

Key Management Interoperability Protocol Specification Version 3.0

Committee Specification Draft 01

23 August 2024

This stage:

<https://docs.oasis-open.org/kmip/kmip-spec/v3.0/csd01/kmip-spec-v3.0-csd01.>docx (Authoritative)

<https://docs.oasis-open.org/kmip/kmip-spec/v3.0/csd01/kmip-spec-v3.0-csd01.html>

<https://docs.oasis-open.org/kmip/kmip-spec/v3.0/csd01/kmip-spec-v3.0-csd01.pdf>

Previous stage:

N/A

This stage:

[https://docs.oasis-open.org/kmip/kmip-spec/v3.0/kmip-spec-v3.0.docx](https://docs.oasis-open.org/kmip/kmip-spec/v3.0/kmip-spec-v3.0.docx%20%20) (Authoritative)

<https://docs.oasis-open.org/kmip/kmip-spec/v3.0/kmip-spec-v3.0.html>

<https://docs.oasis-open.org/kmip/kmip-spec/v3.0/kmip-spec-v3.0.pdf>

Technical Committee:

[OASIS Key Management Interoperability Protocol (KMIP) TC](https://www.oasis-open.org/committees/kmip/)

Chairs:

Greg Scott ([greg.scott@cryptsoft.com](mailto:greg.scott@cryptsoft.com)), [Cryptsoft Pty Ltd.](http://www.cryptsoft.com)

Judith Furlong ([Judith.Furlong@dell.com](mailto:Judith.Furlong@dell.com)), [Dell](http://www.dell.com)

Editors:

Greg Scott ([greg.scott@cryptsoft.com](mailto:greg.scott@cryptsoft.com)), [Cryptsoft Pty Ltd.](http://www.cryptsoft.com)

Tony Cox ([tony.cox@tclogic.com.au](mailto:tony.cox@tclogic.com.au)), TC Logic

Related work:

This specification replaces or supersedes:

* *Key Management Interoperability Protocol Specification Version 2.1*. Edited by Tony Cox and Charles White. OASIS Standard. Latest stage: <https://docs.oasis-open.org/kmip/kmip-spec/v2.1/kmip-spec-v2.1.html>

This specification is related to:

* *Key Management Interoperability Protocol Profiles Version 3.0.* Edited by Tim Chevalier and Tim Hudson. Latest stage: <https://docs.oasis-open.org/kmip/kmip-profiles/v3.0/kmip-profiles-v3.0.html>
* *Key Management Interoperability Protocol Test Cases Version 3.0.* Work in Progress.
* *Key Management Interoperability Protocol Usage Guide Version 3.0.* Work in Progress.

Abstract:

This document is intended for developers and architects who wish to design systems and applications that interoperate using the Key Management Interoperability Protocol Specification.

Status:

This document was last revised or approved by the OASIS Key Management Interoperability Protocol (KMIP) TC on the above date. The level of approval is also listed above. Check the “Latest stage” location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at <https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=kmip#technical>.

Comments from TC members should be sent directly to the TC's mailing list. Comments may be submitted to the project by any other person through the use of the project’s Comment Facility: https://groups.oasis-open.org/communities/community-home?CommunityKey=2b5e5c66-cc41-4aa5-92ee-018f5aa7dfc4

This specification is provided under the [RF on RAND Terms](https://www.oasis-open.org/policies-guidelines/ipr#RF-on-RAND-Mode) Mode of the [OASIS IPR Policy](https://www.oasis-open.org/policies-guidelines/ipr), the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC’s web page (<https://www.oasis-open.org/committees/kmip/ipr.php>).

Key words:

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [[RFC2119](#RFC2119)] and [[RFC8174](#RFC8174)] when, and only when, they appear in all capitals, as shown here.

Note that any machine-readable content ([Computer Language Definitions](https://www.oasis-open.org/policies-guidelines/tc-process#wpComponentsCompLang)) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

When referencing this specification, the following citation format should be used:

[kmip-spec-v3.0]

*Key Management Interoperability Protocol Specification Version 3.0*. Edited by Greg Scott and Tony Cox. 23 August 2024. Committee Specification Draft 01. <https://docs.oasis-open.org/kmip/kmip-spec/v3.0/csd01/kmip-spec-v3.0-csd01.html>. Latest Stage: [https://docs.oasis-open.org/kmip/kmip-spec/v3.0/kmip-spec-v3.0.html.](https://docs.oasis-open.org/kmip/kmip-spec/v3.0/kmip-spec-v3.0.html)

Notices:

Copyright © OASIS Open 2024. All Rights Reserved.

Distributed under the terms of the OASIS IPR Policy, [<https://www.oasis-open.org/policies-guidelines/ipr/>]. For complete copyright information please see the full Notices section in an Appendix below.

Notices

Copyright © OASIS Open 2024. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full [Policy](https://www.oasis-open.org/policies-guidelines/ipr) may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of [OASIS](https://www.oasis-open.org/), the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <https://www.oasis-open.org/policies-guidelines/trademark> for above guidance.

Table of Contents

[1 Introduction 13](#_Toc176875663)

[1.1 IPR Policy 13](#_Toc176875664)

[1.2 Terminology 13](#_Toc176875665)

[1.3 Normative References 16](#_Toc176875666)

[1.4 Non-Normative References 19](#_Toc176875667)

[1.5 Item Data Types 20](#_Toc176875668)

[2 Objects 21](#_Toc176875669)

[2.1 System Objects 21](#_Toc176875670)

[2.1.1 User 21](#_Toc176875671)

[2.1.2 Group 21](#_Toc176875672)

[2.1.3 Credentials 22](#_Toc176875673)

[2.2 User Objects 24](#_Toc176875674)

[2.2.1 Certificate 24](#_Toc176875675)

[2.2.2 Certificate Request 25](#_Toc176875676)

[2.2.3 Opaque Object 25](#_Toc176875677)

[2.2.4 PGP Key 26](#_Toc176875678)

[2.2.5 Private Key 27](#_Toc176875679)

[2.2.6 Public Key 27](#_Toc176875680)

[2.2.7 Secret Data 27](#_Toc176875681)

[2.2.8 Split Key 28](#_Toc176875682)

[2.2.9 Symmetric Key 29](#_Toc176875683)

[3 Object Data Structures 30](#_Toc176875684)

[3.1 Key Block 30](#_Toc176875685)

[3.2 Key Value 31](#_Toc176875686)

[3.3 Key Wrapping Data 31](#_Toc176875687)

[3.4 Transparent Symmetric Key 33](#_Toc176875688)

[3.5 Transparent DSA Private Key 33](#_Toc176875689)

[3.6 Transparent DSA Public Key 33](#_Toc176875690)

[3.7 Transparent RSA Private Key 33](#_Toc176875691)

[3.8 Transparent RSA Public Key 34](#_Toc176875692)

[3.9 Transparent DH Private Key 34](#_Toc176875693)

[3.10 Transparent DH Public Key 34](#_Toc176875694)

[3.11 Transparent EC Private Key 35](#_Toc176875695)

[3.12 Transparent EC Public Key 35](#_Toc176875696)

[4 Object Attributes 36](#_Toc176875697)

[4.1 Activation Date 37](#_Toc176875698)

[4.2 Alternative Name 38](#_Toc176875699)

[4.3 Always Sensitive 38](#_Toc176875700)

[4.4 Application Specific Information 39](#_Toc176875701)

[4.5 Archive Date 40](#_Toc176875702)

[4.6 Certificate Attributes 40](#_Toc176875703)

[4.7 Certificate Type 41](#_Toc176875704)

[4.8 Certificate Length 42](#_Toc176875705)

[4.9 Comment 42](#_Toc176875706)

[4.10 Compromise Date 43](#_Toc176875707)

[4.11 Compromise Occurrence Date 43](#_Toc176875708)

[4.12 Contact Information 44](#_Toc176875709)

[4.13 Counters 44](#_Toc176875710)

[4.13.1 Certify Counter 44](#_Toc176875711)

[4.13.2 Decrypt Counter 45](#_Toc176875712)

[4.13.3 Encrypt Counter 45](#_Toc176875713)

[4.13.4 Sign Counter 46](#_Toc176875714)

[4.13.5 Signature Verify Counter 46](#_Toc176875715)

[4.14 Credential Type 47](#_Toc176875716)

[4.15 Cryptographic Algorithm 47](#_Toc176875717)

[4.16 Cryptographic Domain Parameters 48](#_Toc176875718)

[4.17 Cryptographic Length 49](#_Toc176875719)

[4.18 Cryptographic Parameters 49](#_Toc176875720)

[4.19 Cryptographic Usage Mask 51](#_Toc176875721)

[4.20 Deactivation Date 51](#_Toc176875722)

[4.21 Deactivation Reason 52](#_Toc176875723)

[4.22 Description 53](#_Toc176875724)

[4.23 Destroy Date 53](#_Toc176875725)

[4.24 Digest 53](#_Toc176875726)

[4.25 Digital Signature Algorithm 54](#_Toc176875727)

[4.26 Extractable 55](#_Toc176875728)

[4.27 Fresh 55](#_Toc176875729)

[4.28 Initial Date 56](#_Toc176875730)

[4.29 Key Format Type 56](#_Toc176875731)

[4.30 Key Part Identifier 57](#_Toc176875732)

[4.31 Key Value Location 58](#_Toc176875733)

[4.32 Key Value Present 58](#_Toc176875734)

[4.33 Last Change Date 59](#_Toc176875735)

[4.34 Lease Time 59](#_Toc176875736)

[4.35 Links 60](#_Toc176875737)

[4.35.1 Certificate Link 61](#_Toc176875738)

[4.35.2 Certificate Request Link 61](#_Toc176875739)

[4.35.3 Child Link 62](#_Toc176875740)

[4.35.4 Credential Link 62](#_Toc176875741)

[4.35.5 Derivation Base Object Link 63](#_Toc176875742)

[4.35.6 Derived Object Link 63](#_Toc176875743)

[4.35.7 Group Link 64](#_Toc176875744)

[4.35.8 Joined Split Key Parts Link 64](#_Toc176875745)

[4.35.9 Next Link 65](#_Toc176875746)

[4.35.10 Parent Link 65](#_Toc176875747)

[4.35.11 Password Link 66](#_Toc176875748)

[4.35.12 PKCS#12 Certificate Link 67](#_Toc176875749)

[4.35.13 PKCS#12 Password Link 67](#_Toc176875750)

[4.35.14 Previous Link 68](#_Toc176875751)

[4.35.15 Private Key Link 68](#_Toc176875752)

[4.35.16 Public Key Link 69](#_Toc176875753)

[4.35.17 Replaced Object Link 69](#_Toc176875754)

[4.35.18 Replacement Object Link 70](#_Toc176875755)

[4.35.19 Split Key Base Link 70](#_Toc176875756)

[4.35.20 Wrapping Key Link 71](#_Toc176875757)

[4.36 Name 71](#_Toc176875758)

[4.37 Never Extractable 72](#_Toc176875759)

[4.38 NIST Key Type 72](#_Toc176875760)

[4.39 NIST Security Category 73](#_Toc176875761)

[4.40 Object Class 73](#_Toc176875762)

[4.41 Object Type 74](#_Toc176875763)

[4.42 Opaque Data Type 74](#_Toc176875764)

[4.43 Original Creation Date 75](#_Toc176875765)

[4.44 OTP Counter 75](#_Toc176875766)

[4.45 PKCS#12 Friendly Name 76](#_Toc176875767)

[4.46 Process Start Date 76](#_Toc176875768)

[4.47 Protect Stop Date 77](#_Toc176875769)

[4.48 Protection Level 78](#_Toc176875770)

[4.49 Protection Period 78](#_Toc176875771)

[4.50 Protection Storage Mask 78](#_Toc176875772)

[4.51 Quantum Safe 79](#_Toc176875773)

[4.52 Random Number Generator 79](#_Toc176875774)

[4.53 Revocation Reason 80](#_Toc176875775)

[4.54 Rotate Automatic 81](#_Toc176875776)

[4.55 Rotate Date 81](#_Toc176875777)

[4.56 Rotate Generation 82](#_Toc176875778)

[4.57 Rotate Interval 82](#_Toc176875779)

[4.58 Rotate Latest 83](#_Toc176875780)

[4.59 Rotate Name 83](#_Toc176875781)

[4.60 Rotate Offset 84](#_Toc176875782)

[4.61 Sensitive 84](#_Toc176875783)

[4.62 Short Unique Identifier 85](#_Toc176875784)

[4.63 Split Key Polynomial 85](#_Toc176875785)

[4.64 Split Key Method 86](#_Toc176875786)

[4.65 Split Key Parts 86](#_Toc176875787)

[4.66 Split Key Threshold 87](#_Toc176875788)

[4.67 State 87](#_Toc176875789)

[4.68 Unique Identifier 90](#_Toc176875790)

[4.69 Usage Limits 92](#_Toc176875791)

[4.70 Vendor Attribute 92](#_Toc176875792)

[4.71 X.509 Certificate Identifier 93](#_Toc176875793)

[4.72 X.509 Certificate Issuer 94](#_Toc176875794)

[4.73 X.509 Certificate Subject 94](#_Toc176875795)

[5 Attribute Data Structures 96](#_Toc176875796)

[5.1 Attributes 96](#_Toc176875797)

[5.2 Common Attributes 96](#_Toc176875798)

[5.3 Private Key Attributes 96](#_Toc176875799)

[5.4 Public Key Attributes 96](#_Toc176875800)

[5.5 Attribute Reference 97](#_Toc176875801)

[5.6 Current Attribute 97](#_Toc176875802)

[5.7 New Attribute 97](#_Toc176875803)

[6 Operations 98](#_Toc176875804)

[6.1 Client-to-Server Operations 98](#_Toc176875805)

[6.1.1 Activate 99](#_Toc176875806)

[6.1.2 Add Attribute 99](#_Toc176875807)

[6.1.3 Adjust Attribute 100](#_Toc176875808)

[6.1.4 Archive 101](#_Toc176875809)

[6.1.5 Cancel 102](#_Toc176875810)

[6.1.6 Certify 102](#_Toc176875811)

[6.1.7 Check 103](#_Toc176875812)

[6.1.8 Create 105](#_Toc176875813)

[6.1.9 Create Credential 106](#_Toc176875814)

[6.1.10 Create Group 107](#_Toc176875815)

[6.1.11 Create Key Pair 108](#_Toc176875816)

[6.1.12 Create Split Key 110](#_Toc176875817)

[6.1.13 Create User 111](#_Toc176875818)

[6.1.14 Deactivate 112](#_Toc176875819)

[6.1.15 Decrypt 113](#_Toc176875820)

[6.1.16 Delegated Login 115](#_Toc176875821)

[6.1.17 Delete Attribute 115](#_Toc176875822)

[6.1.18 Derive Key 116](#_Toc176875823)

[6.1.19 Destroy 118](#_Toc176875824)

[6.1.20 Discover Versions 118](#_Toc176875825)

[6.1.21 Encrypt 119](#_Toc176875826)

[6.1.22 Export 121](#_Toc176875827)

[6.1.23 Get 122](#_Toc176875828)

[6.1.24 Get Attributes 123](#_Toc176875829)

[6.1.25 Get Attribute List 124](#_Toc176875830)

