          SET OF CSPFieldType OPTIONAL,

    -- Available data sources for fields.
    fieldSources        SET OF CSPFieldSource OPTIONAL
}
-- ASN1STOP
```

6.14.5.2 CSPFieldSettings

This data structure represents a general *Field Configuration* for the CSP Instance.

It can be configured using the CSPSetup structure.

It can be retrieved using the CSPConfiguration structure.

ASN 6-71: Audit: ASN.1 Definition for CSPFieldSettings

```
-- ASN1START
-- Field configuration for this CSP Instance.
CSPFieldSettings ::= SEQUENCE {

    -- Select the mode for handling unsupported fields.
    fieldMode           CSPFieldMode DEFAULT field-mode-off
}
-- ASN1STOP
```

6.14.5.3 CSPFieldMode

This data structure defines the list of available *Field Modes*.

The fieldMode can be selected using the CSPFieldSettings structure.

The field modes supported can be detected using the CSPFieldSupport structure.

ASN 6-72: Signature: ASN.1 Definition for CSPFieldMode

```
-- ASN1START
-- List of modes how the CSP shall handle signature fields.
```

```
CSPFieldMode ::= ENUMERATED {

    -- FIELD_MODE_OFF:
    -- Fields are disabled or not available.
    field-mode-off          (0),

    -- FIELD_MODE_IGNORE:
    -- Ignore field configurations that are not supported by the platform.
    field-mode-ignore       (1),

    -- FIELD_MODE_STRICT:
    -- Stop operation if a field configured is not supported.
    field-mode-strict       (2)
}
-- ASN1STOP
```

6.14.5.4 CSPField

This data structure is used to configure *Fields* for inclusion in attestation data or log messages. It can be configured using the CSPAttestationAlgorithms structure.

ASN 6-73: Signature: ASN.1 Definition for CSPField

```
-- ASN1START
-- Structure to configure a field for attestation data or log messages.
CSPField ::= SEQUENCE {

    -- Type of the data field.
    field          CSPFieldType,

    -- Source from which the data is taken for the field.
    source         CHOICE {

        -- Type of the source, e.g., signing key or event trigger resource.
        fieldSource CSPFieldSource,

        -- A specific resource used as the data source.
        resourceId  CSPResourceId

    } OPTIONAL
}
-- ASN1STOP
```

6.14.5.5 CSPFieldType

This data structure defines the list of available *Fields*.

A field type can be selected using the CSPField structure.

The field types supported can be detected using the CSPFieldSupport structure.

ASN 6-74: Signature: ASN.1 Definition for CSPFieldType

```
-- ASN1START
-- List of available field types that can be included in attestations and logs.
CSPFieldType ::= ENUMERATED {

    -- FIELD_SYSTEM_TIME:
    -- Estimated system time as Unix timestamp in seconds, 8 bytes.
    field-system-time                (1),

    -- FIELD_TIME_SINCE_BOOT:
    -- The time since boot in seconds, 4 bytes.
    field-time-since-boot            (2),

    -- FIELD_REFERENCE_TIME:
    -- The reference time set by CSP Admin or CSP Client.
    field-reference-time              (3),

    -- FIELD_USAGE_COUNTER:
    -- The value of a resource usage counter.
    field-usage-counter               (4),

    -- FIELD_CSP_CONFIG_VERSION:
    -- The version of the CSP Configuration set by the CSP Admin.
    field-csp-config-version          (5),

    -- FIELD_CSP_PROTOCOL_VERSION:
    -- The version of the CSP Protocol of the platform.
    field-csp-protocol-version        (6),

    -- FIELD_CSP_ELF_VERSION:
    -- The version of the CSP ELF.
    field-csp-elf-version             (7),

    -- FIELD_RESOURCE_STATE:
    -- The resource state.
    field-resource-state              (8),

    -- FIELD_PUBLIC_KEY:
    -- The value of a public key.
    field-pubkey                      (9),

    -- FIELD_MANUAL_COUNTER:
    -- The value of a manual counter.
```

```

field-manual-counter          (10),

-- FIELD_MANUAL_COUNTER_LIMIT:
-- The limit set to a manual counter.
field-manual-counter-limit    (11),

-- FIELD_MANUAL_TIMER:
-- The value of a manual timer.
field-manual-timer            (12),

-- FIELD_MANUAL_TIMER_LIMIT:
-- The limit of a manual timer.
field-manual-timer-limit      (13)
}
-- ASN1STOP

```

6.14.5.6 CSPFieldSource

This data structure specifies the source from which a field retrieves its data.

A field source can be selected using the CSPField structure.

ASN 6-75: Signature: ASN.1 Definition for CSPFieldSource

```

-- ASN1START
-- List of available field sources where the field gets its data from.
CSPFieldSource ::= ENUMERATED {

    -- FIELD_SOURCE_DEFAULT:
    -- System data or if a resource is required the signing key resource.
    default-source          (0),

    -- FIELD_SOURCE_EVENT:
    -- The resource that triggered the audit event.
    event-source            (1)
}
-- ASN1STOP

```

6.14.5.7 CSPFieldValue

This data structure represents the concrete field value.

It is part of CSPLogMessage, CSPConfigAttestation, CSPKeyPoPAttestation, and CSPDataAttestation.

The corresponding field is configured through the CSPAttestationAlgorithms structure.

ASN 6-76: Signature: ASN.1 Definition for CSPFieldValue

```

-- ASN1START
-- The concrete value of a data field added to data before it is signed.
CSPFieldValue ::= SEQUENCE {

```

```
-- Type of the data field.  
field          CSPFieldType,  
  
-- The resource used as the data source.  
resourceId     CSPResourceId OPTIONAL,  
  