[6.1.26 Get Constraints 125](#_Toc176875831)

[6.1.27 Get Usage Allocation 126](#_Toc176875832)

[6.1.28 Hash 126](#_Toc176875833)

[6.1.29 Import 127](#_Toc176875834)

[6.1.30 Interop 129](#_Toc176875835)

[6.1.31 Join Split Key 129](#_Toc176875836)

[6.1.32 Locate 130](#_Toc176875837)

[6.1.33 Log 133](#_Toc176875838)

[6.1.34 Login 133](#_Toc176875839)

[6.1.35 Logout 134](#_Toc176875840)

[6.1.36 MAC 135](#_Toc176875841)

[6.1.37 MAC Verify 136](#_Toc176875842)

[6.1.38 Modify Attribute 138](#_Toc176875843)

[6.1.39 Obliterate 139](#_Toc176875844)

[6.1.40 Obtain Lease 139](#_Toc176875845)

[6.1.41 Ping 140](#_Toc176875846)

[6.1.42 PKCS#11 141](#_Toc176875847)

[6.1.43 Poll 142](#_Toc176875848)

[6.1.44 Process 142](#_Toc176875849)

[6.1.45 Query 143](#_Toc176875850)

[6.1.46 Query Asynchronous Requests 146](#_Toc176875851)

[6.1.47 Recover 147](#_Toc176875852)

[6.1.48 Register 147](#_Toc176875853)

[6.1.49 Revoke 149](#_Toc176875854)

[6.1.50 Re-certify 150](#_Toc176875855)

[6.1.51 Re-key 151](#_Toc176875856)

[6.1.52 Re-key Key Pair 153](#_Toc176875857)

[6.1.53 Re-Provision 156](#_Toc176875858)

[6.1.54 RNG Retrieve 157](#_Toc176875859)

[6.1.55 RNG Seed 157](#_Toc176875860)

[6.1.56 Set Attribute 158](#_Toc176875861)

[6.1.57 Set Constraints 159](#_Toc176875862)

[6.1.58 Set Defaults 159](#_Toc176875863)

[6.1.59 Set Endpoint Role 160](#_Toc176875864)

[6.1.60 Sign 161](#_Toc176875865)

[6.1.61 Signature Verify 162](#_Toc176875866)

[6.1.62 Validate 164](#_Toc176875867)

[6.2 Server-to-Client Operations 165](#_Toc176875868)

[6.2.1 Discover Versions 165](#_Toc176875869)

[6.2.2 Notify 166](#_Toc176875870)

[6.2.3 Put 167](#_Toc176875871)

[6.2.4 Query 168](#_Toc176875872)

[6.2.5 Set Endpoint Role 170](#_Toc176875873)

[7 Operations Data Structures 171](#_Toc176875874)

[7.1 Asynchronous Correlation Values 171](#_Toc176875875)

[7.2 Asynchronous Request 171](#_Toc176875876)

[7.3 Authenticated Encryption Additional Data 171](#_Toc176875877)

[7.4 Authenticated Encryption Tag 171](#_Toc176875878)

[7.5 Capability Information 172](#_Toc176875879)

[7.6 Constraint 172](#_Toc176875880)

[7.7 Constraints 172](#_Toc176875881)

[7.8 Correlation Value 173](#_Toc176875882)

[7.9 Credential Information 173](#_Toc176875883)

[7.10 Data 173](#_Toc176875884)

[7.11 Data Length 173](#_Toc176875885)

[7.12 Defaults Information 174](#_Toc176875886)

[7.13 Derivation Parameters 174](#_Toc176875887)

[7.14 Extension Information 175](#_Toc176875888)

[7.15 Final Indicator 175](#_Toc176875889)

[7.16 Interop Function 175](#_Toc176875890)

[7.17 Interop Identifier 176](#_Toc176875891)

[7.18 Init Indicator 176](#_Toc176875892)

[7.19 Key Wrapping Specification 176](#_Toc176875893)

[7.20 Log Message 177](#_Toc176875894)

[7.21 MAC Data 177](#_Toc176875895)

[7.22 Objects 177](#_Toc176875896)

[7.23 Object Defaults 177](#_Toc176875897)

[7.24 Object Groups 178](#_Toc176875898)

[7.25 Object Types 178](#_Toc176875899)

[7.26 Operations 178](#_Toc176875900)

[7.27 PKCS#11 Function 178](#_Toc176875901)

[7.28 PKCS#11 Input Parameters 179](#_Toc176875902)

[7.29 PKCS#11 Interface 179](#_Toc176875903)

[7.30 PKCS#11 Output Parameters 179](#_Toc176875904)

[7.31 PKCS#11 Return Code 179](#_Toc176875905)

[7.32 Profile Information 179](#_Toc176875906)

[7.33 Profile Version 180](#_Toc176875907)

[7.34 Protection Storage Masks 180](#_Toc176875908)

[7.35 Right 180](#_Toc176875909)

[7.36 Rights 181](#_Toc176875910)

[7.37 RNG Parameters 181](#_Toc176875911)

[7.38 Server Information 182](#_Toc176875912)

[7.39 Signature Data 182](#_Toc176875913)

[7.40 Ticket 182](#_Toc176875914)

[7.41 Usage Limits 182](#_Toc176875915)

[7.42 Validation Information 183](#_Toc176875916)

[8 Messages 184](#_Toc176875917)

[8.1 Requests 184](#_Toc176875918)

[8.1.1 Request Message 184](#_Toc176875919)

[8.1.2 Request Header 184](#_Toc176875920)

[8.1.3 Request Batch Item 185](#_Toc176875921)

[8.2 Responses 185](#_Toc176875922)

[8.2.1 Response Message 185](#_Toc176875923)

[8.2.2 Response Header 185](#_Toc176875924)

[8.2.3 Response Batch Item 186](#_Toc176875925)

[9 Message Data Structures 187](#_Toc176875926)

[9.1 Asynchronous Correlation Value 187](#_Toc176875927)

[9.2 Asynchronous Indicator 187](#_Toc176875928)

[9.3 Attestation Capable Indicator 187](#_Toc176875929)

[9.4 Authentication 187](#_Toc176875930)

[9.5 Batch Error Continuation Option 188](#_Toc176875931)

[9.6 Batch Item 188](#_Toc176875932)

[9.7 Correlation Value (Client) 188](#_Toc176875933)

[9.8 Correlation Value (Server) 188](#_Toc176875934)

[9.9 Credential 189](#_Toc176875935)

[9.10 Maximum Response Size 191](#_Toc176875936)

[9.11 Message Extension 191](#_Toc176875937)

[9.12 Nonce 191](#_Toc176875938)

[9.13 Operation 191](#_Toc176875939)

[9.14 Protocol Version 192](#_Toc176875940)

[9.15 Result Message 192](#_Toc176875941)

[9.16 Result Reason 192](#_Toc176875942)

[9.17 Result Status 192](#_Toc176875943)

[9.18 Time Stamp 193](#_Toc176875944)

[10 Message Protocols 194](#_Toc176875945)

[10.1 TTLV 194](#_Toc176875946)

[10.1.1 Tag 194](#_Toc176875947)

[10.1.2 Type 194](#_Toc176875948)

[10.1.3 Length 195](#_Toc176875949)

[10.1.4 Value 195](#_Toc176875950)

[10.1.5 Padding 195](#_Toc176875951)

[10.2 Other Message Protocols 196](#_Toc176875952)

[10.2.1 HTTPS 196](#_Toc176875953)

[10.2.2 JSON 196](#_Toc176875954)

[10.2.3 XML 196](#_Toc176875955)

[10.3 Authentication 196](#_Toc176875956)

[10.4 Transport 196](#_Toc176875957)

[11 Enumerations 197](#_Toc176875958)

[11.1 Adjustment Type Enumeration 197](#_Toc176875959)

[11.2 Alternative Name Type Enumeration 197](#_Toc176875960)

[11.3 Asynchronous Indicator Enumeration 198](#_Toc176875961)

[11.4 Attestation Type Enumeration 198](#_Toc176875962)

[11.5 Batch Error Continuation Option Enumeration 198](#_Toc176875963)

[11.6 Block Cipher Mode Enumeration 199](#_Toc176875964)

[11.7 Cancellation Result Enumeration 200](#_Toc176875965)

[11.8 Certificate Request Type Enumeration 201](#_Toc176875966)

[11.9 Certificate Type Enumeration 201](#_Toc176875967)

[11.10 Client Registration Method Enumeration 201](#_Toc176875968)

[11.11 Credential Type Enumeration 202](#_Toc176875969)

[11.12 Cryptographic Algorithm Enumeration 202](#_Toc176875970)

[11.13 Data Enumeration 205](#_Toc176875971)

[11.14 Deactivation Reason Code Enumeration 205](#_Toc176875972)

[11.15 Derivation Method Enumeration 206](#_Toc176875973)

[11.16 Destroy Action Enumeration 207](#_Toc176875974)

[11.17 Digital Signature Algorithm Enumeration 207](#_Toc176875975)

[11.18 DRBG Algorithm Enumeration 208](#_Toc176875976)

[11.19 Encoding Option Enumeration 208](#_Toc176875977)

[11.20 Endpoint Role Enumeration 208](#_Toc176875978)

[11.21 Ephemeral Enumeration 209](#_Toc176875979)

[11.22 FIPS186 Variation Enumeration 209](#_Toc176875980)

[11.23 Hashing Algorithm Enumeration 210](#_Toc176875981)

[11.24 Interop Function Enumeration 210](#_Toc176875982)

[11.25 Item Type Enumeration 211](#_Toc176875983)

[11.26 Key Compression Type Enumeration 212](#_Toc176875984)

[11.27 Key Format Type Enumeration 212](#_Toc176875985)

[11.28 Key Role Type Enumeration 213](#_Toc176875986)

[11.29 Key Value Location Type Enumeration 214](#_Toc176875987)

[11.30 Key Wrap Type Enumeration 214](#_Toc176875988)

[11.31 Mask Generator Enumeration 215](#_Toc176875989)

[11.32 NIST Key Type Enumeration 215](#_Toc176875990)

[11.33 Object Class Enumeration 216](#_Toc176875991)

[11.34 Object Type Enumeration 216](#_Toc176875992)

[11.35 Opaque Data Type Enumeration 217](#_Toc176875993)

[11.36 Operation Enumeration 217](#_Toc176875994)

[11.37 OTP Algorithm Enumeration 219](#_Toc176875995)

[11.38 Padding Method Enumeration 219](#_Toc176875996)

[11.39 PKCS#11 Function Enumeration 219](#_Toc176875997)

[11.40 PKCS#11 Return Code Enumeration 219](#_Toc176875998)

[11.41 Processing Stage Enumeration 220](#_Toc176875999)

[11.42 Profile Name Enumeration 220](#_Toc176876000)

[11.43 Protection Level Enumeration 221](#_Toc176876001)

[11.44 Put Function Enumeration 221](#_Toc176876002)

[11.45 Query Function Enumeration 222](#_Toc176876003)

[11.46 Recommended Curve Enumeration 223](#_Toc176876004)

[11.47 Result Reason Enumeration 225](#_Toc176876005)

[11.48 Result Status Enumeration 230](#_Toc176876006)

[11.49 Revocation Reason Code Enumeration 230](#_Toc176876007)

[11.50 RNG Algorithm Enumeration 231](#_Toc176876008)

[11.51 RNG Mode Enumeration 231](#_Toc176876009)

[11.52 Secret Data Type Enumeration 231](#_Toc176876010)

[11.53 Shredding Algorithm Enumeration 232](#_Toc176876011)

[11.54 Split Key Method Enumeration 232](#_Toc176876012)

[11.55 Split Key Polynomial Enumeration 232](#_Toc176876013)

[11.56 State Enumeration 232](#_Toc176876014)

[11.57 Tag Enumeration 233](#_Toc176876015)

[11.58 Ticket Type Enumeration 245](#_Toc176876016)

[11.59 Unique Identifier Enumeration 245](#_Toc176876017)

[11.60 Unwrap Mode Enumeration 246](#_Toc176876018)

[11.61 Usage Limits Unit Enumeration 247](#_Toc176876019)

[11.62 Validity Indicator Enumeration 247](#_Toc176876020)

[11.63 Wrapping Method Enumeration 247](#_Toc176876021)

[11.64 Validation Authority Type Enumeration 248](#_Toc176876022)

[11.65 Validation Type Enumeration 248](#_Toc176876023)

[12 Bit Masks 249](#_Toc176876024)

[12.1 Cryptographic Usage Mask 249](#_Toc176876025)

[12.2 Protection Storage Mask 251](#_Toc176876026)

[12.3 Storage Status Mask 251](#_Toc176876027)

[12.4 Object Class Mask 251](#_Toc176876028)

[13 Algorithm Implementation 252](#_Toc176876029)

[13.1 Split Key Algorithms 252](#_Toc176876030)

[14 KMIP Client and Server Implementation Conformance 253](#_Toc176876031)

[14.1 KMIP Client Implementation Conformance 253](#_Toc176876032)

[14.2 KMIP Server Implementation Conformance 253](#_Toc176876033)

[Appendix A. Acknowledgments 254](#_Toc176876034)

[Appendix B. Acronyms 256](#_Toc176876035)

[Appendix C. List of Figures and Tables 259](#_Toc176876036)

[Appendix D. Revision History 274](#_Toc176876037)

[Appendix E. Notices 276](#_Toc176876038)

# Introduction

This document is intended as a specification of the protocol used for the communication (request and response messages) between clients and servers to perform certain management operations on objects stored and maintained by a key management system. These objects are referred to as Managed Objects in this specification. They include symmetric and asymmetric cryptographic keys and digital certificates. Managed Objects are managed with operations that include the ability to generate cryptographic keys, register objects with the key management system, obtain objects from the system, destroy objects from the system, and search for objects maintained by the system. Managed Objects also have associated attributes, which are named values stored by the key management system and are obtained from the system via operations. Certain attributes are added, modified, or deleted by operations.

This specification is complemented by several other documents. The KMIP Usage Guide [KMIP-UG] provides illustrative information on using the protocol. The KMIP Profiles Specification [KMIP-Prof] provides a normative set of base level conformance profiles and authentication suites that include the specific tests used to test conformance with the applicable KMIP normative documents. The KMIP Test Cases [KMIP-TC] provides samples of protocol messages corresponding to a set of defined test cases that are also used in conformance testing.

This specification defines the KMIP protocol version major 2 and minor 1 (see 6.1).

## IPR Policy

This specification is provided under the [RF on RAND Terms](https://www.oasis-open.org/policies-guidelines/ipr#RF-on-RAND-Mode) Mode of the [OASIS IPR Policy](https://www.oasis-open.org/policies-guidelines/ipr), the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC’s web page (<https://www.oasis-open.org/committees/kmip/ipr.php>).

## Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [[RFC2119](#RFC2119)].

For acronyms used in this document, see Appendix B. For definitions not found in this document, see [SP800-57-1].

| **Term** | **Definition** |
| --- | --- |
| Archive | To place information not accessed frequently into long-term storage. |
| Asymmetric key pair  (key pair) | A public key and its corresponding private key; a key pair is used with a public key algorithm. |
| Authentication | A process that establishes the origin of information, or determines an entity’s identity. |
| Authentication code | A cryptographic checksum based on a security function. |
| Authorization | Access privileges that are granted to an entity; conveying an “official” sanction to perform a security function or activity. |
| Certificate length | The length (in bytes) of an X.509 public key certificate. |
| Certification authority | The entity in a Public Key Infrastructure (PKI) that is responsible for issuing certificates, and exacting compliance to a PKI policy. |
| Ciphertext | Data in its encrypted form. |
| Compromise | The unauthorized disclosure, modification, substitution or use of sensitive data (e.g., keying material and other security-related information). |
| Confidentiality | The property that sensitive information is not disclosed to unauthorized entities. |
| Cryptographic algorithm | A well-defined computational procedure that takes variable inputs, including a cryptographic key and produces an output. |
| Cryptographic key (key) | A parameter used in conjunction with a cryptographic algorithm that determines its operation in such a way that an entity with knowledge of the key can reproduce or reverse the operation, while an entity without knowledge of the key cannot. Examples include:  1. The transformation of plaintext data into ciphertext data,  2. The transformation of ciphertext data into plaintext data,  3. The computation of a digital signature from data,  4. The verification of a digital signature,  5. The computation of an authentication code from data, and  6. The verification of an authentication code from data and a received authentication code. |
| Decryption | The process of changing ciphertext into plaintext using a cryptographic algorithm and key. |
| Digest (or hash) | The result of applying a hashing algorithm to information. |
| Digital signature (signature) | The result of a cryptographic transformation of data that, when properly implemented with supporting infrastructure and policy, provides the services of:  1. origin authentication  2. data integrity, and  3. signer non-repudiation. |
| Digital Signature Algorithm | A cryptographic algorithm used for digital signature. |
| Encryption | The process of changing plaintext into ciphertext using a cryptographic algorithm and key. |
| Hashing algorithm (or hash algorithm, hash function) | An algorithm that maps a bit string of arbitrary length to a fixed length bit string. Approved hashing algorithms satisfy the following properties:  1. (One-way) It is computationally infeasible to find any input that  maps to any pre-specified output, and  2. (Collision resistant) It is computationally infeasible to find any two distinct inputs that map to the same output. |
| Integrity | The property that sensitive data has not been modified or deleted in an unauthorized and undetected manner. |
| Key derivation (derivation) | A function in the lifecycle of keying material; the process by which one or more keys are derived from:  1) Either a shared secret from a key agreement computation or a pre-shared cryptographic key, and  2) Other information. |
| Key management | The activities involving the handling of cryptographic keys and other related security parameters (e.g., IVs and passwords) during the entire life cycle of the keys, including their generation, storage, establishment, entry and output, and destruction. |
| Key wrapping (wrapping) | A method of encrypting and/or MACing/signing keys. |
| Message Authentication Code (MAC) | A cryptographic checksum on data that uses a symmetric key to detect both accidental and intentional modifications of data. |
| PGP Key | A RFC 4880-compliant container of cryptographic keys and associated metadata. Usually text-based (in PGP-parlance, ASCII-armored). |
| Private key | A cryptographic key used with a public key cryptographic algorithm that is uniquely associated with an entity and is not made public. The private key is associated with a public key. Depending on the algorithm, the private key MAY be used to:  1. Compute the corresponding public key,  2. Compute a digital signature that can be verified by the corresponding public key,  3. Decrypt data that was encrypted by the corresponding public key, or  4. Compute a piece of common shared data, together with other information. |
| Profile | A specification of objects, attributes, operations, message elements and authentication methods to be used in specific contexts of key management server and client interactions (see [KMIP-Prof]). |
| Public key | A cryptographic key used with a public key cryptographic algorithm that is uniquely associated with an entity and that MAY be made public. The public key is associated with a private key. The public key MAY be known by anyone and, depending on the algorithm, MAY be used to:  1. Verify a digital signature that is signed by the corresponding private key,  2. Encrypt data that can be decrypted by the corresponding private key, or  3. Compute a piece of shared data. |
| Public key certificate (certificate) | A set of data that uniquely identifies an entity, contains the entity's public key and possibly other information, and is digitally signed by a trusted party, thereby binding the public key to the entity. |
| Public key cryptographic algorithm | A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that determining the private key from the public key is computationally infeasible. |
| Public Key Infrastructure | A framework that is established to issue, maintain and revoke public key certificates. |
| Recover | To retrieve information that was archived to long-term storage. |
| Split Key | A process by which a cryptographic key is split into *n* multiple key components, individually providing no knowledge of the original key, which can be subsequently combined to recreate the original cryptographic key. If knowledge of *k* (where *k* is less than or equal to *n*) components is necessary to construct the original key, then knowledge of any *k*-1 key components provides no information about the original key other than, possibly, its length. |
| Symmetric key | A single cryptographic key that is used with a secret (symmetric) key algorithm. |
| Symmetric key algorithm | A cryptographic algorithm that uses the same secret (symmetric) key for an operation and its inverse (e.g., encryption and decryption). |
| X.509 certificate | The ISO/ITU-T X.509 standard defined two types of certificates – the X.509 public key certificate, and the X.509 attribute certificate. Most commonly (including this document), an X.509 certificate refers to the X.509 public key certificate. |
| X.509 public key certificate | The public key for a user (or device) and a name for the user (or device), together with some other information, rendered un-forgeable by the digital signature of the certification authority that issued the certificate, encoded in the format defined in the ISO/ITU-T X.509 standard. |

Table 1: Terminology

## Normative References

[AWS-SIGV4] *Authenticating Requests (AWS Signature Version 4)* <https://docs.aws.amazon.com/AmazonS3/latest/API/sig-v4-authenticating-requests.htm>

[CHACHA] D. J. Bernstein. ChaCha, a variant of Salsa20. <https://cr.yp.to/chacha/chacha-20080128.pdf>

[ECC-Brainpool] *ECC Brainpool Standard Curves and Curve Generation v. 1.0.19.10.2005*, [https://web.archive.org/web/20180408010242/http://www.ecc-brainpool.org/download/Domain-parameters.pdf](https://web.archive.org/web/20180408010242/http:/www.ecc-brainpool.org/download/Domain-parameters.pdf).

[FIPS180-4] *Secure Hash Standard (SHS)*, FIPS PUB 180-4, August 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

[FIPS186-4] *Digital Signature Standard (DSS)*, FIPS PUB 186-4, July 2013, <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>.

[FIPS197] *Advanced Encryption Standard*, FIPS PUB 197, November 2001, <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.

[FIPS198-1] *The Keyed-Hash Message Authentication Code (HMAC)*, FIPS PUB 198-1, July 2008, <http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf>.

[FIPS202] SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, August 2015. <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf>

[FIPS203] Module-Lattice-Based Key-Encapsulation Mechanism Standard, August 2024. <https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.203.pdf>

[FIPS204] Module-Lattice-Based Digital Signature Standard, August 2024. <https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.204.pdf>

[FIPS205] Stateless Hash-Based Digital Signature Standard, August 2024. <https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.205.pdf>

[IEEE1003-1] IEEE Std 1003.1, Standard for information technology - portable operating system interface (POSIX). Shell and utilities, 2004.

[ISO16609] ISO, Banking -- Requirements for message authentication using symmetric techniques, ISO 16609, 2012.

[ISO9797-1] ISO/IEC, Information technology -- Security techniques -- Message Authentication Codes (MACs) -- Part 1: Mechanisms using a block cipher, ISO/IEC 9797-1, 2011.

[KMIP-Prof] Key Management Interoperability Protocol Profiles Version 3.0. Edited by Tim Hudson and Tim Chevalier. Latest version: Work in Progress

[PKCS#1] RSA Laboratories, *PKCS #1 v2.1: RSA Cryptography Standard*, June 14, 2002, https://tools.ietf.org/html/rfc8017.

[PKCS#5] RSA Laboratories, *PKCS #5 v2.1: Password-Based Cryptography Standard*, October 5, 2006, https://tools.ietf.org/html/rfc8018.

[PKCS#8] RSA Laboratories, *PKCS#8 v1.2: Private-Key Information Syntax Standard*, November 1, 1993, https://tools.ietf.org/html/rfc5958.

[PKCS#10] RSA Laboratories, *PKCS #10 v1.7: Certification Request Syntax Standard*, May 26, 2000, https://tools.ietf.org/html/rfc2986

[PKCS#11] OASIS PKCS#11 Cryptographic Token Interface Base Specification Version 3.0

[POLY1305] Daniel J. Bernstein. *The Poly1305-AES Message-Authentication Code*. In Henri Gilbert and Helena Handschuh, editors, Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers, volume 3557 of Lecture Notes in Computer Science, pages 32–49. Springer, 2005.

[RFC1319] B. Kaliski, *The MD2 Message-Digest Algorithm*, IETF RFC 1319, Apr 1992, <http://www.ietf.org/rfc/rfc1319.txt>.

[RFC1320] R. Rivest, *The MD4 Message-Digest Algorithm*, IETF RFC 1320, April 1992, <http://www.ietf.org/rfc/rfc1320.txt>.

[RFC1321] R. Rivest, *The MD5 Message-Digest Algorithm*, IETF RFC 1321, April 1992, <http://www.ietf.org/rfc/rfc1321.txt>.

[RFC1421] J. Linn, *Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures*, IETF RFC 1421, February 1993, <http://www.ietf.org/rfc/rfc1421.txt>.

[RFC1424] B. Kaliski, *Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services*, IETF RFC 1424, Feb 1993, <http://www.ietf.org/rfc/rfc1424.txt>.

[RFC2104] H. Krawczyk, M. Bellare, R. Canetti, *HMAC: Keyed-Hashing for Message Authentication*, IETF RFC 2104, February 1997, <http://www.ietf.org/rfc/rfc2104.txt>.

[RFC2119] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.

[RFC2898] B. Kaliski, *PKCS #5: Password-Based Cryptography Specification Version 2.0*, IETF RFC 2898, September 2000, <http://www.ietf.org/rfc/rfc2898.txt>.

**[RFC2986]** M. Nystrom and B. Kaliski, *PKCS #10: Certification Request Syntax Specification Version 1.7*, IETF RFC2986, November 2000, <http://www.rfc-editor.org/rfc/rfc2986.txt>.

[RFC3447] J. Jonsson, B. Kaliski, *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*, IETF RFC 3447, Feb 2003, <http://www.ietf.org/rfc/rfc3447.txt>.

[RFC3629] F. Yergeau, *UTF-8, a transformation format of ISO 10646*, IETF RFC 3629, November 2003, <http://www.ietf.org/rfc/rfc3629.txt>.

[RFC3686] R. Housley, *Using Advanced Encryption Standard (AES) Counter Mode with IPsec Encapsulating Security Payload (ESP), IETF RFC 3686,* January 2004, <http://www.ietf.org/rfc/rfc3686.txt>.

[RFC4210] C. Adams, S. Farrell, T. Kause and T. Mononen, *Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)*, IETF RFC 4210, September 2005, <http://www.ietf.org/rfc/rfc4210.txt>.

[RFC4211] J. Schaad, *Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)*, IETF RFC 4211, September 2005, <http://www.ietf.org/rfc/rfc4211.txt>.

[RFC4226] D. M’Raihi, M. Bellare, F. Hoornaert, D. Naccache, O. Ranen, HOTP: An HMACBased One-Time Password Algorithm, IETF RFC 4226, December 2005, <http://www.ietf.org/rfc/rfc4226.txt>.

[RFC4880] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, *OpenPGP Message Format*, IETF RFC 4880, November 2007, <http://www.ietf.org/rfc/rfc4880.txt>.

[RFC4949] R. Shirey, *Internet Security Glossary, Version 2*, IETF RFC 4949, August 2007, <http://www.ietf.org/rfc/rfc4949.txt>.

[RFC5272] J. Schaad and M. Meyers, *Certificate Management over CMS (CMC)*, IETF RFC 5272, June 2008, <http://www.ietf.org/rfc/rfc5272.txt>.

[RFC5280] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk, *Internet X.509 Public Key Infrastructure Certificate*, IETF RFC 5280, May 2008, <http://www.ietf.org/rfc/rfc5280.txt>.

[RFC5639] M. Lochter, J. Merkle, *Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation*, IETF RFC 5639, March 2010, <http://www.ietf.org/rfc/rfc5639.txt>.

**[RFC5869]** H. Krawczyk, HMAC-based Extract-and-Expand Key Derivation Function (HKDF), IETF RFC5869, May 2010, <https://tools.ietf.org/html/rfc5869>

**[RFC5958]** S. Turner, Asymmetric Key Packages, IETF RFC5958, August 2010, <https://tools.ietf.org/rfc/rfc5958.txt>

**[RFC6238]** D. M’Raihi, S. Machani, M. Pei, J. Rydell, TOTP: Time-Based One-Time Password Algorithm, IETF RFC 6238, May 2011, <http://www.ietf.org/rfc/rfc6238.txt>

**[RFC6402]** J. Schaad, *Certificate Management over CMS (CMC) Updates*, IETF RFC6402, November 2011, <http://www.rfc-editor.org/rfc/rfc6402.txt>.

**[RFC6818]** P. Yee, *Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, IETF RFC6818, January 2013, <http://www.rfc-editor.org/rfc/rfc6818.txt>.

[RFC7778] A. Langley, M. Hamburg, S. Turner *Elliptic Curves for Security*, IETF RFC7748, January 2016, <https://tools.ietf.org/html/rfc7748>

[RFC9562] K. Davis, B. Peabody, P. Leach *Universally Unique Identifiers (UUIDs)*, IETF RFC9562, May 2024, [https://tools.ietf.org/html/rfc](https://tools.ietf.org/html/rfc7748)9562

[SEC2] SEC 2: Recommended Elliptic Curve Domain Parameters, <http://www.secg.org/SEC2-Ver-1.0.pdf>.

[SP800-38A] M. Dworkin, *Recommendation for Block Cipher Modes of Operation – Methods and Techniques*, NIST Special Publication 800-38A, December 2001, <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf>

[SP800-38B] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*, NIST Special Publication 800-38B, May 2005, <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38b.pdf>

[SP800-38C] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality*, NIST Special Publication 800-38C, May 2004, updated July 2007 <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf>

[SP800-38D] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, NIST Special Publication 800-38D, Nov 2007, <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf>.

[SP800-38E] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Block-Oriented Storage Devices*, NIST Special Publication 800-38E, January 2010, <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf>.

[SP800-56A] E. Barker, L. Chen, A. Roginsky and M. Smid, *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*, NIST Special Publication 800-56A Revision 2, May 2013, <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf>.

[SP800-57-1] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, *Recommendations for Key Management - Part 1: General (Revision 3)*, NIST Special Publication 800-57 Part 1 Revision 3, July 2012, <http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf>.

[SP800-108] L. Chen, *Recommendation for Key Derivation Using Pseudorandom Functions (Revised)*, NIST Special Publication 800-108, Oct 2009, <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-108.pdf>.

[X.509] International Telecommunication Union (ITU)–T, X.509: Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks, November 2008, https://www.itu.int/rec/dologin\_pub.asp?lang=e&id=T-REC-X.509-201910-I!!PDF-E&type=items.

[X9.24-1] ANSI, X9.24 - Retail Financial Services Symmetric Key Management - Part 1: Using Symmetric Techniques, 2009.

[X9.31] ANSI, X9.31: Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA), September 1998.