-- The value of the field.  
fieldValue     OCTET STRING (SIZE(0..65536))  
}  
-- ASN1STOP
```

6.15 Policy Module

The CSP may implement the PolicyModule to evaluate *Policies*; the choice shall be subject to *Modularity*.

The CSP Admin shall be able to configure policies to resources by using the CSPAccessControl structure:

The CSP Admin may detect whether the platform supports this module using the CSPEnforce command.

For further information, see section 3.2.6, Policies.

6.15.1 Policy Definitions

6.15.1.1 Policy Modes

The PolicyModule may support the policies modes listed in Table 6-141; the exact list shall be subject to *Modularity*. The selected policy mode defines how the CSP shall handle scenarios where a policy configured is supported by the platform.

The CSP Admin may select the policyMode using the CSPPolicyMode structure.

The CSP Admin may detect if a policy mode is supported using the CSPCoreSupport structure.

Table 6-141: Policy Modes

Mode	Value (Hex)	Value (Int)	Description
POLICY_MODE_OFF	0x00	0	Policy evaluation is disabled and not available. Regardless of whether any policy is configured, the CSP shall not evaluate them and shall operate without errors. This is the standard mode when policies are not supported by the platform. Note: This mode may be used to temporarily disable policy handling even if policies are supported by the platform.
POLICY_MODE_IGNORE_UNSUPPORTED	0x01	1	Ignore policy configurations not supported by the platform. The CSP shall evaluate policies for all policy types supported by the platform. For <i>Policy Types</i> not supported, the CSP shall ignore the specific policy configuration.
POLICY_MODE_STRICT	0x02	2	Stop operation if a policy type is not supported by the platform. The CSP shall handle all policies configured by the CSP Admin. For unsupported <i>Policy Types</i> , the CSP shall throw an ERROR_NOT_SUPPORTED [0x8004] exception when processing the affected policy.

6.15.1.2 Policy Types

The PolicyModule may support the policy types listed in Table 6-142; the exact list shall be subject to *Modularity*. Policy rules efficiently remove the necessity for security-related if-conditions within the Client Application, as the CSP internally assesses these conditions.

The CSP Admin may configure policies for a resource using the CSPPolicyType structure. The owner resource is the resource the policy is configured for. Each policy contains:

- A policy type that specifies the scenario to be evaluated.

- A constraining resource that is evaluated according to the policy type.
- Additional input data specific to the policy type.

The CSP shall evaluate policies when their owner resource is used in service operations that support the respective policy type. If a constraining resource is used in other operations, its associated policies shall not be evaluated by the CSP.

The CSP Admin may detect if a policy type is supported using the CSPCoreSupport structure. Each policy supported shall adhere to the combinations specified in:

- Table 6-143 for compatible *Policy and Resource Combinations*.

For further information, see section 3.2, Access Control, section 3.2.6, Policies, section 3.4.1, Import Certificates, and section 6.15.1.3, Policy Combinations.

Table 6-142: Policy Types

Policy Type	Value (Hex)	Value (Int)	Description
POLICY_KEYPAIR	0x01	1	<p>An operation that requires public-private key pairs will fail with <code>ERROR_NOT_ALLOWED [0x500B]</code> if the provided public and private keys are not associated with each other.</p> <p>This policy is evaluated only for private keys with the public key configured as constraining resource (see Table 6-143).</p> <p>The CSP shall throw <code>ERROR_ILLEGAL_CONFIG [0x3009]</code> if the policy is configured on a public key with the private key as the constraining resource.</p> <p>Supported operations:</p> <ul style="list-style-type: none"> • <i>Access Restrictions on Attestation Operations</i> • <i>Access Restrictions on Key Operations</i>
POLICY_SECCHANNEL_ESTABLISHED	0x02	2	<p>A CSP service operation will fail with <code>ERROR_NOT_ALLOWED [0x500B]</code> if the <code>securityState</code> of a specific secure channel is not <code>SECURITY_ESTABLISHED</code>. The secure channel is detected by the constraining resource that is involved in secure channel establishment (see Table 6-143).</p> <p>For example, a key resource with <code>SEC_CA2</code> links to the corresponding EAC secure channel instance.</p> <p>Supported operations:</p> <ul style="list-style-type: none"> • <i>Access Restrictions on Cipher Operations</i> • <i>Access Restrictions on Transform Operations</i> • <i>Access Restrictions on Signature Operations</i> • <i>Access Restrictions on Confidential Data Transfer Operations</i> • <i>Access Restrictions on Password Operations</i> • <i>Access Restrictions on Key Operations</i> • <i>Access Restrictions on Certificate Operations</i> <p>Evaluate Timers (on the secure channel):</p> <ul style="list-style-type: none"> • <code>TIMER_SECURITY_TIMEOUT</code>

Policy Type	Value (Hex)	Value (Int)	Description
POLICY_PASSWORD	0x03	3	<p>A cipher decryption, a signature creation operation or an certificate import will fail with ERROR_NOT_ALLOWED [0x500B] if an associated password is not authenticated.</p> <p>Supported operations:</p> <ul style="list-style-type: none"> Access Restrictions on Cipher Operations Access Restrictions on Signature Operations Access Restrictions on Confidential Data Transfer Operations Access Restrictions on Password Operations Access Restrictions on Key Operations Access Restrictions on Certificate Operations <p>Evaluate Timers (on the constraining password):</p> <ul style="list-style-type: none"> TIMER_AUTH_TIMEOUT <p>Increment Counters (on the constraining password):</p> <ul style="list-style-type: none"> COUNT_AUTH_USAGE
POLICY_UNBLOCK_PASSWORD	0x04	4	<p>Unlocking a password and resetting the tryCounter will fail with ERROR_NOT_ALLOWED [0x500B] if the provided PUK resource is not authenticated and not associated to the password that shall be unblocked.</p> <p>Supported operations:</p> <ul style="list-style-type: none"> Access Restrictions on Password Operations <p>Evaluate Timers (on the constraining password):</p> <ul style="list-style-type: none"> TIMER_AUTH_TIMEOUT <p>Increment Counters (on the constraining password):</p> <ul style="list-style-type: none"> COUNT_AUTH_USAGE
POLICY_PRE_BLOCKED	0x05	5	<p>The last possible password verification before a password gets blocked (i.e., when the tryCounter of the password is already 1) will fail with ERROR_NOT_ALLOWED [0x500B] if the constraining password resource (e.g., a CAN) is not authenticated.</p> <p>Supported operations:</p> <ul style="list-style-type: none"> Access Restrictions on Password Operations <p>Evaluate Timers (on the constraining password):</p> <ul style="list-style-type: none"> TIMER_AUTH_TIMEOUT <p>Increment Counters (on the constraining password):</p> <ul style="list-style-type: none"> COUNT_AUTH_USAGE
POLICY_TA2_ACCESS_FLAG	0x06	6	<p>A CSP service operation will fail with ERROR_NOT_ALLOWED [0x500B] if a TA2 access flag configured in policyData is not present in TA2 certificates received from a terminal according to [TR-03110-3] section 3.</p> <p>The constraining resource must be a certificate with SEC_TA_DV or SEC_TA_TERMINAL used in PROTOCOL_EAC_ID (see Table 6-143).</p> <p>Supported operations:</p> <ul style="list-style-type: none"> Access Restrictions on Cipher Operations Access Restrictions on Transform Operations Access Restrictions on Confidential Data Transfer Operations

Policy Type	Value (Hex)	Value (Int)	Description
POLICY_ASSOCIATION	0x07	7	<p>An operation that requires more than one resource will fail with <code>ERROR_NOT_ALLOWED [0x500B]</code> if the provided resources are not associated with each other.</p> <p>This policy is evaluated only for the main resource used in the operation, i.e., the first resource parameter of the service operation (see Table 6-143).</p> <p>The CSP shall throw an <code>ERROR_ILLEGAL_CONFIG [0x3009]</code> if more than one policy of type <code>POLICY_ASSOCIATION</code> is configured for same owner resource.</p> <p>Note: When this policy is configured, the main resource (e.g., the attestation key) can only be used together with its associated resource, and not with any other resources.</p> <p>Supported operations:</p> <ul style="list-style-type: none"> Access Restrictions on Signature Operations Access Restrictions on Transform Operations Access Restrictions on Attestation Operations

6.15.1.3 Policy and Resource Combinations

The `PolicyModule` shall support the compatible combinations of *Policy Types* listed in Table 6-143 for all *Resource Types* supported by the platform. Unsupported or illegal combinations shall be rejected and handled according to the `errorMode` configured.

For further information, see section 3.2.6, Policy Concept, and section 6.15.1.2, Policy Types.

Table 6-143: Policy and Resource Combinations

Policy Type	Policy Owner Resource	Associated Resource
POLICY_KEYPAIR	Private keys: <ul style="list-style-type: none"> Both private keys in <code>att.init</code> Private key in <code>key.generateKeyPair</code> Private key <code>key.computePublicKey</code> 	Corresponding public key.
POLICY_SECCHANNEL_ESTABLISHED	Resources used in	Key, certificate, or password linking to the specific secure messaging channel.
POLICY_PASSWORD	<ul style="list-style-type: none"> Any key in <code>cipher.init</code> Decryption key in <code>transform.init</code> Any key in <code>sig.init</code> Storage key in <code>sc.initConfidentialWrap</code> Storage key in <code>sc.initConfidentialUnwrap</code> Public key to import in <code>key.initManage</code> Certificate to import in <code>cert.initManage</code> Password in <code>pwd.check</code> Password in <code>pwd.isAuthenticated</code> Password in <code>pwd.update</code> Both passwords in <code>pwd.resetAndUnblock</code> 	Password required to be authenticated.
POLICY_UNBLOCK_PASSWORD	Password to be unblocked in <ul style="list-style-type: none"> <code>pwd.resetAndUnblock</code> 	PUK password required to allow password unblocking.
POLICY_PRE_BLOCKED	Password with <code>tryCounter=1</code> in <ul style="list-style-type: none"> <code>pwd.check</code> 	Password (e.g., CAN) required to allow the final password verification.

Policy Type	Policy Owner Resource	Associated Resource
POLICY_TA2_ACCESS_FLAG	Resources used in <ul style="list-style-type: none"> Any key in cipher.init Decryption key in transform.init Storage key in sc.initConfidentialWrap Storage key in sc.initConfidentialUnwrap 	TA2 certificate configured with SEC_TA_TERMINAL or SEC_TA_DV.
POLICY_ASSOCIATION	The main resources used: <ul style="list-style-type: none"> Signing key in sig.init Decryption key in transform.init (Private) attestation key in att.init 	The other resources used: <ul style="list-style-type: none"> Second key in sig.init Second private key in att.init Encryption key in transform.init

6.15.2 Policy Configuration

The PolicyModule shall support the configuration parameters defined in Table 6-144.

Table 6-144: Policy Configuration Options

Parameter	Description	Parameter Type
policies	Defines enhanced policy rules to further restrict access depending on other resources' states. Technical Details: <ul style="list-style-type: none"> Data type: SET OF CSPPolicy Initial value: empty list Administration: CSPAccessControl 	Resource Parameter
policyType	Defines how the policy is evaluated, selected from the available <i>Policy Types</i> . Technical Details: <ul style="list-style-type: none"> Data type: CSPPolicyType Administration: CSPPolicy 	Policy Parameter
policyData	Used to configure additional data for evaluating the policy. For example for POLICY_TA2_ACCESS_FLAG, the CSP compares the value stored in the policy data field with the access flag present in a Terminal Authentication (TA) certificate. Evaluated by: <ul style="list-style-type: none"> POLICY_TA2_ACCESS_FLAG 	Policy Parameter
policyMode	The policy mode defines how the CSP shall handle scenarios where a configured policy type is not supported by the platform. It is selected from the available <i>Policy Modes</i> . Technical Details: <ul style="list-style-type: none"> Data type: CSPPolicyMode Initial value: 0 (POLICY_MODE_OFF) Administration: CSPPolicySettings 	CSP Instance Parameter

6.15.3 Policy Operations

The CSP does not offer any policy-specific methods or commands.

6.15.4 Policy Lifecycle

Policies are stateless and do not have a lifecycle.

6.15.5 Policy Structures

This section lists ASN.1 structures related to the configuration of policies.

Note: A complete ASN.1 definition is available in [GP CSP ASN1].

6.15.5.1 CSPPolicySupport

This data structure represents features, types, and algorithms of the *Policy Module*.

It is included in the CSPEnforce command to detect their support by the platform.

ASN 6-77: Access: ASN.1 Definition for CSPPolicySupport

```
-- ASN1START
-- Supported policy features.
CSPPolicySupport ::= SEQUENCE {
    -- Policy operation modes to specify the handling of unsupported policies.
    policyModes          SET OF CSPPolicyMode OPTIONAL,

    -- Policy types, e.g., require an authenticated password to allow cipher.
    policyTypes          SET OF CSPPolicyType OPTIONAL
}
-- ASN1STOP
```

6.15.5.2 CSPPolicySettings

This data structure represents a general *Access Configuration* for the CSP Instance.

It can be configured using the CSPSetup structure.

It can be retrieved using the CSPConfiguration structure.

ASN 6-78: Access: ASN.1 Definition for CSPPolicySettings

```
-- ASN1START
-- General policy settings for the CSP Instance.
CSPPolicySettings ::= SEQUENCE {

    -- Select policy mode to specify the handling of unavailable counter types.
    policyMode           CSPPolicyMode

}
-- ASN1STOP
```

6.15.5.3 CSPPolicyMode

This data structure defines the list of available *Policy Modes*.

A policy mode can be selected using the CSPPolicySettings structure.

The policy modes supported can be detected using the CSPCoreSupport structure.

ASN 6-79: Access: ASN.1 Definition for CSPPolicyMode

```
-- ASN1START
-- List of available policy operation modes.
CSPPolicyMode ::= ENUMERATED {

    -- POLICY_MODE_OFF:
    -- Policy evaluation is disabled or not available.
    policy-mode-off                (0),

    -- POLICY_MODE_IGNORE_UNSUPPORTED:
    -- Ignore policy configurations that are not supported by the platform.
    policy-mode-ignore-unsupported (1),

    -- POLICY_MODE_STRICT:
    -- Stop operation if a configured policy is not supported.
    policy-mode-strict             (2)
}
-- ASN1STOP
```

6.15.5.4 CSPPolicy

This data structure represents a container for a resource-specific *Access Configuration*.

It can be configured for a resource using the CSPAccessControl structure.

For further information, see section 3.2, Access Control and section 3.2.6, Policies.

ASN 6-80: Access: ASN.1 Definition for CSPPolicy

```
-- ASN1START
-- Structure to configure advanced access rules for a resource.
CSPPolicy ::= SEQUENCE {

    -- The additional condition that needs to be checked.
    policyType          CSPPolicyType,

    -- The associated resource that is evaluated for the policy.
    constrainingResourceId CSPResourceId,

    -- Policy specific data, e.g., required TA access rights.
    additionalData       OCTET STRING (SIZE(8)) OPTIONAL
}
-- ASN1STOP
```

6.15.5.5 CSPPolicyType

This data structure defines the list of available *Policy Types*.

A policy type can be selected using the CSPPolicySettings structure.

The policy types supported can be detected using the CSPCoreSupport structure.

ASN 6-81: Access: ASN.1 Definition for CSPPolicyType

```
-- ASN1START
-- List of available policy types.
CSPPolicyType ::= ENUMERATED {

    -- POLICY_KEYPAIR:
    -- The public key provided must be associated to the private key.
    policy-keypair                (1),

    -- POLICY_SECCHANNEL_ESTABLISHED:
    -- An associated secure channel must be fully established.
    policy-secchannel-established  (2),

    -- POLICY_PASSWORD:
    -- An associated password must be authenticated.
    policy-password               (3),

    -- POLICY_UNBLOCK_PASSWORD:
    -- The PUK used to unblock a password must be associated to the password.
    policy-unblock-password       (4),

    -- POLICY_PRE_BLOCKED:
    -- A password with tryCounter=1 requires an associated CAN authenticated.
    policy-pre-blocked            (5),

    -- POLICY_TA2_ACCESS_FLAG:
    -- A specific access flag must be present in sec channel TA2 certificates.
    policy-ta2-access-flag        (6),