[X9.42] ANSI, X9.42: Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography, 2003.

[X9.62] ANSI, X9.62: Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), 2005.

[X9.63] ANSI, X9.63: Public Key Cryptography for the Financial Services Industry, Key Agreement and Key Transport Using Elliptic Curve Cryptography, 2011.

[X9.102] ANSI, X9.102: Symmetric Key Cryptography for the Financial Services Industry - Wrapping of Keys and Associated Data, 2008.

[X9 TR-31] ANSI, X9 TR-31: Interoperable Secure Key Exchange Key Block Specification for Symmetric Algorithms, 2010.

## Non-Normative References

[ISO/IEC 9945-2] The Open Group, Regular Expressions, The Single UNIX Specification version 2, 1997, ISO/IEC 9945-2:1993, <http://www.opengroup.org/onlinepubs/007908799/xbd/re.html>.

[KMIP-UG] Key Management Interoperability Protocol Usage Guide Version 3.0. Work in progress.

[KMIP-TC] *Key Management Interoperability Protocol Test Cases Version 3.0*. Edited by Tim Hudson and Mark Joseph. Latest version: Work in Progress

**[RFC6151]** S. Turner and L. Chen, *Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms*, IETF RFC6151, March 2011, <http://www.rfc-editor.org/rfc/rfc6151.txt>.

[w1979] A. Shamir, How to share a secret, Communications of the ACM, vol. 22, no. 11, pp. 612-613, November 1979.

**[RFC7292]** K. Moriarty, M. Nystrom, S. Parkinson, A. Rusch, M. Scott. *PKCS #12: Personal Information Exchange Syntax v1.1, July 2014,* <https://tools.ietf.org/html/rfc7292>

## Item Data Types

The following are the data types of which all items (Objects, Attributes and Messages) are composed of Integer, Long Integer, Big Integer, Enumeration, Boolean, Text String, Byte String, Date Time, Interval, Date Time Extended, and Structure.

# Objects

A Key Management System operates on different classes of objects – objects that are subjects of key

management operations (User Objects) and objects that support the operation of the key management

system (System Objects).

All objects within the key management system SHALL have a minimum set of attributes.

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Unique Identifier | Yes |
| Short Unique Identifier | Yes |
| Object Class | Yes |
| Object Type | Yes |
| Initial Date | Yes |

Table 2: Minimum required Object attributes

## System Objects

System Objects are objects that support the operation of the key management system.

All System Objects SHALL have an Object Class of System.

Special authentication and authorization SHOULD be enforced when creating or destroying System Objects or when setting, modifying or deleting attributes of System Objects.

### User

A user of the key management system.

All User objects within the key management system a minimum set of attributes.

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Unique Identifier | Yes |
| Short Unique Identifier | Yes |
| Object Class | Yes |
| Object Type | Yes |
| Initial Date | Yes |
| Name | Yes |
| Credential Link | Yes |

Table 3: Required User Aattributes

### Group

A collection of objects within the key management system.

All Group objects within the key management system a minimum set of attributes.

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Unique Identifier | Yes |
| Short Unique Identifier | Yes |
| Object Class | Yes |
| Object Type | Yes |
| Initial Date | Yes |
| Name | Yes |

Table 4: Required Group Attributes

### Credentials

For each type of *Credential* in the key management system, there is a corresponding *System Object* that provides the necessary information for use of the credential by the system for identification or authentication purposes.

#### Password Credential

For the Credential Type of *Password*, the information required for validating a user-provided password may be supplied in the following methods:

* The Password field specifies the secret
* The Password Link attribute references a Secret Data managed object that specifies the secret.
* The Salted Password and Password Salt Algorithm combined with the optional Password Salt provides an algorithm-based comparison mechanism

One of the above methods SHALL be specified.

If the server stores a salted version of the password rather than the plaintext password then the Password Salt algorithm specifies the salting algorithm and the Password Salt is the optional parameter along with the Password to the salting algorithm and the Salted Password is the result of salting algorithm. If the password is stored in salted form the Password field SHALL NOT be specified.

If the Interation Count is unspecified, the value SHALL default to 100.

The client may either specify the password (in plaintext) or provide sufficient other fields to enable secure initialization.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Password Credential | Structure |  |
| Password | Text String | No |
| Password Salt | Byte String | No |
| Password Salt Algorithm | Cryptographic Algorithm Enumeration | No |
| Salted Password | Byte String | No |
| Iteration Count | Integer | No |

Table 5: Password Credential Structure

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Credential Type | Yes |
| Password Link | No |

Table 6: Password Credential Attributes

#### Device Credential

For the Credential Type of *Device*, one or a combination of the *Device Serial Number*, *Network Identifier*, *Machine Identifier*, and *Media Identifier* SHALL be unique. Server implementations MAY enforce policies on uniqueness for individual fields. A shared secret or password MAY also be used to authenticate the client. The client SHALL provide at least one field.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Device Credential | Structure |  |
| Device Serial Number | Text String | No |
| Device Identifier | Text String | No |
| Network Identifier | Text String | No |
| Machine Identifier | Text String | No |
| Media Identifier | Text String | No |

Table 7: Device Credential Structure

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Credential Type | Yes |

Table 8: Device Credential Attributes

#### One Time Password Credential

If the Credential Type in the Credential is One Time Password, then Credential Value is a structure. The Username field identifies the client, and the Password field is a secret that authenticates the client. The One Time Password field contains a one time password (OTP) which may only be used for a single authentication.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| OTP Credential | Structure |  |
| OTP Algorithm | Enumeration | Yes |
| OTP Digest | Cryptographic Algorithm Enumeration | No |
| OTP Serial | Text String | No |
| OTP Seed | Byte String | No |
| OTP Interval | Interval | No |
| OTP Digits | Integer | No |

Table 9: One Time Password Credential Structure

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Credential Type | Yes |
| OTP Counter | No |

Table 10: One Time Password Credential Attributes

#### Hashed Password Credential

If the Credential Type in the Credential is Hashed Password, then Credential Value is a structure. The Username field identifies the client. The timestamp is the current timestamp used to produce the hash and SHALL monotonically increase. The Hashing Algorithm SHALL default to SHA 256. The Hashed Password is defined as:

Hashed Password = Hash(S1 || Timestamp) || S2

Where:

* S1 = Hash(Username || Password)
* S2 = Hash(Password || Username)

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Hashed Password Credential | Structure |  |
| Hashing Algorithm | Enumeration | No |
| Hash Username Password | Byte String | Yes |
| Hash Password Username | Byte String | Yes |

Table 11: Hashed Password Credential Structure

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Credential Type | Yes |

Table 12: Hashed Password Credential Attributes

## User Objects

Managed Objects are objects that are the subjects of key management operations. *Managed Cryptographic Objects* are the subset of Managed Objects that contain cryptographic material.

All User Objects SHALL have an Object Class of User.

### Certificate

A Managed Cryptographic Object that is a digital certificate. It is a DER-encoded X.509 public key certificate.

All Certificate objects within the key management system SHALL have a minimum set of attributes.

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Unique Identifier | Yes |
| Short Unique Identifier | Yes |
| Object Class | Yes |
| Object Type | Yes |
| Initial Date | Yes |

Table 13: Minimum required Certificate Object attributes

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Certificate | Structure |  |
| Certificate Type | Enumeration | Yes |
| Certificate Value | Byte String | Yes |

Table 14: Certificate Object Structure

### Certificate Request

A Managed Cryptographic Object containing the Certificate Request.

All Certificate Request objects within the key management system SHALL have a minimum set of attributes.

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Unique Identifier | Yes |
| Short Unique Identifier | Yes |
| Object Class | Yes |
| Object Type | Yes |
| Initial Date | Yes |

Table 15: Minimum required Certificate Request Object attributes

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Certificate Request | Structure |  |
| Certificate Request Type | Enumeration | Yes |
| Certificate Request Value | Byte String | Yes |

Table 16: Certificate Request Structure

### Opaque Object

A Managed Object that the key management server is possibly not able to interpret. The context information for this object MAY be stored and retrieved using Custom Attributes.

An Opaque Object MAY be a Managed Cryptographic Object depending on the client context of usage and as such is treated in the same manner as a Managed Cryptographic Object for handling of attributes.

All Opaque Object objects within the key management system SHALL have a minimum set of attributes.

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Unique Identifier | Yes |
| Short Unique Identifier | Yes |
| Object Class | Yes |
| Object Type | Yes |
| Initial Date | Yes |

Table 17: Minimum required Opaque Object Object attributes

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Opaque Object | Structure |  |
| Opaque Data Type | Enumeration | Yes |
| Opaque Data Value | Byte String | Yes |

Table 18: Opaque Object Structure

### PGP Key

A Managed Cryptographic Object that is a text-based representation of a PGP key. The Key Block field, indicated below, will contain the ASCII-armored export of a PGP key in the format as specified in RFC 4880. It MAY contain only a public key block, or both a public and private key block. Two different versions of PGP keys, version 3 and version 4, MAY be stored in this Managed Cryptographic Object.

KMIP implementers SHOULD treat the Key Block field as an opaque blob. PGP-aware KMIP clients SHOULD take on the responsibility of decomposing the Key Block into other Managed Cryptographic Objects (Public Keys, Private Keys, etc.).

All PGP Key objects within the key management system SHALL have a minimum set of attributes.

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Unique Identifier | Yes |
| Short Unique Identifier | Yes |
| Object Class | Yes |
| Object Type | Yes |
| Initial Date | Yes |

Table 19: Minimum required PGP Key Object attributes

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| PGP Key | Structure |  |
| PGP Key Version | Integer | Yes |
| Key Block | Object Data Structure | Yes |

Table 20: PGP Key Object Structure

### Private Key

A Managed Cryptographic Object that is the private portion of an asymmetric key pair.

All Private Key objects within the key management system SHALL have a minimum set of attributes.

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Unique Identifier | Yes |
| Short Unique Identifier | Yes |
| Object Class | Yes |
| Object Type | Yes |
| Initial Date | Yes |

Table 21: Minimum required Private Key Object attributes

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Private Key | Structure |  |
| Key Block | Object Data Structure | Yes |

Table 22: Private Key Object Structure

### Public Key

A Managed Cryptographic Object that is the public portion of an asymmetric key pair. This is only a public key, not a certificate.

All Public Key objects within the key management system SHALL have a minimum set of attributes.

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Unique Identifier | Yes |
| Short Unique Identifier | Yes |
| Object Class | Yes |
| Object Type | Yes |
| Initial Date | Yes |

Table 23: Minimum required Public Key Object attributes

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Public Key | Structure |  |
| Key Block | Object Data Structure | Yes |

Table 24: Public Key Object Structure

### Secret Data

A Managed Cryptographic Object containing a shared secret value that is not a key or certificate (e.g., a password). The Key Block of the *Secret Data* object contains a Key Value of the Secret Data Type. The Key Value MAY be wrapped.

All Secret Data objects within the key management system SHALL have a minimum set of attributes.

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Unique Identifier | Yes |
| Short Unique Identifier | Yes |
| Object Class | Yes |
| Object Type | Yes |
| Initial Date | Yes |

Table 25: Minimum required Secret Data Object attributes

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Secret Data | Structure |  |
| Secret Data Type | Enumeration | Yes |
| Key Block | Object Data Structure | Yes |

Table 26: Secret Data Object Structure

### Split Key

A Managed Cryptographic Object that is a *Split Key*. A split key is a secret, usually a symmetric key or a private key that has been split into a number of parts, each of which MAY then be distributed to several key holders, for additional security. The *Split Key Parts* field indicates the total number of parts, and the *Split Key Threshold* field indicates the minimum number of parts needed to reconstruct the entire key. The *Key Part Identifier* indicates which key part is contained in the cryptographic object, and SHALL be at least 1 and SHALL be less than or equal to Split Key Parts.

All Split Key objects within the key management system SHALL have a minimum set of attributes.

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Unique Identifier | Yes |
| Short Unique Identifier | Yes |
| Object Class | Yes |
| Object Type | Yes |
| Initial Date | Yes |

Table 27: Minimum required Split Key Object attributes

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Split Key | Structure |  |
| Split Key Parts | Integer | Yes |
| Key Part Identifier | Integer | Yes |
| Split Key Threshold | Integer | Yes |
| Split Key Method | Enumeration | Yes |
| Prime Field Size | Big Integer | No, REQUIRED only if Split Key Method is Polynomial Sharing Prime Field. |
| Key Block | Object Data Structure | Yes |

Table 28: Split Key Object Structure

### Symmetric Key

A Managed Cryptographic Object that is a symmetric key.

All Symmetric Key objects within the key management system SHALL have a minimum set of attributes.

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Unique Identifier | Yes |
| Short Unique Identifier | Yes |
| Object Class | Yes |
| Object Type | Yes |
| Initial Date | Yes |

Table 29: Minimum required Symmetric Key Object attributes

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Symmetric Key | Structure |  |
| Key Block | Structure | Yes |

Table 30: Symmetric Key Object Structure

# Object Data Structures

## Key Block

A *Key Block* object is a structure used to encapsulate all of the information that is closely associated with a cryptographic key.

The Key Block MAY contain the Key Compression Type, which indicates the format of the elliptic curve public key. By default, the public key is uncompressed.

The Key Block also has the Cryptographic Algorithm and the Cryptographic Length of the key contained in the Key Value field. Some example values are:

|  |  |
| --- | --- |
| **Value** | **Description** |
| RSA keys | Typically 1024, 2048 or 3072 bits in length. |
| 3DES keys | Typically from 112 to 192 bits (depending upon key length and the presence of parity bits). |
| AES keys | 128, 192 or 256 bits in length |

Table 31: Key Block Cryptographic Algorithm & Length Description

The Key Block SHALL contain a Key Wrapping Data structure if the key in the Key Value field is wrapped (i.e., encrypted, or MACed/signed, or both).

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Key Block | Structure |  |
| Key Format Type | Enumeration | Yes |
| Key Compression Type | Enumeration | No |
| Key Value | Byte String: for wrapped Key Value; Structure: for plaintext Key Value | No |
| Cryptographic Algorithm | Enumeration | Yes. MAY be omitted only if this information is available from the Key Value. Does not apply to Secret Data or Opaque If present, the Cryptographic Length SHALL also be present. |
| Cryptographic Length | Integer | Yes. MAY be omitted only if this information is available from the Key Value. Does not apply to Secret Data (or Opaque. If present, the Cryptographic Algorithm SHALL also be present. |
| Key Wrapping Data | Object Data Structure | No. SHALL only be present if the key is wrapped. |

Table 32: Key Block Object Structure

## Key Value

The *Key Value* is used only inside a Key Block and is either a Byte String or a:

* The Key Value structure contains the key material, either as a byte string or as a Transparent Key structure, and OPTIONAL attribute information that is associated and encapsulated with the key material. This attribute information differs from the attributes associated with Managed Objects, and is obtained via the Get Attributes operation, only by the fact that it is encapsulated with (and possibly wrapped with) the key material itself.
* The Key Value Byte String is either the wrapped TTLV-encoded Key Value structure, or the wrapped un-encoded value of the Byte String Key Material field.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Key Value | Structure |  |
| Key Material | Byte String: for Raw, Opaque, PKCS1, PKCS8, ECPrivateKey, or Extension Key Format types;  Structure: for Transparent, or Extension Key Format Types | Yes |
| Attributes | Structure | No |

Table 33: Key Value Object Structure

## Key Wrapping Data

The Key Block MAY also supply OPTIONAL information about a cryptographic key wrapping mechanism used to wrap the Key Value. This consists of a *Key Wrapping Data* structure. It is only used inside a Key Block.