    -- POLICY_ASSOCIATION:
    -- The second resource involved must be associated to the main resource.
    policy-association            (7)
}
-- ASN1STOP
```

6.16 Random Data Module

The CSP may implement the `RandomDataModule`; the choice shall be subject to *Modularity*.

The Client Application may generate random data using the `rand.nextBytes` operation, after initialization with `rand.setSeed`, as specified in Table 6-145.

The CSP shall use a Secure Random Number Generator (SRNG) conforming to one of the following functionality classes defined in [AIS 20/31]:

- Physical True Random Number Generator of class PTG.3
- Deterministic Random Number Generator of class DRG.4, seeded by a Physical True Random Number Generator of class PTG.2 or PTG.3

The CSP Admin may detect whether the platform supports this module using the `CSPEnforce` command.

Table 6-145: Random Data Operations

Operation	Description
<code>rand.nextBytes</code>	Generates random data. CSP API: <ul style="list-style-type: none"> • <code>org.globalplatform.csp.api.RandomDataService.nextBytes(..)</code>
<code>rand.setSeed</code>	Seeds the random data generator. CSP API: <ul style="list-style-type: none"> • <code>org.globalplatform.csp.api.RandomDataService.setSeed(..)</code>

7 CSP PROTOCOL

The CSP Protocol defines two types of commands for interacting with the CSP:

- CSPAdminCommand – for managing the CSP Instance.
- CSPClientCommand – for accessing cryptographic services by off-card Clients.

Both command types are transmitted using the same APDU format: the STORE DATA instruction (CLA = 0x80, INS = 0xE2), with standard GlobalPlatform chaining behavior. The data is encoded using ASN.1 DER format and structured as a SEQUENCE containing a CHOICE, which identifies the specific operation through context-specific tagging. See [GP CSP ASN1] for the complete tagging definitions.

The CSP shall encode and serialize all CSP command structures defined in this chapter in accordance with [TCA eUICC] and the following rules:

- All objects shall be encoded using the ASN.1 Distinguished Encoding Rules (DER).
- Elements denoted as OPTIONAL may or may not be present in the encoded representation.
- Mandatory elements that have no value shall still be present in the encoded representation with a length of 0x00, indicating an empty value.
- Implicitly tagged elements shall not include additional explicit tags; for example, a sequence tagged with 0xA1 shall not be preceded by an explicit 0x30 tag.

7.1 CSPAdminCommand

The CSPAdminCommand provides administrative operations for managing the CSP. This command includes several *Shared Command* Structures, one for each operation (e.g., CSPRegisterClient, CSPCreateResource). For a complete list, see ASN 7-1.

The CSP shall process the CSPAdminCommand via a STORE DATA APDU, for example by:

- implementing the processData method of the org.globalplatform.Personalization interface ([GP API]) to handle the CSPAdminCommand. In this case, the CSP shall ensure the command is received over a secure channel with security level authenticated.
- implementing the process method of the javacard.framework.Applet class ([JC-API]) to handle the CSPAdminCommand. In this case, the CSP is responsible for establishing a secure channel using Security Domain services and for unwrapping the commands received with security level authenticated.

The CSP Admin may invoke the CSPAdminCommand via the Store Data APDU specified in Table 7-1, secured by a secure channel to the CSP's SD using the GlobalPlatform Personalization interface according to [GP Card Spec] section 7.3.2.

This APDU may be chained by sending multiple Store Data APDUs with the appropriate P1 and P2 values, as defined in [GP Card Spec].

For further information, see section 3.1, Resource Management.

Table 7-1: CSP Protocol: CSP-Admin Commands

Name	CLA	INS	P1	P2	ASN.1 Data
STORE DATA	0x80	0xE2	0x11 or 0x91 (for last block)	0x00 - 0xFF	Contains the CSPAdminCommand specified in ASN 7-1 including the encapsulating CHOICE tag.

ASN 7-1: CSP Protocol: ASN.1 Definition for the CSPAdminCommand

```
-- ASN1START
GPCSPDefinitions {
    iso(1) member-body(2) country-USA(840) globalPlatform (114283)
    card-management(100) modules(0) gp-csp-admin(3)
}

DEFINITIONS
AUTOMATIC TAGS ::=
BEGIN

-- Administrative operations to manage a CSP Application through its SD.
CSPAdminCommand ::= [1] SEQUENCE {
    adminCommandChoice CHOICE {

        -- Detect features and/or algorithms supported by this platform.
        enforce                CSPEnforce,

        -- Register the AID of a Client Application for use with this CSP Instance.
        registerApplication     CSPRegisterClient,

        -- Unregister a Client Application from this CSP Instance.
        unregisterApplication    CSPUnregisterClient,

        -- Create a key, certificate, password, counter or timer as resource.
        createResource          CSPCreateResource,

        -- Destroy a resource and free memory.
        destroyResource          CSPDestroyResource,

        -- Change the configuration of a resource.
        configureResource         CSPConfigureResource,

        -- Change the general settings of the CSP Instance.
        setup                    CSPSetup,

        -- Activate this CSP Instance for operational use.
        activate                  CSPActivate,

        -- Deactivate this CSP Instance, thus Client Applications cannot use it.
        deactivate                CSPDeactivate,

        -- Retrieve the entire configuration settings of this CSP Instance.
        getConfig                  CSPGetConfiguration,

        -- Set the value of a resource (e.g., key value, password, certificate).
```

```

setValue                CSPSetValue,

-- Securely wipe the value of a resource.
clearResource           CSPClearResource,

-- Compute an SE Platform attestation or CSP Config attestation.
systemAttestation       CSPSystemAttestation,

-- Generate a symmetric or private key value with random data.
generateKey             CSPGenerateKey,

-- Compute the public part of a private key value.
computePublicKey        CSPComputePublicKey,

-- Derive a key from a source key and optional input data.
deriveKey               CSPDeriveKey,

-- Set the reference time that is used to estimate the system time.
settime                 CSPSetTime
}
}
END
-- ASN1STOP

```

For all operations except CSPDeactivate, the CSP may return the following Status Word (SW) in addition to the SW specific to each command operation:

- ERROR_ILLEGAL_CONFIG:
 - The CSP is in CSP_MODE_OPERATIONAL and cannot be configured until deactivated [0x3005].

7.1.1 CSPEnforce

This command checks if specific features or algorithms are supported on a platform as part of the *Modularity* concept. The CSP Admin provides a list of features and algorithms. If any are unsupported, the command returns an ERROR_NOT_SUPPORTED.

For further information, see section 5.1.3, System Operations.

ASN 7-2: Admin: ASN.1 Definition for CSPEnforce

```

-- ASN1START
-- Command that checks features and algorithms on the platform.
CSPEnforce ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion    CSPProtocolVersion,

    -- Optional core functionality (e.g., policies).
    coreSupport           CSPCoreSupport OPTIONAL,

```

```
-- Cipher and padding algorithms.
cipherSupport          CSPPCipherSupport OPTIONAL,

-- Signature algorithms.
signatureSupport       CSPSignatureSupport OPTIONAL,

-- Cipher encryption transformation.
transformSupport       NULL OPTIONAL,

-- Secure channel protocols.
secChannelSupport      CSPSecureChannelSupport OPTIONAL,

-- Confidential data transfer extension of the secure channel service.
confidentialSupport    NULL OPTIONAL,

-- Attestation types.
attestationSupport     CSPAttestationSupport OPTIONAL,

-- Key type, size, curve, along with derivation and agreement algorithms.
keySupport             CSPKeySupport OPTIONAL,

-- Certificate types.
certificateSupport      CSPCertificateSupport OPTIONAL,

-- Password management operations.
passwordSupport         CSPPasswordSupport OPTIONAL,

-- Counter types.
counterSupport          CSPCounterSupport OPTIONAL,

-- Time management and timer types.
timeSupport            CSPTIMESupport OPTIONAL,

-- Secure auditing and event types.
auditSupport           CSAuditSupport OPTIONAL,

-- Resource import and export functionality.
offloadingSupport      NULL OPTIONAL,

-- Fields to be included in attestation results and log messages.
fieldSupport           CSPFieldSupport OPTIONAL,

-- Constraint-based access control using policies.
policySupport          CSPPolicySupport OPTIONAL
}
```

```
-- ASN1STOP
```

This command returns a Status Word (SW) without any additional data:

- 9000: Success
 - All requested features and algorithms are supported.
- ERROR_NOT_SUPPORTED:
 - One or more requested features or algorithms are not supported, e.g., [0x8012], [0x80B1].

7.1.2 CSPRegisterClient

This command registers a Client Application or an off-card Client to the CSP.

The *Client Application Registration* process for on-card Clients includes at least registering the Application Identifier (AID) of the Client Application. Additional *Access Control* checks may be configured within the CSPClientApplication structure, such as SD, DAP, or LFDBH verification. These checks are verified by the CSP when an Application attempts to *Retrieve the CSP Instance*.

Furthermore, the command allows to configure authResources and authProtocol used for *Client Authentication* of both Client Applications and off-card Clients. If configured, the CSP shall *Enforce Client Authentication* before allowing the invocation of application-specific CSP services

This command allocates memory to store these configuration settings per CSP Client, which can be freed by unregistering the Client using the CSPUnregisterClient command.

Note: If the SD verification is activated, the Client Application's SD should exist before the CSP Admin invokes this command.

For further information, see section 5.1.3, System Operations.

ASN 7-3: Admin: ASN.1 Definition for CSPRegisterClient

```
-- ASN1START
-- Command to register a Client Application or an off-card Client to the CSP.
CSPRegisterClient ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- The Client Application or off-card Client to register.
    client                  CSPClient
}
-- ASN1STOP
```

This command returns a Status Word (SW) without any additional data:

- 9000: Success
 - Ignore if clientId or AID is already registered (re-invoking for same AID overwrites settings).

Note: The CSP may defer validation of cryptographic configurations until runtime, such as invalid authProtocol or authResources, or missing authentication setup for off-card clients.

7.1.3 CSPUnregisterClient

This command deregisters a CSP Client, revoking all its access to the CSP.

It frees the memory allocated during the Client registration process via *CSPRegisterClient*.

For further information, see section 5.1.3, System Operations.

ASN 7-4: Admin: ASN.1 Definition for CSPUnregisterClient

```
-- ASN1START
-- Command to unregister a CSP Client, revoking its access to the CSP.
CSPUnregisterClient ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion    CSPProtocolVersion,

    -- The client identifier or the AID of a Client Application.
    client                CSPClientReference
}
-- ASN1STOP
```

This command returns a Status Word (SW) without any additional data:

- 9000: Success
 - Ignore if the provided `clientId` or AID is not registered.

7.1.4 CSPCreateResource

This command creates a new CSP Resource, such as a key or password, with specific parameters like key type and size. Resources created with this *Create Resources* command are in `STATE_UNINITIALIZED` and have no cryptographic value assigned. The actual resource value is assigned subsequently, using commands such as *CSPSetValue*, *CSPGenerateKey*, *CSPComputePublicKey*, or *CSPDeriveKey*.

It allocates memory to store the resource parameters and pre-reserves space for the cryptographic value; both freed when the resource is removed using the *CSPDestroyResource* command.

Note: Resource parameters related to access control, algorithms, counters, timers and events can be modified using the *CSPConfigureResource* command. Parameters that define the resource itself, such as resource type and sizes, are not modifiable.

For further information, see section 5.1.3, System Operations.

ASN 7-5: Admin: ASN.1 Definition for CSPCreateResource

```
-- ASN1START
-- Command to create a key, certificate or password resource.
CSPCreateResource ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion    CSPProtocolVersion,

    -- The resource data (e.g., resource ID, resource type, etc.).
    resouceData           CSPResource
}
```

```
}
-- ASN1STOP
```

This command returns a Status Word (SW) without any additional data:

- 9000: Success
- ERROR_ILLEGAL_VALUE:
 - Resource identifier already exists [0x2002].
- ERROR_ILLEGAL_CONFIG:
 - CSP not in CSP_MODE_CONFIGURATION [0x3005].

Note: The CSP may defer validation of cryptographic configurations until runtime, such as inconsistent algorithm configuration, incompatible key parameters, usage type, or unsupported features.

7.1.5 CSPDestroyResource

This command deregisters and destroys a resource. This *Destroy Resources* command can only be applied to resources in STATE_UNINITIALIZED, with their cryptographic value already securely cleared.

It frees the memory allocated during resource creation with CSPCreateResource.

For further information, see section 5.1.3, System Operations.

ASN 7-6: Admin: ASN.1 Definition for CSPDestroyResource

```
-- ASN1START
-- Command to remove a key or password resource.
CSPDestroyResource ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- The resource ID that shall be unregistered.
    resourceId              CSPResourceId
}
-- ASN1STOP
```

This command returns a Status Word (SW) without any additional data:

- 9000: Success
 - Ignore if the resourceId does not exist.
- ERROR_ILLEGAL_CONFIG:
 - Resource not in STATE_UNINITIALIZED [0x3004].
 - CSP not in CSP_MODE_CONFIGURATION [0x3005].

7.1.6 CSPConfigureResource

This command modifies resource parameters to *Configure Resources*, changing access control, algorithm, counters, timers and event logging configuration of the resource. Parameters that define the resource itself, such as resource type and sizes, cannot be changed.

Repeated executions for the same resource identifier overwrite previous configurations, with only the most recent change taking effect. Omitted parameters leave the corresponding configuration parts unchanged.

For further information, see section 5.1.3, System Operations.

ASN 7-7: Admin: ASN.1 Definition for CSPConfigureResource

```
-- ASN1START
-- Command to configure a resource.
CSPConfigureResource ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- The resource to be modified.
    resourceId              CSPResourceId,

    -- The access control configuration for the resource.
    accessControl           CSPAccessControl OPTIONAL,

    -- The algorithm configuration of the resource.
    algorithms              CSPAlgorithms OPTIONAL,

    -- Usage counter or authentication counter configurations of the resource.
    counters                CSPCounters OPTIONAL,

    -- Validity date, validity period or timeout configurations.
    timers                  CSPTimers OPTIONAL,

    -- The events that shall be logged for this resource.
    resourceEvents          SET OF CSPResourceEvent OPTIONAL
}
-- ASN1STOP
```

This command returns a Status Word (SW) without any additional data:

- 9000: Success
- ERROR_ILLEGAL_VALUE:
 - Resource identifier does not exist [0x2001].
- ERROR_ILLEGAL_CONFIG:
 - CSP not in CSP_MODE_CONFIGURATION [0x3005].

Note: The CSP may defer validation of cryptographic configurations until runtime, such as inconsistent algorithm configuration, incompatible key parameters, usage type, or unsupported features.

7.1.7 CSPSetup

This command sets general system settings for the CSP Instance.

Repeated executions overwrite previous configurations, with only the most recent change taking effect. Omitted parameters leave the corresponding configuration parts unchanged.

For further information, see section 5.1.3, System Operations.

ASN 7-8: Admin: ASN.1 Definition for CSPSetup

```
-- ASN1START
-- Command to setup general settings of the CSP Instance.
CSPSetup ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- Version, name, attestation key and error handling of this CSP Instance.
    cspSettings             CSPSettings OPTIONAL,

    -- Set the general secure channel authentication timeout.
    secureChannelSettings   CSPSecureChannelSettings OPTIONAL

    -- Select policy mode for handling unavailable policy types.
    policySettings          CSPPolicySettings OPTIONAL,

    -- Select counter mode for handling unavailable counter types and sizes.
    counterSettings         CSPCounterSettings OPTIONAL,

    -- Configure time management and handling of unavailable time.
    timeSettings            CSPTIMESettings OPTIONAL,

    -- Configure audit event logging and handling of unavailable event types.
    auditSettings           CSPAuditSettings OPTIONAL,

    -- Select field mode for handling unavailable signature fields.
    fieldSettings           CSPFieldSettings OPTIONAL
}
-- ASN1STOP
```

This command returns a Status Word (SW) without any additional data:

- 9000: Success
- ERROR_ILLEGAL_CONFIG:
 - CSP not in CSP_MODE_CONFIGURATION [0x3005].

Note: The CSP may defer validation of cryptographic configurations until runtime, such as inconsistent algorithm configuration, or unsupported features.

7.1.8 CSPActivate

This command activates the CSP, transitioning it from CSP_MODE_CONFIGURATION to CSP_MODE_OPERATIONAL and making it available for use by Client Applications, as illustrated in Figure 5-1 *System Lifecycle*.

It can be deactivated again using the CSPDeactivate command.

Invocations of this command can be logged, as specified for EVENT_CSP_CONFIG_UPDATED.

For further information, see section 5.1.3, System Operations.

ASN 7-9: Admin: ASN.1 Definition for CSPActivate

```
-- ASN1START
-- Command to finalize a CSP Configuration.
CSPActivate ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- Custom version of the CSP Configuration chosen by the CSP Admin.
    configVersion           CSPConfigVersion OPTIONAL
}
-- ASN1STOP
```

This command returns a Status Word (SW) without any additional data:

- 9000: Success
 - Ignore if the CSP is already in CSP_MODE_OPERATIONAL.

7.1.9 CSPDeactivate

This command deactivates the CSP, transitioning it from CSP_MODE_OPERATIONAL to CSP_MODE_CONFIGURATION and enabling CSP Admins to modify the CSP Configuration, as illustrated in Figure 5-1 *System Lifecycle*.

It can be activated again using the CSPActivate command.

For further information, see section 5.1.3, System Operations.

ASN 7-10: Admin: ASN.1 Definition for CSPDeactivate

```
-- ASN1START
-- Command to deactivate a CSP Configuration.
CSPDeactivate ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion
}
-- ASN1STOP
```

This command returns a Status Word (SW) without any additional data:

- 9000: Success
 - Ignore if the CSP is already in CSP_MODE_CONFIGURATION.

7.1.10 CSPGetConfiguration

This command returns the complete configuration of the CSP Instance.

ASN 7-11: Admin: ASN.1 Definition for CSPGetConfiguration

```
-- ASN1START
-- Command to retrieve the entire CSP Configuration.
CSPGetConfiguration ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion
}
-- ASN1STOP
```

This command returns a Status Word (SW) with additional data specified below:

- 9000: Success

ASN 7-12: Admin: ASN.1 Definition for CSPGetConfigurationResponse

```
-- ASN1START
-- Response of the CSPGetConfiguration command.
CSPGetConfigurationResponse ::= SEQUENCE {

    -- Version of the CSP Admin Protocol used.
    cspProtocolVersion      CSPProtocolVersion,

    -- CSP Configuration of the entire CSP Instance.
    cspConfiguration        CSPConfiguration
}
-- ASN1STOP
```

7.1.11 CSPSetValue

This command imports a resource value, such as a cryptographic key, certificate, password, counter or timer, transitioning the resource to STATE_OPERATIONAL. It can be used, for example, to *Load Resource for Personalization*, *Import Certificates*, or for *Updating Passwords*.

The command also resets associated timers and counters to their initial values, such as a usageCounter.

It supports all resource types except KEY_MASTER_SECRET, KEY_DERIVED_SECRET, and KEY_SHARED_SECRET.

The resource must be in STATE_UNINITIALIZED with ADMIN_SETUP right granted.

The command supports adding an additional encryption layer for importing sensitive data, such as private keys or passwords, by providing a decryption resource with USAGE_SYSTEM.

It does not allocate additional memory, as memory allocation is already completed during resource creation.

Invocations of this command can be logged, as specified for EVENT_RESOURCE_VALUE_SET.

Note: As with other commands, importing values over 255 bytes uses extended length mechanisms in the APDU transport layer ([ISO 7816-4]).

For further information, see section 5.2.3, Resource Operations.

ASN 7-13: Resource: ASN.1 Definition for CSPSetValue

```
-- ASN1START
```

```
-- Command to set the value of a key, certificate or pwd resource.
CSPSetValue ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion    CSPProtocolVersion,

    -- The identifier of the resource to set the value for.
    resourceId            CSPResourceId,

    -- If set, the data is decrypted before being assigned to the resource.
    decryptionResourceId   CSPResourceId OPTIONAL,

    -- The value of the resource.
    data                  OCTET STRING (SIZE(0..65536)),

    -- Algorithm-specific initialization data for decryption, e.g., iv data.
    initializationData     OCTET STRING (SIZE(0..16)) OPTIONAL,

    -- Is this an initial value requiring change (only for passwords)?
    inTransport            BOOLEAN
}
-- ASN1STOP
```

This command returns a Status Word (SW) without any additional data:

- 9000: Success
 - Ignore policy rules.
 - Ignore password minimum and maximum size configurations.
- ERROR_ILLEGAL_VALUE:
 - Resource identifier does not exist [0x2001].
- ERROR_ILLEGAL_USE:
 - Incompatible data format [0x6001].
- ERROR_ILLEGAL_CONFIG:
 - Resource is not in STATE_UNINITIALIZED [0x3004].
- ERROR_NOT_ALLOWED:
 - Resource missing ADMIN_SETUP [0x5008].

7.2 CSPClientCommand

Support for the CSPClientCommand is optional. This command provides operations intended for use by off-card Clients to utilize the pre-defined cryptographic services of the CSP, as specified in ASN 7-14.

An off-card Client may invoke the CSPClientCommand via the Store Data APDU specified in Table 7-2. All structures within the requiresAuthentication element require access protection using a secure channel established according to section 5.3.1.1, *Client Authentication*.

The CSP shall reject any invocation of a command listed under the requiresAuthentication unless *Client Authentication* has been successfully established via the processAuthentication element.

This APDU may be chained by sending multiple Store Data APDUs with the appropriate P1 and P2 values, as defined in [GP Card Spec].

Table 7-2: CSP Protocol: CSP-Client Commands

Name	CLA	INS	P1	P2	ASN.1 Data
STORE DATA	0x80	0xE2	0x11 or 0x91 (for last block)	0x00 - 0xFF	Contains the CSPClientCommand specified in ASN 7-2 including the encapsulating CHOICE tag.

ASN 7-14: CSP Protocol: ASN.1 Definition for the CSPClientCommand

```
-- ASN1START
GPCSPDefinitions {
    iso(1) member-body(2) country-USA(840) globalPlatform (114283)
    card-management(100) modules(0) gp-csp-client(4)
}

DEFINITIONS
AUTOMATIC TAGS ::=
BEGIN

-- Client operations to use the cryptographic services of the CSP.
CSPClientCommand ::= [2] SEQUENCE {
    clientCommandChoice CHOICE {

        -- Compute an SE platform attestation or a CSP Config attestation.
        systemAttestation          CSPSystemAttestation,

        -- Process secure channel authentication defined via authProtocol.
        processAuthentication      CSPProcessSecurity,

        -- Command structures that require Client Authentication.
        requiresAuthentication     CHOICE {

            -- Retrieve the entire configuration settings of this CSP Instance.
            getConfig                CSPGetConfiguration,

            -- Securely wipe the value of a resource.
```

```

clearResource          CSPClearResource,

-- Generate a symmetric or private key value with random data.
generateKey            CSPGenerateKey,