This structure contains fields for:

|  |  |
| --- | --- |
| **Value** | **Description** |
| Wrapping Method | Indicates the method used to wrap the Key Value. |
| Encryption Key Information | Contains the Unique Identifier value of the encryption key and associated cryptographic parameters. |
| MAC/Signature Key Information | Contains the Unique Identifier value of the MAC/signature key and associated cryptographic parameters. |
| MAC/Signature | Contains a MAC or signature of the Key Value |
| IV/Counter/Nonce | If REQUIRED by the wrapping method. |
| Encoding Option | Specifies the encoding of the Key Material within the Key Value structure of the Key Block that has been wrapped. If No Encoding is specified, then the Key Value structure SHALL NOT contain any attributes. |

Table 34: Key Wrapping Data Structure Description

If wrapping is used, then the whole Key Value structure is wrapped unless otherwise specified by the Wrapping Method. The algorithms used for wrapping are given by the Cryptographic Algorithm attributes of the encryption key and/or MAC/signature key; the block-cipher mode, padding method, and hashing algorithm used for wrapping are given by the Cryptographic Parameters in the Encryption Key Information and/or MAC/Signature Key Information, or, if not present, from the Cryptographic Parameters attribute of the respective key(s). Either the Encryption Key Information or the MAC/Signature Key Information (or both) in the Key Wrapping Data structure SHALL be specified.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Key Wrapping Data | Structure |  |
| Wrapping Method | Enumeration | Yes |
| Encryption Key Information | Structure, see below | No. Corresponds to the key that was used to encrypt the Key Value. |
| MAC/Signature Key Information | Structure, see below | No. Corresponds to the symmetric key used to MAC the Key Value or the private key used to sign the Key Value |
| MAC/Signature | Byte String | No |
| IV/Counter/Nonce | Byte String | No |
| Encoding Option | Enumeration | No. Specifies the encoding of the Key Value Byte String. If not present, the wrapped Key Value structure SHALL be TTLV encoded. |

Table 35: Key Wrapping Data Object Structure

The structures of theEncryption Key Information and the MAC/Signature Key Information are as follows:

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Encryption Key Information | Structure |  |
| Unique Identifier | Reference or Name Reference or Unique Identifier Enumeration or Integer | Yes |
| Cryptographic Parameters | Structure | No |

Table 36: Encryption Key Information Object Structure

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| MAC/Signature Key Information | Structure |  |
| Unique Identifier | Reference or Name Reference or Unique Identifier Enumeration or Integer | Yes. It SHALL be either the Unique Identifier of the Symmetric Key used to MAC, or of the Private Key (or its corresponding Public Key) used to sign. |
| Cryptographic Parameters | Structure | No |

Table 37: MAC/Signature Key Information Object Structure

## Transparent Symmetric Key

If the Key Format Type in the Key Block is *Transparent Symmetric Key*, then Key Material is a structure.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Key Material | Structure |  |
| Key | Byte String | Yes |

Table 38: Key Material Object Structure for Transparent Symmetric Keys

## Transparent DSA Private Key

If the Key Format Type in the Key Block is *Transparent DSA Private Key*, then Key Material is a structure.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Key Material | Structure |  |
| P | Big Integer | Yes |
| Q | Big Integer | Yes |
| G | Big Integer | Yes |
| X | Big Integer | Yes |

Table 39: Key Material Object Structure for Transparent DSA Private Keys

## Transparent DSA Public Key

If the Key Format Type in the Key Block is *Transparent DSA Public Key*, then Key Material is a structure.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Key Material | Structure |  |
| P | Big Integer | Yes |
| Q | Big Integer | Yes |
| G | Big Integer | Yes |
| Y | Big Integer | Yes |

Table 40: Key Material Object Structure for Transparent DSA Public Keys

## Transparent RSA Private Key

If the Key Format Type in the Key Block is *Transparent RSA Private Key*, then Key Material is a structure.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Key Material | Structure |  |
| Modulus | Big Integer | Yes |
| Private Exponent | Big Integer | No |
| Public Exponent | Big Integer | No |
| P | Big Integer | No |
| Q | Big Integer | No |
| Prime Exponent P | Big Integer | No |
| Prime Exponent Q | Big Integer | No |
| CRT Coefficient | Big Integer | No |

Table 41: Key Material Object Structure for Transparent RSA Private Keys

One of the following SHALL be present (refer to [PKCS#1]):

* Private Exponent,
* P and Q (the first two prime factors of Modulus), or
* Prime Exponent P and Prime Exponent Q.

## Transparent RSA Public Key

If the Key Format Type in the Key Block is *Transparent RSA Public Key*, then Key Material is a structure.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Key Material | Structure |  |
| Modulus | Big Integer | Yes |
| Public Exponent | Big Integer | Yes |

Table 42: Key Material Object Structure for Transparent RSA Public Keys

## Transparent DH Private Key

If the Key Format Type in the Key Block is *Transparent DH Private Key*, then Key Material is a structure.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Key Material | Structure |  |
| P | Big Integer | Yes |
| Q | Big Integer | No |
| G | Big Integer | Yes |
| J | Big Integer | No |
| X | Big Integer | Yes |

Table 43: Key Material Object Structure for Transparent DH Private Keys

## Transparent DH Public Key

If the Key Format Type in the Key Block is *Transparent DH Public Key*, then Key Material is a.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Key Material | Structure |  |
| P | Big Integer | Yes |
| Q | Big Integer | No |
| G | Big Integer | Yes |
| J | Big Integer | No |
| Y | Big Integer | Yes |

Table 44: Key Material Object Structure for Transparent DH Public Keys

## Transparent EC Private Key

If the Key Format Type in the Key Block is *Transparent EC Private Key*, then Key Material is a structure.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Key Material | Structure |  |
| Recommended Curve | Enumeration | Yes |
| D | Big Integer | Yes |

Table 45: Key Material Object Structure for Transparent EC Private Keys

## Transparent EC Public Key

If the Key Format Type in the Key Block is *Transparent EC Public Key*, then Key Material is a structure.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Key Material | Structure |  |
| Recommended Curve | Enumeration | Yes |
| Q String | Byte String | Yes |

Table 46: Key Material Object Structure for Transparent EC Public Keys

# Object Attributes

The following subsections describe the attributes that are associated with Managed Objects. Attributes that an object MAY have multiple instances of are referred to as *multi-instance attributes*. All instances of an attribute SHOULD have a different value. Similarly, attributes which an object SHALL only have at most one instance of are referred to as *single-instance attributes*. Attributes are able to be obtained by a client from the server using the Get Attribute operation. Some attributes are able to be set by the Add Attribute operation or updated by the Modify Attribute operation, and some are able to be deleted by the Delete Attribute operation if they no longer apply to the Managed Object. *Read-only attributes* are attributes that SHALL NOT be modified by either server or client, and that SHALL NOT be deleted by a client.

When attributes are returned by the server (e.g., via a Get Attributes operation), the attribute value returned SHALL NOT differ for different clients unless specifically noted against each attribute.

The first table in each subsection contains the attribute name in the first row. This name is the canonical name used when managing attributes using the Get Attributes, Get Attribute List, Add Attribute, Modify Attribute, and Delete Attribute operations.

A server SHALL NOT delete attributes without receiving a request from a client until the object is destroyed. After an object is destroyed, the server MAY retain all, some or none of the object attributes, depending on the object type and server policy.

The second table in each subsection lists certain attribute characteristics (e.g., “SHALL always have a value. The server policy MAY further restrict these attribute characteristics.

|  |  |
| --- | --- |
| SHALL always have a value | All Managed Objects that are of the Object Types for which this attribute applies, SHALL always have this attribute set once the object has been created or registered, up until the object has been destroyed. |
| Initially set by | Who is permitted to initially set the value of the attribute (if the attribute has never been set, or if all the attribute values have been deleted)? |
| Modifiable by server | Is the server allowed to change an existing value of the attribute without receiving a request from a client? |
| Modifiable by client | Is the client able to change an existing value of the attribute value once it has been set? |
| Deletable by client | Is the client able to delete an instance of the attribute? |
| Multiple instances permitted | Are multiple instances of the attribute permitted? |
| When implicitly set | Which operations MAY cause this attribute to be set even if the attribute is not specified in the operation request itself? |
| Applies to Object Types | Which Managed Objects MAY have this attribute set? |

Table 47: Attribute Rules

There are default values for some mandatory attributes of Cryptographic Objects. The values in use by a particular server are available via Query. KMIP servers SHALL supply values for these attributes if the client omits them.

|  |  |
| --- | --- |
| Object | Attribute |
| Symmetric Key | Cryptographic Algorithm  Cryptographic Length  Cryptographic Usage Mask |
| Private Key | Cryptographic Algorithm  Cryptographic Length  Cryptographic Usage Mask |
| Public Key | Cryptographic Algorithm  Cryptographic Length  Cryptographic Usage Mask |
| Certificate | Cryptographic Algorithm  Cryptographic Length  Digital Signature Algorithm |
| Split Key | Cryptographic Algorithm  Cryptographic Length  Cryptographic Usage Mask |
| Secret Data | Cryptographic Usage Mask |

Table 48: Default Cryptographic Parameters

All characters within an Attribute Name are significant. A server SHALL NOT trim leading or trailing whitespace from Attribute Names if the client uses attributes of this format.

## Activation Date

The *Activation Date* attribute contains the date and time when the Managed Object MAY begin to be used. This time corresponds to state transition. The object SHALL NOT be used for any cryptographic purpose before the *Activation Date* has been reached. Once the state transition from Pre-Active has occurred, then this attribute SHALL NOT be changed or deleted before the object is destroyed.

|  |  |
| --- | --- |
| Item | Encoding |
| Activation Date | Date-Time |

Table 49: Activation Date Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server or Client |
| Modifiable by server | Yes, only while in Pre-Active state |
| Modifiable by client | Yes, only while in Pre-Active state |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Activate Certify, Re-certify, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 50: Activation Date Attribute Rules

## Alternative Name

The *Alternative* *Name* attribute is used to identify and locate the object. This attribute is assigned by the client, and the *Alternative* *Name Value* is intended to be in a form that humans are able to interpret. The key management system MAY specify rules by which the client creates valid alternative names. Clients are informed of such rules by a mechanism that is not specified by this standard. Alternative Names MAY NOT be unique within a given key management server.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Alternative Name | Structure |  |
| Alternative Name Value | Text String | Yes |
| Alternative Name Type | Enumeration | Yes |

Table 51: Alternative Name Attribute Structure

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client |
| Modifiable by server | Yes (Only if no value present) |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| Applies to Object Types | All Objects |

Table 52: Alternative Name Attribute Rules

## Always Sensitive

The server SHALL create this attribute, and set it to True if the Sensitive attribute has always been True. The server SHALL set it to False if the Sensitive attribute has ever been set to False.

|  |  |
| --- | --- |
| Item | Encoding |
| Sensitive | Boolean |

Table 53: Always Sensitive Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | When Sensitive attribute is set or changed |
| Applies to Object Types | All Objects |

Table 54: Always Sensitive Attribute Rules

## Application Specific Information

The *Application Specific Information* attribute is a structure used to store data specific to the application(s) using the Managed Object. It consists of the following fields: an *Application Namespace* and *Application Data* specific to that application namespace.

Clients MAY request to set (i.e., using any of the operations that result in new Managed Object(s) on the server or adding/modifying the attribute of an existing Managed Object an instance of this attribute with a particular *Application Namespace* while omitting *Application Data*. In that case, if the server supports this namespace (as indicated by the Query operation), then it SHALL return a suitable *Application Data* value. If the server does not support this namespace, then an error SHALL be returned.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Application Specific Information | Structure |  |
| Application Namespace | Text String | Yes |
| Application Data | Text String | No |

Table 55: Application Specific Information Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server (only if the Application Data is omitted, in the client request) |
| Modifiable by server | Yes (only if the Application Data is omitted in the client request) |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Re-key, Re-key Key Pair, Re-certify |
| Applies to Object Types | All Objects |

Table 56: Application Specific Information Attribute Rules

## Archive Date

The *Archive Date* attribute is the date and time when the Managed Object was placed in archival storage. This value is set by the server as a part of the Archive operation. The server SHALL delete this attribute whenever a Recover operation is performed.

|  |  |
| --- | --- |
| Item | Encoding |
| Archive Date | Date-Time |

Table 57: Archive Date Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Archive |
| Applies to Object Types | All Objects |

Table 58: Archive Date Attribute Rules

## Certificate Attributes

The Certificate Attributes are the various items included in a certificate or certificate request. The following list is based on RFC2253.

|  |  |
| --- | --- |
| Item | Encoding |
| Certificate Subject CN | Text String |
| Certificate Subject O | Text String |
| Certificate Subject OU | Text String |
| Certificate Subject Email | Text String |
| Certificate Subject C | Text String |
| Certificate Subject ST | Text String |
| Certificate Subject L | Text String |
| Certificate Subject UID | Text String |
| Certificate Subject Serial Number | Text String |
| Certificate Subject Title | Text String |
| Certificate Subject DC | Text String |
| Certificate Subject DN Qualifier | Text String |
| Certificate Subject DN | Text String |

Table 59: Certificate Attributes (Subject)

The Certificate Attributes are the various items included in a certificate. The following list is based on RFC2253.

|  |  |
| --- | --- |
| Item | Encoding |
| Certificate Issuer CN | Text String |
| Certificate Issuer O | Text String |
| Certificate Issuer OU | Text String |
| Certificate Issuer Email | Text String |
| Certificate Issuer C | Text String |
| Certificate Issuer ST | Text String |
| Certificate Issuer L | Text String |
| Certificate Issuer UID | Text String |
| Certificate Issuer Serial Number | Text String |
| Certificate Issuer Title | Text String |
| Certificate Issuer DC | Text String |
| Certificate Issuer DN Qualifier | Text String |
| Certificate Issuer DN | Text String |

Table 60: Certificate Attributes (Issuer)

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | Yes |
| When implicitly set | Register, Certify, Re-certify |
| Applies to Object Types | Certificates |

Table 61: Certificate Attribute Rules

## Certificate Type

The *Certificate Type* attribute is a type of certificate (e.g., X.509).

The *Certificate Type* value SHALL be set by the server when the certificate is created or registered and then SHALL NOT be changed or deleted before the object is destroyed.

|  |  |  |
| --- | --- | --- |
| Item | Encoding |  |
| Certificate Type | Enumeration |  |

Table 62: Certificate Type Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Register, Certify, Re-certify |
| Applies to Object Types | Certificates |

Table 63: Certificate Type Attribute Rules

## Certificate Length

The *Certificate Length* attribute is the length in bytes of the Certificate object. The *Certificate Length* SHALL be set by the server when the object is created or registered, and then SHALL NOT be changed or deleted before the object is destroyed.

|  |  |
| --- | --- |
| Item | Encoding |
| Certificate Length | Integer |

Table 64: Certificate Length Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Register, Certify, Re-certify |
| Applies to Object Types | Certificates |

Table 65: Certificate Length Attribute Rules

## Comment

The Comment attribute is used for descriptive purposes only. It is not used for policy enforcement. The attribute is set by the client or the server.

|  |  |
| --- | --- |
| Item | Encoding |
| Description | Text String |

Table 66: Comment Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| Applies to Object Types | All Objects |

Table 67: Comment Rules

## Compromise Date

The *Compromise Date* attribute contains the date and time when the Managed Cryptographic Object entered into the compromised state. This time corresponds to state transitions 3, 5, 8, or 10. This time indicates when the key management system was made aware of the compromise, not necessarily when the compromise occurred. This attribute is set by the server when it receives a Revoke operation with a *Revocation Reason* containing a *Revocation Reason Code* of Compromised code, or due to server policy or out-of-band administrative action.

|  |  |
| --- | --- |
| Item | Encoding |
| Compromise Date | Date-Time |

Table 68: Compromise Date Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Revoke |
| Applies to Object Types | All Objects |

Table 69: Compromise Date Attribute Rules

## Compromise Occurrence Date

The *Compromise Occurrence Date* attribute is the date and time when the Managed Object was first believed to be compromised. If it is not possible to estimate when the compromise occurred, then this value SHOULD be set to the Initial Date for the object.

|  |  |
| --- | --- |
| Item | Encoding |
| Compromise Occurrence Date | Date-Time |

Table 70: Compromise Occurrence Date Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Revoke |
| Applies to Object Types | All Objects |

Table 71: Compromise Occurrence Date Attribute Rules

## Contact Information

The *Contact Information* attribute is used for descriptive purposes only. It is not used for policy enforcement. The attribute is set by the client or the server.

|  |  |
| --- | --- |
| Item | Encoding |
| Contact Information | Text String |

Table 72: Contact Information Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 73: Contact Information Attribute Rules

## Counters

There are object attributes with name suffixes that identify themselves as “Counter” attributes. These attributes serve to record a successful instance of usage of an object as the subject of a KMIP operation. The prefix of the attribute name states the nature of the counter (which operation). Modification of this attribute is performed by the server via incrementing the counter value on each instance of “use”.

“Counter” attributes SHALL be present for Certificates, Certificate Requests, Private keys, Public keys and Symmetric keys.

### Certify Counter

The *Certify Counter* attribute is recorded against a given object and records each instance of a successful certify operation being performed on that object.

The *Certify Counter* SHALL be incremented upon completion of a successful Certify or Recertify operation.

|  |  |
| --- | --- |
| Item | Encoding |
| Certify Counter | Long Integer |

Table 74: Certify Counter Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | Yes – incremented on Certify & ReCertify |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Import, Register |
| Applies to Object Types | Certificate Request, Public Key |

Table 75: Certify Counter Attribute Rules

### Decrypt Counter

The *Decrypt Counter* attribute is recorded against a given object and records each incidence of a successful decrypt operation being performed using that object.

The Decrypt Counter SHALL be incremented upon completion of a successful Decrypt operation.

|  |  |
| --- | --- |
| Item | Encoding |
| Decrypt Counter | Long Integer |

Table 76: Decrypt Counter Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | Yes – incremented on Decrypt |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Register, Import |
| Applies to Object Types | Cryptographic Objects |

Table 77: Decrypt Counter Attribute Rules

### Encrypt Counter

The *Encrypt Counter* attribute is recorded against a given object and records each incidence of a successful encrypt operation being performed using that object.

The Encrypt Counter SHALL be incremented upon completion of a successful Encrypt operation.

|  |  |
| --- | --- |
| Item | Encoding |
| Encrypt Counter | Long Integer |

Table 78: Encrypt Counter Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | Yes – incremented on Encrypt |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Register, Import |
| Applies to Object Types | Cryptographic Objects |

Table 79: Encrypt Counter Attribute Rules

### Sign Counter

The *Sign Counter* attribute is recorded against a object and records each incidence of a successful sign operation being performed using that object.

The Sign Counter SHALL be incremented upon completion of a successful Sign operation.

|  |  |
| --- | --- |
| Item | Encoding |
| Sign Counter | Long Integer |

Table 80: Sign Counter Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | Yes – incremented on Sign |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Register, Import |
| Applies to Object Types | Cryptographic Objects |

Table 81: Sign Counter Attribute Rules

### Signature Verify Counter

The *Signature* *Verify Counter* attribute is recorded against an object and records each incidence of a successful Signature Verify operation being performed on that object.

The Signature Verify Counter SHALL be incremented upon completion of a successful Signature Verify operation.

|  |  |
| --- | --- |
| Item | Encoding |
| Signature Verify Counter | Long Integer |

Table 82: Signature Verify Counter Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | Yes – incremented on Signature Verify |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Register, Import |
| Applies to Object Types | Cryptographic Objects |

Table 83: Signature Verify Counter Attribute Rules

## Credential Type

The *Credential Type* of a System Object SHALL be set by the server when the object is created and then SHALL NOT be changed or deleted before the object is destroyed.

|  |  |
| --- | --- |
| Item | Encoding |
| Credential Type | Enumeration |

Table 84: Credential Type Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Register |
| Applies to Object Types | Credential Objects |

Table 85: Credential Type Attribute Rules

## Cryptographic Algorithm

The *Cryptographic Algorithm* of an object. The Cryptographic Algorithm of a Certificate object identifies the algorithm for the public key contained within the Certificate. The digital signature algorithm used to sign the Certificate is identified in the Digital Signature Algorithm attribute. This attribute SHALL be set by the server when the object is created or registered and then SHALL NOT be changed or deleted before the object is destroyed.

|  |  |
| --- | --- |
| Item | Encoding |
| Cryptographic Algorithm | Enumeration |

Table 86: Cryptographic Algorithm Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes (except for Secret Data and Opaque Object) |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Certify, Create, Create Key Pair, Re-certify, Register, Derive Key, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 87: Cryptographic Algorithm Attribute Rules

## Cryptographic Domain Parameters

The *Cryptographic Domain Parameters* attribute is a structure that contains fields that MAY need to be specified in the Create Key Pair Request Payload. Specific fields MAY only pertain to certain types of Managed Cryptographic Objects.

The domain parameter Qlength correponds to the bit length of parameter Q (refer to **[RFC7778]**, **[SEC2]** and **[SP800-56A]**).

Qlength applies to algorithms such as DSA and DH. The bit length of parameter P (refer to to **[RFC7778]**, **[SEC2]** and **[SP800-56A]**) is specified separately by setting the Cryptographic Length attribute.

Recommended Curve is applicable to elliptic curve algorithms such as ECDSA, ECDH, and ECMQV.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | Required |
| Cryptographic Domain Parameters | Structure | Yes |
| Qlength | Integer | No |
| Recommended Curve | Enumeration | No |

Table 88: Cryptographic Domain Parameters Attribute Structure

|  |  |
| --- | --- |
| Shall always have a value | No |
| Initially set by | Client |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Re-key, Re-key Key Pair |
| Applies to Object Types | Public Keys, Private Keys |

Table 89: Cryptographic Domain Parameters Attribute Rules

## Cryptographic Length

For keys, *Cryptographic Length* is the length in bits of the clear-text cryptographic key material of the Managed Cryptographic Object. For certificates, *Cryptographic Length* is the length in bits of the public key contained within the Certificate. This attribute SHALL be set by the server when the object is created or registered, and then SHALL NOT be changed or deleted before the object is destroyed.

|  |  |
| --- | --- |
| Item | Encoding |
| Cryptographic Length | Integer |

Table 90: Cryptographic Length Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes (Except for Opaque Object) |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Certify, Create, Create Key Pair, Re-certify, Register, Derive Key, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 91: Cryptographic Length Attribute Rules

## Cryptographic Parameters

The *Cryptographic Parameters* attribute is a structure that contains a set of OPTIONAL fields that describe certain cryptographic parameters to be used when performing cryptographic operations using the object. Specific fields MAY pertain only to certain types of Managed Objects. The Cryptographic Parameters attribute of a Certificate object identifies the cryptographic parameters of the public key contained within the Certificate.

The Cryptographic Algorithm is also used to specify the parameters for cryptographic operations. For operations involving digital signatures, either the Digital Signature Algorithm can be specified or the Cryptographic Algorithm and Hashing Algorithm combination can be specified.

Random IV can be used to request that the KMIP server generate an appropriate IV for a cryptographic operation that uses an IV. The generated Random IV is returned in the response to the cryptographic operation.

IV Length is the length of the Initialization Vector in bits. This parameter SHALL be provided when the specified Block Cipher Mode supports variable IV lengths such as CTR or GCM.

Tag Length is the length of the authenticator tag in bytes. This parameter SHALL be provided when the Block Cipher Mode is GCM.

The IV used with counter modes of operation (e.g., CTR and GCM) cannot repeat for a given cryptographic key. To prevent an IV/key reuse, the IV is often constructed of three parts: a fixed field, an invocation field, and a counter as described in [SP800-38A] and [SP800-38D]. The Fixed Field Length is the length of the fixed field portion of the IV in bits. The Invocation Field Length is the length of the invocation field portion of the IV in bits. The Counter Length is the length of the counter portion of the IV in bits.

Initial Counter Value is the starting counter value for CTR mode (for [RFC3686] it is 1).

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Cryptographic Parameters | Structure |  |
| Block Cipher Mode | Enumeration | No |
| Padding Method | Enumeration | No |
| Hashing Algorithm | Enumeration | No |
| Key Role Type | Enumeration | No |
| Digital Signature Algorithm | Enumeration | No |
| Cryptographic Algorithm | Enumeration | No |
| Random IV | Boolean | No |
| IV Length | Integer | No unless Block Cipher Mode supports variable IV lengths |
| Tag Length | Integer | No unless Block Cipher Mode is GCM |
| Fixed Field Length | Integer | No |
| Invocation Field Length | Integer | No |
| Counter Length | Integer | No |
| Initial Counter Value | Integer | No |
| Salt Length | Integer | No (if omitted, defaults to the block size of the Mask Generator Hashing Algorithm) |
| Mask Generator | Enumeration | No (if omitted defaults to MGF1). |
| Mask Generator Hashing Algorithm | Enumeration | No. (if omitted defaults to SHA-1). |
| P Source | Byte String | No (if omitted, defaults to an empty byte string for encoding input P in OAEP padding) |
| Trailer Field | Integer | No (if omitted, defaults to the standard one-byte trailer in PSS padding) |

Table 92: Cryptographic Parameters Attribute Structure

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client |
| Modifiable by server | No |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Re-key, Re-key Key Pair, Re-certify |
| Applies to Object Types | All Objects |

Table 93: Cryptographic Parameters Attribute Rules

## Cryptographic Usage Mask

The *Cryptographic Usage Mask* attribute defines the cryptographic usage of a key. This is a bit mask that indicates to the client which cryptographic functions MAY be performed using the key, and which ones SHALL NOT be performed.

The *State* of a managed object SHALL also be checked prior to depending on the *Cryptographic Usage Mask* as the mask represents the permitted usage subject to the *State* of the managed object. Changes to an object *State* SHALL NOT change the *Cryptographic Usage Mask*.

|  |  |
| --- | --- |
| Item | Encoding |
| Cryptographic Usage Mask | Integer |

Table 94: Cryptographic Usage Mask Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes (Except for Opaque Object) |
| Initially set by | Server or Client |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 95: Cryptographic Usage Mask Attribute Rules

## Deactivation Date

The *Deactivation Date* attribute is the date and time when the Managed Object SHALL NOT be used for any purpose, except for decryption, signature verification, or unwrapping, but only under extraordinary circumstances and only when special permission is granted. This time corresponds to state transition 6. This attribute SHALL NOT be changed or deleted before the object is destroyed, unless the object is in the Pre-Active or Active state.

|  |  |
| --- | --- |
| Item | Encoding |
| Deactivation Date | Date-Time |

Table 96: Deactivation Date Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server or Client |
| Modifiable by server | Yes, only while in Pre-Active or Active state |
| Modifiable by client | Yes, only while in Pre-Active or Active state |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Deactivate, Derive Key, Revoke Certify, Re-certify, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 97: Deactivation Date Attribute Rules

## Deactivation Reason

The Deactivation Reason attribute records the reason for an object’s deactivation (e.g., “Usage limit reached”, “Validity Date” passed, etc).

The Deactivation Message is an OPTIONAL field that is used exclusively for audit trail/logging purposes and MAY contain additional information about why the object was deactivated (e.g., “Machine decommissioned”).

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Deactivation Reason | Structure |  |
| Deactivation Reason Code | Enumeration | Yes |
| Deactivation Message | Text String | No |

Table 98: Deactivation Reason Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server or Client |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Deactivate |
| Applies to Object Types | All Objects |

Table 99: Deactivation Reason Attribute Rules

## Description

The Description attribute is used for descriptive purposes only. It is not used for policy enforcement. The attribute is set by the client or the server.

|  |  |
| --- | --- |
| Item | Encoding |
| Description | Text String |

Table 100: Description Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| Applies to Object Types | All Objects |

Table 101: Description Attribute Rules

## Destroy Date

The *Destroy Date* attribute is the date and time when the Managed Object was destroyed. This time corresponds to state transitions 2, 7, or 9 This value is set by the server when the object is destroyed due to the reception of a Destroy operation, or due to server policy or out-of-band administrative action.

|  |  |
| --- | --- |
| Item | Encoding |
| Destroy Date | Date-Time |

Table 102: Destroy Date Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Destroy |
| Applies to Object Types | All Objects |

Table 103: Destroy Date Attribute Rules

## Digest

The *Digest* attribute is a structure that contains the digest value of the key or secret data (i.e., digest of the Key Material), certificate (i.e., digest of the Certificate Value), or opaque object (i.e., digest of the Opaque Data Value). If the Key Material is a Byte String, then the Digest Value SHALL be calculated on this Byte String. If the Key Material is a structure, then the Digest Value SHALL be calculated on the TTLV-encoded Key Material structure. The Key Format Type field in the Digest attribute indicates the format of the Managed Object from which the Digest Value was calculated. Multiple digests MAY be calculated using different algorithms and/or key format types. If this attribute exists, then it SHALL have a mandatory attribute instance computed with the SHA-256 hashing algorithm and the default Key Value Format for this object type and algorithm. Clients may request via supplying a non-default Key Format Value attribute on operations that create a Managed Object, and the server SHALL produce an additional Digest attribute for that Key Value Type. The digest(s) are static and SHALL be set by the server when the object is created or registered, provided that the server has access to the Key Material or the Digest Value (possibly obtained via out-of-band mechanisms).