-- Compute the public part of a private key value.
computePublicKey       CSPComputePublicKey,

-- Derive a key from a source key and optional input data.
deriveKey              CSPDeriveKey,

-- Set the reference time that is used to estimate the system time.
settime                CSPSetTime,

-- Create a signature.
sign                   CSPSign,

-- Verify a signature.
verifySignature        CSPVerifySignature,

-- Encrypt data.
encrypt                CSPEncrypt,

-- Decrypt data.
decrypt                CSPDecrypt,

-- Compute a resource attestation.
resourceAttestation    CSPResourceAttestation
}
}
}
END
-- ASN1STOP

```

For all operations, the CSP may return the following Status Word (SW) in addition to the SW specific to each command operation:

- ERROR_ILLEGAL_CONFIG:
 - The CSP is in CSP_MODE_CONFIGURATION and cannot be utilized [0x3001].

7.2.1 CSPProcessSecurity

This command processes security for secure channel authentication according to authProtocol and authResources configured.

Note: The operation behaves similarly to sc.processSecurity but is dedicated to *Client Authentication*.

For further information, see section 3.2.2, Enforce Client Authentication.

ASN 7-15: Secure Channel: ASN.1 Definition for CSPPProcessSecurity

```
-- ASN1START
-- Command to process security for secure channel authentication.
CSPPProcessSecurity ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- APDU data for processing security.
    apduData                OCTET STRING (SIZE(1..32767))
}
-- ASN1STOP
```

This command returns a Status Word (SW) and the output APDU specified below:

- 9000: Success
- ERROR_ILLEGAL_CONFIG:
 - CSP not in CSP_MODE_OPERATIONAL [0x3001].
 - Missing resource(s) [0x3002], not initialized [0x3003], or inconsistent secure channel configuration [0x3040].
- ERROR_NOT_ALLOWED:
 - Resource(s) missing ACCESS_USE [0x5007], not configured for USAGE_SECCHANNEL [0x5040], is in STATE_EXHAUSTED [0x50A0] or in STATE_EXPIRED [0x50B1].
- ERROR_ILLEGAL_USE:
 - Illegal input APDU [0x6040].
- ERROR_NOT_SUPPORTED:
 - Authentication protocol [0x8041] not supported.

ASN 7-16: Attestations: ASN.1 Definition for CSPPProcessSecurityResponse

```
-- ASN1START
-- Response of the CSPPProcessSecurity command.
CSPPProcessSecurityResponse ::= SEQUENCE {

    -- Output APDU resulting from secure channel processing.
    outputData      OCTET STRING (SIZE(1..32767))
}
-- ASN1STOP
```

7.2.2 CSPSign

This command creates a signature.

Note: The operation behaves similarly to `sig.sign`.

For further information, see section 6.2.3, Signature Operations.

ASN 7-17: Signature: ASN.1 Definition for CSPSign

```
-- ASN1START
-- Command for data signing.
CSPSign ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- The resource ID used to compute the signature.
    signingResourceId       CSPResourceId,

    -- The data to sign.
    inputData               OCTET STRING (SIZE (0..32767))
}
-- ASN1STOP
```

This command returns a Status Word (SW) and the signature computed:

- 9000: Success
- ERROR_ILLEGAL_VALUE:
 - Resource ID does not exist [0x2001] or is not a RESOURCE_KEY [0x2021].
- ERROR_ILLEGAL_CONFIG:
 - CSP not in CSP_MODE_OPERATIONAL [0x3001].
 - Resource not initialized [0x3003], inconsistent signature configuration [0x3020], inconsistent policy config [0x3009], or invalid field configuration [0x30E0].
- ERROR_NOT_ALLOWED:
 - Client not authenticated [0x5006].
 - Resource missing ACCESS_USE [0x5007], not configured for USAGE_SIGNATURE [0x5020], is in STATE_EXHAUSTED [0x50A0] or in STATE_EXPIRED [0x50B1], or a policy failed [0x500B].
- ERROR_ILLEGAL_USE:
 - Invalid input [0x6010]/[0x6011]/[0x6020]/[0x6021], or counter capacity reached [0x60A2].
- ERROR_NOT_SUPPORTED:
 - Padding [0x8011], message digest [0x8021], or signature algorithm [0x8022] not supported.

ASN 7-18: Attestations: ASN.1 Definition for CSPSignResponse

```
-- ASN1START
-- Response of the CSPSign command: The signature computed by the CSP.
CSPSignResponse ::= SEQUENCE {
    signature      CSPSignature
}
-- ASN1STOP
```

7.2.3 CSPVerifySignature

This command verifies a signature.

Note: The operation behaves similarly to `sig.verify`.

For further information, see section 6.2.3, Signature Operations.

ASN 7-19: Signature: ASN.1 Definition for CSPVerifySignature

```
-- ASN1START
-- Command to verify a signature.
CSPVerifySignature ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- The resource ID used to compute the signature.
    signingResourceId       CSPResourceId,

    -- The signed data.
    data                    OCTET STRING (SIZE (0..32767)),

    -- The signature to verify.
    signature               CSPSignature
}
-- ASN1STOP
```

This command returns a Status Word (SW) and the verification result:

- 9000: Success
- ERROR_ILLEGAL_VALUE:
 - Resource ID does not exist [0x2001] or is not a RESOURCE_KEY [0x2021].
- ERROR_ILLEGAL_CONFIG:
 - CSP not in CSP_MODE_OPERATIONAL [0x3001].
 - Resource not initialized [0x3003] or inconsistent signature configuration [0x3020].
- ERROR_NOT_ALLOWED:
 - Client not authenticated [0x5006].
 - Resource missing ACCESS_USE [0x5007], not configured for USAGE_SIGNATURE [0x5020], is in STATE_EXHAUSTED [0x50A0] or in STATE_EXPIRED [0x50B1], or a policy failed [0x500B].
- ERROR_ILLEGAL_USE:
 - Invalid input [0x6010]/[0x6011]/[0x6020]/[0x6021], or counter capacity reached [0x60A2].
- ERROR_NOT_SUPPORTED:
 - Padding [0x8011], message digest [0x8021], or signature algorithm [0x8022] not supported.

ASN 7-20: Attestations: ASN.1 Definition for CSPVerifySignatureResponse

```
-- ASN1START
```

```
-- Response of the CSPVerifySignature command.
CSPVerifySignatureResponse ::= SEQUENCE {
    response      CSPBoolean
}
-- ASN1STOP
```

7.2.4 CSPEncrypt

This command encrypts data.

Note: The operation behaves similarly to `cipher.doFinal` in CIPHER_MODE_ENCRYPT.

For further information, see section 6.1.3, Cipher Operations.

ASN 7-21: Cipher: ASN.1 Definition for CSPEncrypt

```
-- ASN1START
-- Command to encrypt data.
CSPEncrypt ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- The resource ID used to encrypt the data.
    keyResourceId           CSPResourceId,

    -- Algorithm-specific initialization data for encryption, e.g., IV data.
    initializationData      OCTET STRING (SIZE(0..16)) OPTIONAL,

    -- The data to encrypt.
    inputData               OCTET STRING (SIZE (0..32767))
}
-- ASN1STOP
```

This command returns a Status Word (SW) and the encrypted data:

- 9000: Success
- ERROR_ILLEGAL_VALUE:
 - Resource ID does not exist [0x2001] or is not a RESOURCE_KEY [0x2021].
- ERROR_ILLEGAL_CONFIG:
 - CSP not in CSP_MODE_OPERATIONAL [0x3001].
 - Resource not initialized [0x3003], inconsistent cipher configuration [0x3010], or inconsistent policy config [0x3009].
- ERROR_NOT_ALLOWED:
 - Client not authenticated [0x5006].
 - Resource missing ACCESS_USE [0x5007], not configured for USAGE_CIPHER [0x5010], is in STATE_EXHAUSTED [0x50A0] or in STATE_EXPIRED [0x50B1], or a policy failed [0x500B].

- **ERROR_ILLEGAL_USE:**
 - Invalid input [0x6010]/[0x6011]/[0x6012]/[0x6013]/[0x6014], or counter capacity reached [0x60A2].
- **ERROR_NOT_SUPPORTED:**
 - Padding [0x8011] or cipher algorithm [0x8022] not supported.

ASN 7-22: Attestations: ASN.1 Definition for CSPEncryptResponse

```
-- ASN1START
-- Response of the CSPEncrypt command: the encrypted data.
CSPEncryptResponse ::= OCTET STRING (SIZE (0..32767))
-- ASN1STOP
```

7.2.5 CSPDecrypt

This command decrypts data.

Note: The operation behaves similarly to `cipher.doFinal` in `CIPHER_MODE_DECRYPT`.

For further information, see section 6.1.3, Cipher Operations.

ASN 7-23: Cipher: ASN.1 Definition for CSPDecrypt

```
-- ASN1START
-- Command to decrypt data.
CSPDecrypt ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- The resource ID used to decrypt the data.
    keyResourceId           CSPResourceId,

    -- Algorithm-specific initialization data for decryption, e.g., IV data.
    initializationData      OCTET STRING (SIZE(0..16)) OPTIONAL,

    -- The data to decrypt.
    inputData               OCTET STRING (SIZE (0..32767))
}
-- ASN1STOP
```

This command returns a Status Word (SW) and the decrypted data:

- 9000: Success
- **ERROR_ILLEGAL_VALUE:**
 - Resource ID does not exist [0x2001] or is not a `RESOURCE_KEY` [0x2021].
- **ERROR_ILLEGAL_CONFIG:**
 - CSP not in `CSP_MODE_OPERATIONAL` [0x3001].
 - Resource not initialized [0x3003], inconsistent cipher configuration [0x3010], or inconsistent policy config [0x3009].

- ERROR_NOT_ALLOWED:
 - Client not authenticated [0x5006].
 - Resource missing ACCESS_USE [0x5007], not configured for USAGE_CIPHER [0x5010], is in STATE_EXHAUSTED [0x50A0] or in STATE_EXPIRED [0x50B1], or a policy failed [0x500B].
- ERROR_ILLEGAL_USE:
 - Invalid input [0x6010]/[0x6011]/[0x6012]/[0x6013]/[0x6014], or counter capacity reached [0x60A2].
- ERROR_NOT_SUPPORTED:
 - Padding [0x8011] or cipher algorithm [0x8022] not supported.

ASN 7-24: Attestations: ASN.1 Definition for CSPDecryptResponse

```
-- ASN1START
-- Response of the CSPDecrypt command: the decrypted data.
CSPDecryptResponse ::= OCTET STRING (SIZE (0..32767))
-- ASN1STOP
```

7.2.6 CSPResourceAttestation

This command computes resource attestations.

Note: The operation behaves similarly to att.computeAttestation.

For further information, see section 6.6.3, Attestation Operations.

ASN 7-25: Attestations: ASN.1 Definition for CSPResourceAttestation

```
-- ASN1START
-- Command to compute resource attestations.
CSPResourceAttestation ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- The attestation type used.
    attestationType         CSPResourceAttestationType,

    -- The resource ID used to create the attestation signature.
    attestationResourceId   CSPResourceId,