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Digest | Structure |  |
| Hashing Algorithm | Enumeration | Yes |
| Digest Value | Byte String | Yes, if the server has access to the Digest Value or the Key Material (for keys and secret data), the Certificate Value (for certificates) or the Opaque Data Value (for opaque objects). |
| Key Format Type | Enumeration | Yes, if the Managed Object is a key or secret data object. |

Table 104: Digest Attribute Structure

|  |  |
| --- | --- |
| SHALL always have a value | Yes, if the server has access to the Digest Value or the Key Material (for keys and secret data), the Certificate Value (for certificates) or the Opaque Data Value (for opaque objects). |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | Yes |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 105: Digest Attribute Rules

## Digital Signature Algorithm

The *Digital Signature Algorithm* attribute identifies the digital signature algorithm associated with a digitally signed object (e.g., Certificate). This attribute SHALL be set by the server when the object is created or registered and then SHALL NOT be changed or deleted before the object is destroyed.

|  |  |
| --- | --- |
| Item | Encoding |
| Digital Signature Algorithm | Enumeration |

Table 106: Digital Signature Algorithm Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | Yes for PGP keys. No for X.509 certificates. |
| When implicitly set | Certify, Re-certify, Register |
| Applies to Object Types | Certificates, PGP keys |

Table 107: Digital Signature Algorithm Attribute Rules

## Extractable

If False then the server SHALL prevent the object value being retrieved (via the Get operation). The server SHALL set its value to True if not provided by the client.

|  |  |
| --- | --- |
| Item | Encoding |
| Extractable | Boolean |

Table 108: Extractable Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes (but only from True to False) |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | When object is created or registered |
| Applies to Object Types | All Objects |

Table 109: Extractable Attribute Rules

## Fresh

The *Fresh* attribute is a Boolean attribute that indicates that the object has not yet been served to a client using a Get operation. The Fresh attribute SHALL be set to True when a new object is created on the server unless the client provides a False value in Register or Import. The server SHALL change the attribute value to False as soon as the object has been served via the Get operation to a client.

|  |  |
| --- | --- |
| Item | Encoding |
| Fresh | Boolean |

Table 110: Fresh Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key, Re-key Key Pair, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 111: Fresh Attribute Rules

## Initial Date

The *Initial Date* attribute contains the date and time when the Managed Object was first created or registered at the server. This time corresponds to state transition 1. This attribute SHALL be set by the server when the object is created or registered, and then SHALL NOT be changed or deleted before the object is destroyed. This attribute is also set for non-cryptographic objects when they are first registered with the server.

|  |  |
| --- | --- |
| Item | Encoding |
| Initial Date | Date-Time |

Table 112: Initial Date Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 113: Initial Date Attribute Rules

## Key Format Type

The Key Format Type attribute is a required attribute of a Cryptographic Object. It is set by the server, but a particular Key Format Type MAY be requested by the client if the cryptographic material is produced by the server (i.e., Create, Create Key Pair, Create Split Key, Re-key, Re-key Key Pair, Derive Key) on the client’s behalf. The server SHALL comply with the client’s requested format or SHALL fail the request. When the server calculates a Digest for the object, it SHALL compute the digest on the data in the assigned Key Format Type, as well as a digest in the default KMIP Key Format Type for that type of key and the algorithm requested (if a non-default value is specified).

|  |  |
| --- | --- |
| Object | Encoding |
| Key Format Type | Enumeration |

Table 114: Key Format Type Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| Applies to Object Types | All Objects |

Table 115: Key Format Type Attribute Rules

Keys have a default Key Format Type that SHALL be produced by KMIP servers. The default Key Format Type by object (and algorithm) is listed in the following table:

|  |  |
| --- | --- |
| Object | Default Key Format Type |
| Certificate | Raw |
| Certificate Request | PKCS#10 |
| Opaque Object | Opaque |
| PGP Key | Raw |
| Secret Data | Raw |
| Symmetric Key | Raw |
| Split Key | Raw |
| RSA Private Key | PKCS#1 |
| RSA Public Key | PKCS#1 |
| EC Private Key | Transparent EC Private Key |
| EC Public Key | Transparent EC Public Key |
| DSA Private Key | Transparent DSA Private Key |
| DSA Public Key | Transparent DSA Public Key |

Table 116: Default Key Format Type, by Object

## Key Part Identifier

The *Key Part Identifier* is specified by the client to indicate which key part is contained in the Split Key object.

|  |  |
| --- | --- |
| Item | Encoding |
| Key Part Identifier | Integer |

Table 117: Key Part Identifier Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Client |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create Split Key |
| Applies to Object Types | Split Key |

Table 118: Key Part Identifier Rules

## Key Value Location

*Key Value Location* MAY be specified by the client when the Key Value is omitted from the Key Block in a Register request. Key Value Location is used to indicate the location of the Key Value absent from the object being registered..

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Key Value Location | Structure |  |
| Key Value Location Value | Text String | Yes |
| Key Value Location Type | Enumeration | Yes |

Table 119: Key Value Location Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client |
| Modifiable by server | No |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Never |
| Applies to Object Types | All Objects |

Table 120: Key Value Location Attribute Rules

## Key Value Present

*Key Value Present* is an attribute of the managed object created by the server. It SHALL NOT be specified by the client in a Register request. *Key Value Present* SHALL be created by the server if the Key Value is absent from the Key Block in a Register request. The value of Key Value Present SHALL NOT be modified by either the client or the server. *Key Value Present* attribute MAY be used as a part of the Locate operation.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Key Value Present | Boolean | No |

Table 121: Key Value Present Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | During Register operation |
| Applies to Object Types | All Objects |

Table 122: Key Value Present Attribute Rules

## Last Change Date

The *Last Change Date* attribute contains the date and time of the last change of the specified object.

|  |  |
| --- | --- |
| Item | Encoding |
| Last Change Date | Date-Time |

Table 123: Last Change Date Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Activate, Deactivate, Revoke, Destroy, Archive, Recover, Certify, Re-certify, Re-key, Re-key Key Pair, Add Attribute, Modify Attribute, Delete Attribute, Get Usage Allocation |
| Applies to Object Types | All Objects |

Table 124: Last Change Date Attribute Rules

## Lease Time

The *Lease Time* attribute defines a time interval for a Managed Object beyond which the client SHALL NOT use the object without obtaining another lease. This attribute always holds the initial length of time allowed for a lease, and not the actual remaining time. Once its lease expires, the client is only able to renew the lease by calling Obtain Lease. A server SHALL store in this attribute the maximum Lease Time it is able to serve and a client obtains the lease time (with Obtain Lease) that is less than or equal to the maximum Lease Time. This attribute is read-only for clients. It SHALL be modified by the server only.

|  |  |
| --- | --- |
| Item | Encoding |
| Lease Time | Interval |

Table 125: Lease Time Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 126: Lease Time Attribute Rules

## Links

There are object attributes with name suffixes that identify themselves as “Link” attributes. These attributes serve to document a relationship from a particular Managed Object (the “source”) to another, closely related (the “target”) Managed Object. The prefix of the attribute name states the nature of the relationship between the source Managed Object and the target Managed Object. The Link identifies the target Managed Object by its Unique Identifier (or by a reference to a Unique Identifier in the current batch of operations) or by the Name of the Managed Object. Examples of such “Link” attributes would include the private key corresponding to a public key; the parent certificate for a certificate in a chain; or for a derived symmetric key, the base key from which it was derived.

A Name Reference indicates a link to whichever Managed Object has the matching Name attribute. A Reference is early-binding (to a specific unique object) and a Name Reference is late-binding (to whichever object currently has the matching Name attribute). The Managed Object that is the target of the Link MAY NOT exist (the object may not have been present in the key management system or may have been present and subsequently destroyed or obliterated.

“Link” attributes SHOULD be present for private keys and public keys for which a certificate chain is stored by the server, and for certificates in a certificate chain. Where possible, reverse relationships (e.g, the public key corresponding to a private key) SHOULD also be present, but in these cases the “source” and “target” roles are reversed.

Some “Link” attributes are multi-instance (e.g., the Symmetric Key or Secret Data objects that were derived from a base object), but usually “Link” attributes are single instance.

It is also possible that a Managed Object does not have “links” to associated cryptographic objects. This MAY occur in cases where the associated key material is not available to the server or client (e.g., the registration of a CA Signer certificate with a server, where the corresponding private key is held in a different manner).

|  |  |
| --- | --- |
| Encoding | Description |
| Enumeration | Unique Identifier Enumeration |
| Integer | Zero based nth Unique Identifier in the response. If negative the count is backwards from the beginning of the current operation’s batch item. |

Table 127: Linked Object Identifier encoding descriptions

### Certificate Link

The *Certificate Link* attribute expresses an association from related Managed Cryptographic Objects (e.g. Public Key(s) or Certificate) to a Certificate. The value identifies the target Certificate by its Unique Identifier (or functional equivalent). A public key may be associated with multiple Certificates, so the attribute is multi-valued. A Certificate in a certificate chain SHOULD have a *Certificate Link* to its parent certificate (if any).

It is also possible that a Managed Object does not have links to associated cryptographic objects. This MAY occur in cases where the associated key material is not made available to the server or client (e.g., the registration of a certificate with a server without the registration of the corresponding public key with a *Certificate Link* to that certificate).

|  |  |
| --- | --- |
| Item | Encoding |
| Certificate Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 128: Certificate Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Certify, Re-certify |
| Applies to Object Types | All Objects |

Table 129: Certificate Link Attribute Rules

### Certificate Request Link

The *Certificate Request Link* attribute expresses an association from related Managed Cryptographic Objects (e.g. Public Key(s) or Certificate) to a Certificate Request. The value identifies the target Certificate Request by its Unique Identifier (or functional equivalent). A certificate request may be associated with multiple Certificates, so the attribute is multi-valued. A Certificate produced from a Certificate Request as the result of a Certify or Re-certify Operation SHALL have a Certificate Request Link.

|  |  |
| --- | --- |
| Item | Encoding |
| Certificate Request Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 130: Certificate Request Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Certify, Re-certify |
| Applies to Object Types | All Objects |

Table 131: Certificate Request Link Attribute Rules

### Child Link

The *Child Link* attribute expresses an association (subordination, delegation, or other association) from one Managed Cryptographic Object to another. The value identifies the target Object by its Unique Identifier (or functional equivalent).

|  |  |
| --- | --- |
| Item | Encoding |
| Child Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 132: Child Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| When implicitly set | Never |
| Applies to Object Types | All Objects |

Table 133: Child Link Attribute Rules

### Credential Link

The *Credential Link* attribute expresses an association between Objects where the target is used as a Credential. The value identifies the target Object by its Unique Identifier (or functional equivalent).

The *Credential Link* target object will typically be one of a Certificate Managed Object or a Credentials System Object. Special authentication and authorization SHOULD be enforced when setting, modifying or deleting this attribute.

|  |  |
| --- | --- |
| Item | Encoding |
| Credential Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 134: Credential Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | N/A |
| Applies to Object Types | All Objects |

Table 135: Credential Link Attribute Rules

### Derivation Base Object Link

The *Derivation Base Object Link* attribute expresses an association from a derived Symmetric Key or Secret Data object to the Managed Cryptographic Object(s) from which it was derived. The value identifies the target base Object(s) by its/their Unique Identifier(s) (or functional equivalent).

|  |  |
| --- | --- |
| Item | Encoding |
| Derivation Base Object Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 136: Derivation Base Object Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Derive Key |
| Applies to Object Types | All Objects |

Table 137: Derivation Base Object Link Attribute Rules

### Derived Object Link

The *Derived Object Link* attribute identifies the Symmetric Keys or Secret Data objects derived from other Managed Cryptographic Objects. The value identifies the target Object by its Unique Identifier (or functional equivalent).

|  |  |
| --- | --- |
| Item | Encoding |
| Derived Object Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 138: Derived Object Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Derive Key |
| Applies to Object Types | All Objects |

Table 139: Derived Object Link Attribute Rules

### Group Link

The *Group Link* attribute identifies the group that a managed object is a member of. The value identifies the target Object by its Unique Identifier (or functional equivalent).

|  |  |
| --- | --- |
| Item | Encoding |
| Group Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 140: Group Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Never |
| Applies to Object Types | All Objects |

Table 141: Group Link Attribute Rules

### Joined Split Key Parts Link

The *Joined Split Key Parts Link* attribute identifies the joined key with the split key parts from which it was created (joined). The value identifies the target Object by its Unique Identifier (or functional equivalent).

|  |  |
| --- | --- |
| Item | Encoding |
| Joined Split Key Parts Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 142: Joined Split Key Parts Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Join Split Key |
| Applies to Object Types | All Objects |

Table 143: Joined Split Key Parts Link Attribute Rules

### Next Link

The *Next Link* attribute expresses an association from one Managed Cryptographic Object to the next one, typically in a Group. The value identifies the target Object by its Unique Identifier (or functional equivalent).

|  |  |
| --- | --- |
| Item | Encoding |
| Next Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 144: Next Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| When implicitly set | Never |
| Applies to Object Types | All Objects |

Table 145: Next Link Attribute Rules

### Parent Link

The *Parent Link* attribute expresses an association from one Managed Object to another. The association is that of super ordination, amalgamation, or sublimation into a larger whole, so effectively is the converse of Child Link. The value identifies the target Managed Object by its Unique Identifier (or functional equivalent).

|  |  |
| --- | --- |
| Item | Encoding |
| Parent Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 146: Parent Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Never |
| Applies to Object Types | Group |

Table 147: Parent Link Attribute Rules

### Password Link

The *Password Link* attribute expresses an association between Objects where the target is used as a password. The value identifies the target Object by its Unique Identifier (or functional equivalent).

The *Password Link* target object will typically be a Secret Data Managed Object or a Password Credential or Hashed Password Credential System Object.

Special authentication and authorization SHOULD be enforced when setting, modifying or deleting this attribute.

|  |  |
| --- | --- |
| Item | Encoding |
| Password Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 148: Password Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | N/A |
| Applies to Object Types | All Objects |

Table 149: Password Link Attribute Rules

### PKCS#12 Certificate Link

The *PKCS#12 Certificate Link* attribute expresses an association from a Private Key to an associated Certificate. The value identifies the target Certificate by its Unique Identifier (or functional equivalent).

|  |  |
| --- | --- |
| Item | Encoding |
| PKCS#12 Certificate Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 150: PKCS#12 Certificate Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| When implicitly set | Register (if KeyFormatType is PKCS#12) |
| Applies to Object Types | All Objects |

Table 151: PKCS#12 Certificate Link Attribute Rules

### PKCS#12 Password Link

The *PKCS#12 Password Link* attribute expresses an association from a Private Key to the Secret Data holding the password to be used to wrap/unwrap the PKCS#12 content delivered when the key is registered or gotten. The value identifies the target Secret Data object by its Unique Identifier (or functional equivalent).

|  |  |
| --- | --- |
| Item | Encoding |
| PKCS#12 Password Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 152: PKCS#12 Password Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| When implicitly set | Never |
| Applies to Object Types | All Objects |

Table 153: PKCS#12 Password Link Attribute Rules

### Previous Link

The *Previous Link* attribute expresses an association (e.g., chaining) from certain related Managed Cryptographic Objects to others, usually in a Group. The value identifies the target Object by its Unique Identifier (or functional equivalent).

|  |  |
| --- | --- |
| Item | Encoding |
| Previous Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 154: Previous Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| When implicitly set | Certify, Re-certify |
| Applies to Object Types | All Objects |

Table 155: Previous Link Attribute Rules

### Private Key Link

The *Private Key Link* attribute is an attribute used to express an association from a Public Key to its associated Private Key. The value identifies the target Object by its Unique Identifier (or functional equivalent).