    -- Additional input data to be included in the attestation.
    inputData               OCTET STRING (SIZE (16..512))
}
-- ASN1STOP
```

This command returns a Status Word (SW) with the signed attestation data specified below:

- 9000: Success
- ERROR_ILLEGAL_CONFIG:
 - CSP not in CSP_MODE_OPERATIONAL [0x3001].

- Resource not initialized [0x3003], inconsistent signature configuration [0x3020], or invalid field configuration [0x30E0].
- ERROR_NOT_ALLOWED:
 - Attestation key missing ACCESS_USE [0x5007], is not configured for USAGE_ATTESTATION [0x5060], or is in STATE_EXHAUSTED [0x50A0] or STATE_EXPIRED [0x50B1].
- ERROR_NOT_SUPPORTED:
 - Padding [0x8011], message digest [0x8021], or signature algorithm [0x8022] not supported.

ASN 7-26: Attestations: ASN.1 Definition for CSPResourceAttestationResponse

```
-- ASN1START
-- Response of the CSPResourceAttestation command.
CSPResourceAttestationResponse ::= CHOICE {

    -- Conditional: Response of a data attestation.
    dataAttestation          CSPDataAttestation,

    -- Conditional: Response of the key attestation.
    keyAttestation           CSPKeyPoPAttestation
}
-- ASN1STOP
```

7.3 Shared Command Structures

This section defines ASN.1 command structures that may be present in the CSPAdminCommand and the CSPClientCommand.

7.3.1 CSPClearResource

This command securely erases the value of a resource, transitioning it to STATE_UNINITIALIZED to *Clear Resources*. The resource requires ADMIN_CLEAR and can be cleared in any state, including STATE_UNINITIALIZED.

Invocations of this command can be logged, as specified for EVENT_RESOURCE_CLEARED.

Note: The operation behaves exactly the same as resource.clear.

For further information, see section 5.2.3, Resource Operations.

ASN 7-27: Resource: ASN.1 Definition for CSPClearResource

```
-- ASN1START
-- Command to remove the value of a key or password resource.
CSPClearResource ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion    CSPProtocolVersion,

    -- The identifier of the resource that shall be cleared.
    resourceId            CSPResourceId
}
-- ASN1STOP
```

This command returns a Status Word (SW) without any additional data:

- 9000: Success
 - No action if resource is already in STATE_UNINITIALIZED.
- ERROR_ILLEGAL_VALUE:
 - Resource identifier does not exist [0x2001].
- ERROR_NOT_ALLOWED:
 - Resource missing ADMIN_CLEAR [0x5009].

7.3.2 CSPSystemAttestation

This command computes attestations to verify the authenticity and identity of the CSP-enabled SE platform.

For *Platform Attestation* it uses the CASD authority signature according to [GP Amd A] and for *Config Attestation* it uses the configAttestationKey configured by the CSP Admin.

Note: The operation behaves similarly to att.computeAttestation, but is dedicated to *System Attestations*.

For further information, see section 5.1.3, System Operations.

ASN 7-28: Attestations: ASN.1 Definition for CSPSystemAttestation

```
-- ASN1START
```

```
-- Command to retrieve signed attestation data of the platform and the CSP.
CSPSystemAttestation ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- CASD-based CSP platform or CSP-specific config attestation.
    attestationType         CSPSystemAttestationType,

    -- Challenge for verification of attestation integrity.
    challenge                CSPChallenge OPTIONAL
}
-- ASN1STOP
```

This command returns a Status Word (SW) with the signed attestation data specified below:

- 9000: Success
- ERROR_ILLEGAL_CONFIG:
 - No CASD attestation configured [0x3006] or inconsistent cryptographic configurations [0x3007].
 - No config attestation key configured [0x3008], inconsistent signature configuration [0x3020], or invalid field configuration [0x30E0].
 - Missing Resource [0x3002] or resource not initialized [0x3003].
- ERROR_NOT_ALLOWED:
 - Config attestation key missing ACCESS_USE [0x5007].
 - Resource is not configured for USAGE_ATTESTATION [0x5060], or is in STATE_EXHAUSTED [0x50A0] or STATE_EXPIRED [0x50B1].
- ERROR_NOT_SUPPORTED:
 - Padding [0x8011], message digest [0x8021], or signature algorithm [0x8022] not supported.

ASN 7-29: Attestations: ASN.1 Definition for CSPSystemAttestationResponse

```
-- ASN1START
-- Response of the CSPSystemAttestation command.
CSPSystemAttestationResponse ::= CHOICE {

    -- Conditional: Response of the CASD-based SE platform attestation.
    platformAttestation      CSPPPlatformAttestation,

    -- Conditional: Response of the CSP-specific config attestation.
    configAttestation         CSPConfigAttestation
}
-- ASN1STOP
```

7.3.3 CSPGenerateKey

This command randomly generates the cryptographic value of a symmetric or private key resource for *Key Generation*, transitioning the resource to STATE_OPERATIONAL. Corresponding public keys can be computed using the CSPComputePublicKey command.

The command also resets associated timers and counters to their initial values, such as a usageCounter.

It supports KEY_AES, KEY_HMAC, KEY_ECC_PRIVATE, KEY_RSA_PRIVATE, and KEY_MASTER_SECRET, but does not support KEY_DERIVED_SECRET and KEY_SHARED_SECRET.

The resource must be in STATE_UNINITIALIZED with ADMIN_SETUP right granted.

It does not allocate additional memory, as memory allocation is already completed during resource creation.

Note: The operation behaves exactly the same as key.generate.

For further information, see section 6.7.3, Key Operations.

ASN 7-30: Key: ASN.1 Definition for CSPGenerateKey

```
-- ASN1START
-- Command for key generation.
CSPGenerateKey ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- The key for which the value should be generated.
    keyResourceId           CSPResourceId
}
-- ASN1STOP
```

This command returns a Status Word (SW) without any additional data:

- 9000: Success
 - Ignore policy rules.
- ERROR_ILLEGAL_VALUE:
 - Resource identifier does not exist [0x2001].
- ERROR_ILLEGAL_CONFIG:
 - Resource already initialized [0x3004].
- ERROR_NOT_ALLOWED:
 - Resource missing ACCESS_SETUP [0x5008], or is in STATE_EXHAUSTED [0x50A0] or STATE_EXPIRED [0x50B1].

7.3.4 CSPComputePublicKey

This command calculates the cryptographic value of a public key from a given initialized private key, transitioning the public key to STATE_OPERATIONAL as part of a *Key Generation*.

The command also resets associated timers and counters of the public key to their initial values.

It supports the following key pair combinations:

- KEY_ECC_PUBLIC, KEY_ECC_PRIVATE
- KEY_RSA_PUBLIC, KEY_RSA_PRIVATE

The private key must be in STATE_OPERATIONAL with ADMIN_USE, and the public key in STATE_UNINITIALIZED with ADMIN_SETUP right granted.

It does not allocate additional memory, as memory allocation is already completed during resource creation.

Note: The operation behaves exactly the same as `key.computePublicKey`.

For further information, see section 6.7.3, Key Operations.

ASN 7-31: Key: ASN.1 Definition for CSPComputePublicKey

```
-- ASN1START
-- Command to compute a public key from its private key.
CSPComputePublicKey ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- The corresponding private key that is already initialized.
    privateResourceId       CSPResourceId,

    -- The public key that shall be computed.
    publicResourceId        CSPResourceId
}
-- ASN1STOP
```

This command returns a status word (SW) without any additional data:

- 9000: Success
 - Ignore policy rules.
- ERROR_ILLEGAL_VALUE:
 - A resource identifier does not exist [0x2001].
 - First resource is not a private key [0x2072] or second is not a public key [0x2073].
- ERROR_ILLEGAL_CONFIG:
 - Private key not initialized [0x3003] or public key already initialized [0x3004].
- ERROR_NOT_ALLOWED:
 - Private key missing ACCESS_USE [0x5007], or is in STATE_EXHAUSTED [0x50A0] or STATE_EXPIRED [0x50B1].
 - Public key missing ACCESS_SETUP [0x5008].

7.3.5 CSPDeriveKey

This command derives a new key value from a source secret or password, transitioning the destination resource to STATE_OPERATIONAL.

The command also resets associated timers and counters of the destination resource to their initial values.

It supports PWD_STRONG and KEY_SHARED_SECRET as sources for *Key Derivation*. The type of the derived key depends on the *Key Derivation Algorithms* configured to the source resource.

The source must be in STATE_OPERATIONAL with ADMIN_USE, and the derived key in STATE_UNINITIALIZED with ADMIN_SETUP right granted.

It does not allocate additional memory, as memory allocation is already completed during resource creation.

Note: The operation behaves exactly the same as key.derive.

For further information, see section 6.7.3, Key Operations.

ASN 7-32: Key: ASN.1 Definition for CSPDeriveKey

```
-- ASN1START
-- Command for key derivation.
CSPDeriveKey ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- The resource that is base source for key derivation.
    sourceResourceId        CSPResourceId,

    -- The destination resource to store the derived key value.
    destResourceId          CSPResourceId,

    -- Additional input data given into the derivation algorithm
    inputData               OCTET STRING (SIZE (16..512))
}
-- ASN1STOP
```

This command returns a Status Word (SW) without any additional data:

- 9000: Success
 - Ignore policy rules.
- ERROR_ILLEGAL_VALUE:
 - Resource identifier does not exist [0x2001].
- ERROR_ILLEGAL_CONFIG:
 - Source not initialized [0x3003] or destination already initialized [0x3004].
 - Inconsistent key derivation configuration [0x3070].
- ERROR_NOT_ALLOWED:
 - Source missing ACCESS_USE [0x5007], is not configured for USAGE_KEY [0x5070], or is in STATE_EXHAUSTED [0x50A0] or STATE_EXPIRED [0x50B1].
 - Destination missing ACCESS_SETUP [0x5008].
- ERROR_NOT_SUPPORTED:
 - Derivation algorithm [0x8074] not supported.

7.3.6 CSPSetTime

This command sets a new reference time to the CSP Instance used for *Time Synchronization*.

Invocations of this command can be logged, as specified for EVENT_CSP_TIME_SET.

Note: The operation behaves exactly the same as `time.setReferenceTime`.

For further information, see section 6.11.3, Time Operations.

ASN 7-33: Time: ASN.1 Definition for CSPSetTime

```
-- ASN1START
-- Command to set a new reference time to the CSP Instance.
CSPSetTime ::= SEQUENCE {

    -- Minimum API level required for the CSP Protocol.
    cspProtocolVersion      CSPProtocolVersion,

    -- The new reference time.
    newTime                  CSPTimestamp,

    -- Signature to verify the authenticity of the new reference time.
    signature                 CSPSignature OPTIONAL
}
-- ASN1STOP
```

This command returns a Status Word (SW) and the update result:

- 9000: Success
- ERROR_ILLEGAL_CONFIG:
 - No time verification key configured [0x30B0] or inconsistent signature configuration [0x3020].
 - Missing verification key [0x3002] or resource not initialized [0x3003].
- ERROR_NOT_ALLOWED:
 - Time verification key missing ACCESS_USE [0x5007], not configured for USAGE_SIGNATURE [0x50B0], or is in STATE_EXPIRED [0x50B1].
 - Time signature verification failed [0x50B3].
- ERROR_NOT_SUPPORTED:
 - System time is not supported, but mode is set to TIME_MODE_STRICT [0x80B2].
 - Padding [0x8011], message digest [0x8021], or signature algorithm [0x8022] not supported.

ASN 7-34: Attestations: ASN.1 Definition for CSPSetTimeResponse

```
-- ASN1START
-- Response of the CSPSetTime command: TRUE if time was successfully updated.
CSPSetTimeResponse ::= SEQUENCE {
    response      CSPBoolean
}
-- ASN1STOP
```

8 CSP API

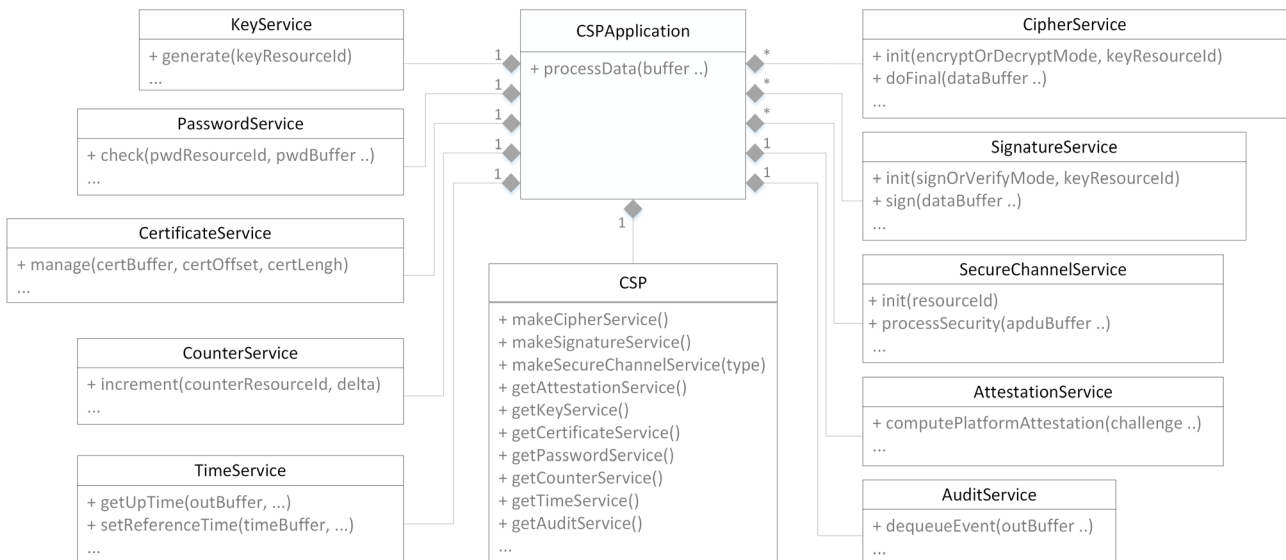
The CSP API is a Java Card interface that provides pre-defined cryptographic services to authorized on-card Client Applications ([GP CSP API]), as illustrated in Figure 8-1. This API offers various security-related operations described in section 3, *Features*, utilizing the GlobalPlatform technology defined in section 4, *Architecture and Instantiation*.

For supported *Modules*, the CSP shall implement the corresponding CSP API interfaces listed in Table 5-2.

The CSP shall adhere to the method definitions specified in [GP CSP API].

For further information, see section 3, *Features*, section 4, *Architecture*, and section 6, *Optional Modules*.

Figure 8-1: CSP API



8.1 Prerequisite for Using the CSP API

A Client Application should utilize its own CSP Application, a dedicated instance of the CSP ELF tailored to a specific use case. The decision to instantiate a new CSP Application shall be based on the required level of trust. Client Applications using the same CSP Instance are considered trustworthy, as they may share cryptographic services within the same CSP Application.

The CSP Application shall provide exactly one CSP Instance through a Global Service. This CSP Instance is an object instance of the `org.globalplatform.csp.api.CSP` interface. It is accessible only to authorized Client Applications previously registered by the CSP Admin.

CSP API operations shall only be usable when the corresponding CSP Configuration has been activated by the CSP Admin. This restriction prevents issues arising from incomplete resource configuration and ensures that all necessary cryptographic resources, such as keys and passwords, are pre-configured for the cryptographic services of the CSP API.

In summary, the steps before the CSP API can be used by Client Applications are as follows:

1. The SE Admin instantiates a new CSP Application.
2. The CSP Admin registers the Client Application(s).
3. The CSP Admin creates and configures cryptographic resources and algorithms required.
4. The CSP Admin activates the CSP configuration.

For further information, see section 5.1.4, System Lifecycle.

8.2 Retrieve the CSP Instance

The CSP shall offer a Global Service according to [GP Card Spec], allowing Client Applications to retrieve an instance of the `org.globalplatform.csp.api.CSP` interface. This instance serves as the primary entry point for accessing or creating cryptographic services provided by the CSP. A code sample demonstrating how to retrieve this CSP Instance, referred to as 'csp', is provided in Sample 8-1.

Sample 8-1: Retrieve CSP Instance

```
// AID of the CSP Instance.
AID cspAID = JCSYSTEM.lookupAID(CSP_AID, (short) 0, (byte) CSP_AID.length);

// Retrieve the Global Service for the CSP Instance.
GlobalService cspGlobalService = GPSystem.getService(cspAID,
                                                    CSP.GLOBAL_SERVICE_ID);

// AID of the Client Application that wants to use CSP services.
AID clientAID = JCSYSTEM.getAID();

// Retrieve the Registry Entry of the Client Application.
GPRegistryEntry clientRegistryEntry = GPSystem.getRegistryEntry(clientAID);

// Retrieve the CSP Shareable Object; performs access control checks.
CSP csp = (CSP) cspGlobalService.getServiceInterface(clientRegistryEntry,
                                                    CSP.SERVICE_ID, null, (short) 0, (short) 0);
```

The CSP shall perform the following access control checks when a Client Application invokes the `org.globalplatform.GlobalService.getServiceInterface` operation to ensure that only authorized Client Applications can access the CSP Instance:

- **AID Registration Check:** The CSP shall verify that the AID of the Client Application has been registered with the CSP Instance.
- **Invoking AID Check:** The CSP shall verify that the AID of the invoking Application matches the AID of the `org.globalplatform.GPSystem.GPRegistryEntry` parameter provided during the method call.
- **SD AID Check:** The CSP shall verify that the AID of the SD associated with the Client Application, as indicated in the `GPRegistryEntry`, matches the AID of the Client's SD if this check is configured.
- **DAP Verification:** The CSP shall ensure that the Client Application was verified during installation using the Data Authentication Pattern (DAP) ([GP Card Spec]), if required.
- **Hash Verification:** The CSP shall verify that the Load File Data Block Hash (LFDBH) of the Client Application's ELF ([GP Card Spec]) matches the hash configured by the CSP Admin.

All these checks can be configured by the CSP Admin through `CSPRegisterClient`.

To ensure proper access control and compliance with security requirements, CSP implementations must adhere to the following rules:

- The `GPSystem.getService` method shall not return a CSP Instance under any circumstances.
- Only the `GlobalService.getServiceInterface` method shall return the CSP Instance.

- The CSP implementation of the GlobalService interface shall not be an instance of `org.globalplatform.csp.api.CSP`.
- It shall not be possible to bypass the access control checks required to authorize Client Applications.

For further information, see section 3.2.1, Client Application Registration, and section 4.4.2, Global Service ID.

8.3 Byte Buffer Parameters

The CSP API uses byte arrays to transfer data (e.g., data to encrypt or sign) between the Client Application and the CSP Application. However, Applications on a Secure Element are isolated and cannot share byte arrays directly. Thus, arrays and other objects are restricted to the Application that created them unless explicitly shared through controlled mechanisms.

Furthermore, the CSP requires the use of *Sensitive Arrays*, which offer integrity protection based on `javacard.framework.SensitiveArrays` ([JCAPI]). To prepare these arrays, Client Applications can use the following mechanisms:

- **APDU Buffer** (all Java Card versions): The APDU buffer is a shared memory area provided by the Java Card Runtime Environment (JCRE) and accessible via `javacard.framework.APDU` ([JCAPI]). To add integrity protection, Client Application must invoke `org.globalplatform.csp.CSPSensitiveArrays.makeIntegritySensitiveArrayView(...)` ([GP CSP API]) on the APDU buffer before using it as byte buffer parameter in the CSP API.
- **Array Views** (Java Card 3.1): Array Views allow controlled sharing of arrays with defined access rights (e.g., read-only). To prepare an array for sharing, Client Application must first create an integrity-protected array via `javacard.framework.SensitiveArrays.makeIntegritySensitiveArray(...)` ([JCAPI]), copy the data to be shared into this array and then grant controlled access to the CSP via `javacard.framework.JCSystem.makeArrayView(...)` ([JCAPI]).

The Java Card version supported by the platform is specified in the corresponding GlobalPlatform configuration document related to the CSP.

Note: The `javacard.framework.JCSystem.makeGlobalArray(...)` method, which creates global arrays accessible to all Applications on the SE [JCAPI], may not be suitable to share data between CSP and Client Applications. According to [JCRE] section 6.2.2, global arrays cannot be stored in class member variables. As a result, Client Applications would need to create a new global array for each CSP invocation that requires buffers. Combined with garbage collection, this could lead to performance issues, making global arrays impractical for regular use.

For further information, see section 5.1.3.4, Sensitive Arrays.

8.4 Listener Mechanism

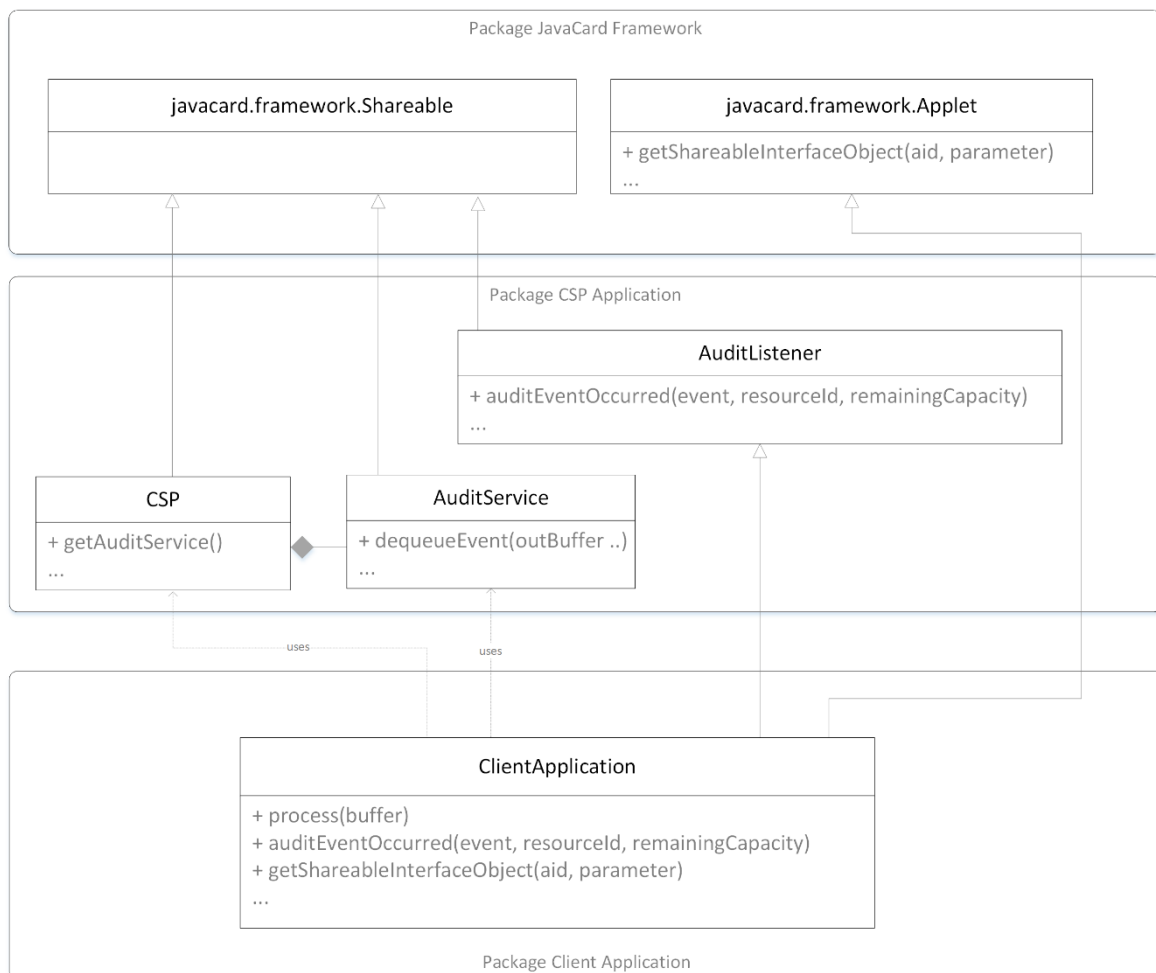
The CSP offers a listener mechanism through the CSP API, allowing a Client Application to be actively notified by the CSP. To enable this, the Client Application shall implement specific callback functions, register them as a Shareable Interface Object (SIO) and provide them for invocation by the CSP as illustrated in Figure 8-2. The Client Application shall return an instance of this listener implementation using the `javacard.framework.Applet#getShareableInterfaceObject` operation ([JCAPI]) with:

- Sharable parameter `0x0C`: Returns `org.globalplatform.csp.api.AuditListener`.

If the AuditModule is supported, the CSP shall retrieve the listener SIO from the Client Application using `javacard.framework.JCSystem#getAppletShareableInterfaceObject` and handles it as specified in Table 6-127: Audit Listener.

Note: If multiple Client Applications are registered to the same CSP Instance, each providing a listener instance, all listeners will be notified.

Figure 8-2: CSP API Listener



Annex A CUSTOM SECURE MESSAGING (INFORMATIVE)

This informative annex lists secure channel protocols that can be constructed using the CSP's cipher and key management services, including:

- ECKA-DH-AES: Combine Elliptic Curve Diffie-Hellman (ECDH) key agreement with AES cipher.
- ECKA-EG-AES: Combine Curve Key Agreement ElGamal (ECKA-EG) with AES cipher.
- ECIES: Implement the Elliptic Curve Integrated Encryption Scheme (ECIES).
- MDL: Implement the Mobile Driving Licence (MDL) standards by using ECDH key agreement and ECDSA (Elliptic Curve Digital Signature Algorithm).
- FIDO2: Use ECKA-DH key agreement to implement Fast IDentity Online (FIDO2).

Note: When using custom secure messaging, the confidential data transfer feature of the CSP cannot be used and must be replaced with the CSP's cipher service to encrypt sensitive data for safe transfer.

For further information, see section 3.7.2, Built-in Secure Channel Protocols, and section 3.7.3, Building Blocks.

A.1 ECDH-AES / ECKA-DH-AES

Elliptic Curve Diffie-Hellman (ECDH) key agreement, also known as ECKA-DH (Elliptic Curve Key Agreement Diffie-Hellman) uses elliptic curve cryptography to securely establish a shared secret between two parties. When integrated with AES for cipher operations, the ECKA-DH becomes a tool for developing bespoke secure messaging solutions within a Client Application. It requires the following CSP resources:

- ECC Key Pair: This static key pair, once generated or imported by the CSP Admin, remains unchanged and is used as the private and public keys for ECKA-DH.
- Shared Secret: A temporary resource representing the shared secret, which can then be used as input for further key derivation operations.
- AES Key: The session-specific AES key, crucial for the encryption and decryption of messages, is derived from the Master Key using Key Derivation Functions (KDF) provided by the CSP.

For further information, see section 3.3.3.1, ECKA-DH Key Agreement, section 6.7.1.6, Key Derivation Algorithms, section 6.7.1.3, ECC Curves, and section 6.1.1.3, Cipher Algorithms.

A.2 ECKA-EG-AES

Elliptic Curve Key Agreement using the ElGamal algorithm (ECKA-EG) is a key agreement process that, combined with AES encryption, enables Client Applications to develop tailored secure communication channels. Unlike the Diffie-Hellman method, the initiating and responding parties require different input parameters when using the ElGamal scheme. Key resource definitions essential for this process are:

- Static ECC Key Pair: Once generated or imported by the CSP Admin, this key pair remains unchanged and is utilized for initializing ECKA-EG agreements.
- Ephemeral ECC Key Pair: This key pair is freshly generated for each session when the CSP is the initiator, thus enhancing security through forward secrecy.
- Remote ECC Public Key: A resource with ACCESS_MOVE right that enables the Client Application to import the initiator's ephemeral public key.
- Shared Secret: A temporary resource representing the shared secret, which can then be used as input for further key derivation operations.

- **AES Key:** The session-specific AES key, crucial for the encryption and decryption of messages, is derived from the Ephemeral ECC Key Pair using Key Derivation Functions (KDF) provided by the CSP.

For further information, see section 3.3.3.2, ECKA-EG Key Agreement, section 6.7.1.6, Key Derivation Algorithms, section 6.7.1.3, ECC Curves, and section 6.1.1.3, Cipher Algorithms.

A.3 ECIES

Elliptic Curve Integrated Encryption Scheme (ECIES) is a cryptographic framework that combines elliptic curve cryptography for secure key exchange with symmetric encryption and message authentication for data protection. It enables the secure encryption and decryption of messages between two parties without the need for a shared secret key in advance.

ECIES involves several cryptographic steps:

1. **Key Pair Generation (Sender):** The sender generates a temporary elliptic curve public/private key pair (e.g., using the P-256 curve) specifically for this session.
2. **Public Key Transmission:** The sender transmits the temporary public key to the receiver.
3. **Shared Secret Calculation based on Elliptic Curve Diffie-Hellman (ECDH):**
 - a. **Receiver Side:** The receiver uses the sender's public key and their own private key to calculate a shared secret through elliptic curve multiplication.
 - b. **Sender Side:** The sender calculates the same shared secret using the receiver's public key and their temporary private key.
4. **Key Derivation:** Both parties use a Key Derivation Function (KDF), such as HKDF, to derive a symmetric encryption key from the shared secret.
5. **Encryption (Sender):** The sender encrypts the message using the derived symmetric key with a symmetric encryption algorithm (e.g., AES-GCM).
6. **MAC Calculation (Sender):** The sender computes a Message Authentication Code (MAC) on the encrypted message using an algorithm like HMAC-SHA-256 to ensure integrity and authenticity.
7. **Message and MAC Transmission:** The sender transmits both the encrypted message and the MAC to the receiver.
8. **Decryption and MAC Verification (Receiver):** The receiver decrypts the message using the symmetric key and verifies the MAC with HMAC-SHA-256 to ensure the message's integrity and authenticity.

The required CSP resources are as follows:

- **ECC Key Pair:** This static key pair, once generated or imported by the CSP Admin, remains unchanged and is used as the private and public keys for ECKA-DH.
- **Shared Secret:** A temporary resource representing the shared secret, which can then be used as input for further key derivation operations.
- **AES Key:** The session-specific AES key, crucial for the encryption and decryption of messages, is derived from the Master Key using Key Derivation Functions (KDF) provided by the CSP.
- **HMAC Key:** The session-specific HMAC key, used to compute and verify the MAC on the encrypted message.

A.4 Mobile Driving Licence

A Mobile Driving Licence (MDL) represents a secure, electronic form of traditional driver's license information specified in [ISO 18013-5]. It utilizes Elliptic Curve Diffie-Hellman (ECDH) for secure key agreement, ensuring that the exchange of information between the user's device and the licensing authority is protected. For the crucial aspects of authentication and integrity verification, the MDL employs the Elliptic Curve Digital Signature Algorithm (ECDSA). This algorithm is specifically used to digitally sign the MDL, establishing a robust link between the license and its cryptographic assurance of authenticity and integrity.

The MDL-Issuance process steps are as follows:

1. **Key Agreement (ECDH):** The user's device and the Licensing Authority (LA) securely exchange public keys using the Elliptic Curve Diffie-Hellman (ECDH) algorithm to establish a shared secret. This shared secret is used to encrypt the communication during the registration process, ensuring that sensitive information (personal identification data, photographs, etc.) is transmitted securely.
2. **Signing the MDL (ECDSA):** Once the user's information is verified, the LA signs the digital version of the driver's license using its private ECC key. This digital signature ensures the authenticity and integrity of the MDL.
3. **Encryption for Issuance:** The MDL data, along with the LA's digital signature, is encrypted using the shared secret (established via ECDH) before being transmitted to the user's device, ensuring secure delivery.
4. **MDL Activation:** Upon receiving the MDL, the user's device uses the shared secret to decrypt the data. The LA's public key is then used to verify the digital signature on the MDL, confirming its authenticity.
5. **Secure Storage:** The MDL is securely stored on the device, typically in a secure element or isolated storage area, to prevent unauthorized access and tampering.

The MDL-Presentation process steps are as follows:

1. **Key Agreement (ECDH):** When presenting the MDL, a secure, ephemeral key agreement (ECDH) is established between the user's device and the verifier's device to encrypt the communication.
2. **MDL Verification (ECDSA):** The verifier uses the LA's public key to check the digital signature on the MDL, ensuring it is valid and has not been tampered with.

The required CSP resources are as follows:

- **User ECC Key Pair:** Used for secure communication (ECDH) and possibly for signing requests or data related to the user's identity.
- **User Signing ECC Key Pair:** Specifically designated for signing requests or data by the user using ECDSA algorithm.
- **Ephemeral ECC Key Pairs for ECDH:** Temporary key pairs generated for secure key agreement during MDL presentation.

For further information, see section 3.3.3.1, ECKA-DH Key Agreement, section 6.7.1.6, Key Derivation Algorithms, section 6.7.1.3, ECC Curves, and section 6.2.1.3, Signature Algorithms.

A.5 FIDO2

Fast IDentity Online (FIDO2) is an open authentication standard that aims to enhance the security of online identities. It incorporates the Client To Authenticator Protocol (CTAP) [FIDO2-CTAP], which facilitates communication between a client device, such as a web browser, and an external authenticator, for example, a hardware security token.

CTAP utilizes Elliptic Curve Key Agreement-Diffie-Hellman (ECKA-DH) for secure key agreement between the client (a remote entity) and the authenticator, such as a CSP-enabled Secure Element, as specified in [FIDO2-CTAP].

For further information, see section 3.3.3.1, ECKA-DH Key Agreement, section 6.7.1.6, Key Derivation Algorithms, section 6.7.1.3, ECC Curves, and section 6.1.1.3, Cipher Algorithms.