It is also possible that a Public Key may have been Registered without the associated Private Key, so this link attribute may quite often be absent.

|  |  |
| --- | --- |
| Item | Encoding |
| Private Key Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 156: Private Key Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| When implicitly set | Certify, Re-certify |
| Applies to Object Types | All Objects |

Table 157: Private Key Link Attribute Rules

### Public Key Link

The *Private Key Link* attribute is an attribute used to express an association from certain related Managed Cryptographic Objects (e.g. Certificate, Private Key) to a Public Key. The value identifies the target object by its Unique Identifier (or functional equivalent)

|  |  |
| --- | --- |
| Item | Encoding |
| Public Key Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 158: Public Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| When implicitly set | Create Key Pair, Rekey Key Pair, Certify, Re-certify |
| Applies to Object Types | All Objects |

Table 159: Public Key Link Attribute Rules

### Replaced Object Link

The *Replaced Object Link* attribute is an attribute used to express an association from certain Managed Cryptographic Objects to their predecessor. For a symmetric Key, a Private Key or a Public Key, the link specifies the Unique Identifier of the key that was re-keyed to obtain the current object. For a Certificate, the link specifies the certificate that was re-certified to obtain the current one. The value identifies the target object by its Unique Identifier (or functional equivalent).

|  |  |
| --- | --- |
| Item | Encoding |
| Replaced Object Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 160: Replaced Object Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| When implicitly set | Re-certify, Rekey Key Pair |
| Applies to Object Types | All Objects |

Table 161: Replaced Object Link Attribute Rules

### Replacement Object Link

The *Replacement Object Link* attribute is an attribute used to express an association from outdated Managed Cryptographic Objects (e.g. Private Key, Public Key) to their replacements. For a Symmetric Key, a Private Key or a Public Key, the value refers to the key that resulted from the re-key of the current object. For a Certificate, the link refers to the certificate that resulted from the re-certification of the current one. There SHALL be only one replacement object per Managed Object

|  |  |
| --- | --- |
| Item | Encoding |
| Replacement Object Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 162: Replacement Object Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| When implicitly set | Re-certify, Rekey, Rekey Key Pair |
| Applies to Object Types | All Objects |

Table 163: Replacement Object Link Attribute Rules

### Split Key Base Link

The *Split Key Base Link* attribute associates the split key parts with the original object that was split. The value identifies the target Object by its Unique Identifier (or functional equivalent).

|  |  |
| --- | --- |
| Item | Encoding |
| Split Key Base Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 164: Split Key Base Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Create Split Key |
| Applies to Object Types | All Objects |

Table 165: Split Key Base Link Attribute Rules

### Wrapping Key Link

The *Wrapping Key Link* attribute is an attribute used to express an association from a wrapped key to the Managed Cryptographic Object used to wrap (encrypt) it.

|  |  |
| --- | --- |
| Item | Encoding |
| Wrapping Key Link | Reference or Name Reference or Unique Identifier Enumeration or Integer |

Table 166: Wrapping Key Link Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| When implicitly set | Register |
| Applies to Object Types | All Objects |

Table 167: Wrapping Key Link Attribute Rules

## Name

The *Name* attribute is a text string used to identify and locate an object. This attribute is assigned by the client, and the *Name* is intended to be in a form that humans are able to interpret. The key management system MAY specify rules by which the client creates valid names. Clients are informed of such rules by a mechanism that is not specified by this standard. Names SHALL be unique within a given key management server, but are NOT REQUIRED to be globally unique.

|  |  |
| --- | --- |
| Item | Encoding |
| Name | Text String |

Table 168: Name Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Re-key, Re-key Key Pair, Re-certify |
| Applies to Object Types | All Objects |

Table 169: Name Attribute Rules

## Never Extractable

The server SHALL create this attribute, and set it to True if the Extractable attribute has always been False.

The server SHALL set it to False if the Extractable attribute has ever been set to True.

|  |  |
| --- | --- |
| Item | Encoding |
| Never Extractable | Boolean |

Table 170: Never Extractable Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | When Never Extractable attribute is  set or changed |
| Applies to Object Types | All Objects |

Table 171: Never Extractable Attribute Rules

## NIST Key Type

The NIST SP800-57 Key Type is an attribute of a Key (or Secret Data object). It MAY be set by the client, preferably when the object is registered or created. Although the attribute is optional, once set, MAY NOT be deleted or modified by either the client or the server. This attribute is intended to reflect the NIST SP-800-57 view of cryptographic material, so an object SHOULD have only one usage (see [SP800-57-1] for rationale), but this is not enforced at the server.

|  |  |  |
| --- | --- | --- |
| Item |  | **Encoding** |
| NIST Key Type |  | Enumeration |

Table 172 SP800-57 Key Type Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | Yes |
| Applies to Object Types | All Objects |

Table 173 SP800-57 Key Type Attribute Rules

## NIST Security Category

The NIST Security Category is a number associated with the security strength of a post-quantum cryptographic algorithm as specified by NIST. FIPS-203, FIPS-204 and FIPS-205 define the current meaning of the category values.

|  |  |  |
| --- | --- | --- |
| Item |  | **Encoding** |
| NIST Security Category |  | Integer |

Table 172 NIST Security Category Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| Applies to Object Types | All Objects |

Table 173 NIST Security Category Attribute Rules

## Object Class

All Managed Objects SHALL have a specified Object Class. The object class for a managed object is immutable. If an Object Class is not explicitly specified when creating an object, the Object Class SHALL be User.

|  |  |
| --- | --- |
| Item | Encoding |
| Object Class | Enumeration |

Table 174: Object Class Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Client or Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 175: Object Class Attribute Rules

## Object Type

The *Object Type* of a Managed Object (e.g., public key, private key, symmetric key, etc.) SHALL be set by the server when the object is created or registered and then SHALL NOT be changed or deleted before the object is destroyed.

|  |  |
| --- | --- |
| Item | Encoding |
| Object Type | Enumeration |

Table 176: Object Type Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 177: Object Type Attribute Rules

## Opaque Data Type

The *Opaque Data Type* of an Opaque Object SHALL be set by the server when the object is registered and then SHALL NOT be changed or deleted before the object is destroyed.

|  |  |
| --- | --- |
| Item | Encoding |
| Opaque Data Type | Enumeration |

Table 178: Opaque Data Type Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Register |
| Applies to Object Types | Opaque Objects |

Table 179: Opaque Data Type Attribute Rules

## Original Creation Date

The *Original Creation Date* attribute contains the date and time the object was originally created, which can be different from when the object is registered with a key management server.

It is OPTIONAL for an object being registered by a client. The *Original Creation Date* MAY be set by the client during a Register operation. If no *Original Creation Date* attribute was set by the client during a Register operation, it MAY do so at a later time through an Add Attribute operation for that object.

It is mandatory for an object created on the server as a result of a Create, Create Key Pair, Derive Key, Re-key, or Re-key Key Pair operation. In such cases the *Original Creation Date* SHALL be set by the server and SHALL be the same as the *Initial Date* attribute.

In all cases, once the *Original Creation Date* is set, it SHALL NOT be deleted or updated.

|  |  |
| --- | --- |
| Item | Encoding |
| Original Creation Date | Date-Time |

Table 180: Original Creation Date Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server (when object is generated by Server) |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Derive Key, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 181: Original Creation Date Attribute Rules

## OTP Counter

A One Time Password Counter Credential System Object MAY require counters be tracked for the authentication process depending on the OTP Algorithm in used. If the OTP Algorithm requires multiple counters the server SHALL encode those counters in this attribute.

|  |  |
| --- | --- |
| Item | Encoding |
| OTP Counter | Byte String |

Table 182: OTP Counter Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | N/A |
| Applies to Object Types | All Objects |

Table 183: OTP Counter Attribute Rules

## PKCS#12 Friendly Name

PKCS#12 Friendly Name is an attribute used for descriptive purposes. If supplied on a Register Private Key with Key Format Type PKCS#12, it informs the server of the alias/friendly name (see [RFC7292]) under which the private key and its associated certificate chain SHALL be found in the Key Material. If no such alias/friendly name is supplied, the server SHALL record the alias/friendly name (if any) it finds for the first Private Key in the Key Material.

When a Get with Key Format Type PKCS#12 is issued, this attribute informs the server what alias/friendly name the server SHALL use when encoding the response. If this attribute is absent for the object on which the Get is issued, the server SHOULD use an alias/friendly name of “alias”. Since the PKCS#12 Friendly Name is defined in ASN.1 with an EQUALITY MATCHING RULE of caseIgnoreMatch, clients and servers SHOULD utilize a lowercase text string.

|  |  |
| --- | --- |
| Item | Encoding |
| PKCS#12 Friendly Name | Text String |

Table 184: PKCS#12 Friendly Name Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | No |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| Applies to Object Types | All Objects |

Table 185: Friendly Name Attribute Rules

## Process Start Date

The *Process Start Date* attribute is the date and time when a valid Managed Object MAY begin to be used to process cryptographically protected information (e.g., decryption or unwrapping), depending on the value of its Cryptographic Usage Mask attribute. The object SHALL NOT be used for these cryptographic purposes before the *Process Start Date* has been reached. This value MAY be equal to or later than, but SHALL NOT precede, the Activation Date. Once the Process Start Date has occurred, then this attribute SHALL NOT be changed or deleted before the object is destroyed.

|  |  |
| --- | --- |
| Item | Encoding |
| Process Start Date | Date-Time |

Table 186: Process Start Date Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server or Client |
| Modifiable by server | Yes, only while in Pre-Active or Active state and as long as the Process Start Date has been not reached. |
| Modifiable by client | Yes, only while in Pre-Active or Active state and as long as the Process Start Date has been not reached. |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Register, Derive Key, Re-key |
| Applies to Object Types | All Objects |

Table 187: Process Start Date Attribute Rules

## Protect Stop Date

The *Protect Stop Date* attribute is the date and time after which a valid Managed Object SHALL NOT be used for applying cryptographic protection (e.g., encryption or wrapping), depending on the value of its Cryptographic Usage Mask attribute. This value MAY be equal to or earlier than, but SHALL NOT be later than the Deactivation Date. Once the *Protect Stop Date* has occurred, then this attribute SHALL NOT be changed or deleted before the object is destroyed.

|  |  |
| --- | --- |
| Item | Encoding |
| Protect Stop Date | Date-Time |

Table 188: Protect Stop Date Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server or Client |
| Modifiable by server | Yes, only while in Pre-Active or Active state and as long as the Protect Stop Date has not been reached. |
| Modifiable by client | Yes, only while in Pre-Active or Active state and as long as the Protect Stop Date has not been reached. |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Register, Derive Key, Re-key |
| Applies to Object Types | All Objects |

Table 189: Protect Stop Date Attribute Rules

## Protection Level

The *Protection Level* attribute is the Level of protection required for a given object.

|  |  |
| --- | --- |
| Item | Encoding |
| Protection Level | Enumeration |

Table 190: Protection Level Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client |
| Modifiable by server | No |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| When implicitly set |  |
| Applies to Object Types | All Objects |

Table 191: Protection Level Attribute Rules

## Protection Period

The *Protection Period* attribute is the period of time for which the output of an operation or a *Managed Cryptographic Object* SHALL remain safe. The *Protection Period* is specified as an *Interval*.

|  |  |
| --- | --- |
| Item | Encoding |
| Protection Period | Interval |

Table 192: Protection Period Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client |
| Modifiable by server | No |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| When implicitly set |  |
| Applies to Object Types | All Objects |

Table 193: Protection Period Attribute Rules

## Protection Storage Mask

The *Protection Storage Mask* attribute records which of the requested mask values have been used for protection storage.

|  |  |
| --- | --- |
| Item | Encoding |
| Protection Storage Mask | Integer |

Table 194: Protection Storage Mask

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | When object is stored |
| Applies to Object Types | All Objects |

Table 195: Protection Storage Mask Rules

## Quantum Safe

The *Quantum Safe* attribute is a flag to be set to indicate an object is required to be Quantum Safe for the given *Protection Period* and *Protection Level.*

|  |  |
| --- | --- |
| Item | Encoding |
| Quantum Safe | Boolean |

Table 196: Quantum Safe Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client |
| Modifiable by server | No |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| When implicitly set |  |
| Applies to Object Types | All Objects |

Table 197: Quantum Safe Attribute Rules

## Random Number Generator

The *Random Number Generator* attribute contains the details of the random number generator used during the creation of the managed cryptographic object.

The *Random Number Generator* MAY be set by the client during a Register operation. If no *Random Number Generator* attribute was set by the client during a Register operation, it MAY do so at a later time through an Add Attribute operation for that object.

It is mandatory for an object created on the server as a result of a Create, Create Key Pair, Derive Key, Re-key, or Re-key Key Pair operation. In such cases the *Random Number Generator* SHALL be set by the server depending on which random number generator was used. If the specific details of the random number generator are unknown then the RNG Algorithm within the RNG Parameters structure SHALL be set to *Unspecified*.

If one or more *Random Number Generator* attribute values are provided in the Attributes in a Create, Create Key Pair, Derive Key, Re-key, or Re-key Key Pair operation then the server SHALL use a random number generator that matches one of the *Random Number Generator* attributes. If the server does not support or is otherwise unable to use a matching random number generator then it SHALL fail the request.

The *Random Number Generator* attribute SHALL NOT be copied from the original object in a Re-key or Re-key Key Pair operation.

In all cases, once the *Random Number Generator* attribute is set, it SHALL NOT be deleted or updated.

Note: the encoding is the same as the RNG Parameters structure using a different tag; it does not contain a nested RNG Parameters structure.

|  |  |
| --- | --- |
| Item | Encoding |
| Random Number Generator | RNG Parameters |

Table 198: Random Number Generator Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client (when the object is generated by the Client and registered) or Server (when object is generated by Server) |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Derive Key, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 199: Random Number Generator Attribute Rules

## Revocation Reason

The *Revocation Reason* attribute is a structure used to indicate why the Managed Cryptographic Object was revoked (e.g., “compromised”, “expired”, “no longer used”, etc.). This attribute is only set by the server as a part of the Revoke Operation.

The *Revocation Message* is an OPTIONAL field that is used exclusively for audit trail/logging purposes and MAY contain additional information about why the object was revoked (e.g., “Laptop stolen”

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Revocation Reason | Structure |  |
| Revocation Reason Code | Enumeration | Yes |
| Revocation Message | Text String | No |

Table 200: Revocation Reason Attribute Structure

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Revoke |
| Applies to Object Types | All Objects |

Table 201: Revocation Reason Attribute Rules

## Rotate Automatic

If set to True, specifies the Managed Object will be automatically rotated by the server using the Rotate Interval via the equivalent of the ReKey, ReKeyKeyPair or ReCertify operation performed by the server.

|  |  |
| --- | --- |
| Item | Encoding |
| Rotate Automatic | Boolean |

Table 202: Rotate Automatic Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | N/A |
| Applies to Object Types | All Objects |

Table 203: Rotate Automatic Attribute Rules

## Rotate Date

The time when the Managed Object was rotated.

|  |  |
| --- | --- |
| Item | Encoding |
| Rotate Date | Date Time |

Table 204: Rotate Date Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | When object is rotated |
| Applies to Object Types | All Objects |

Table 205: Rotate Date Attribute Rules

## Rotate Generation

The count from zero of the number of automatic rotates or ReKey, ReKeyKeyPair, or ReCertify operations that have occurred in order to create this Managed Object.

|  |  |
| --- | --- |
| Item | Encoding |
| Rotate Generation | Integer |

Table 206: Rotate Generation Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | When object is rotated |
| Applies to Object Types | All Objects |

Table 207: Rotate Generation Attribute Rules

## Rotate Interval

If set and *Rotate Automatic* is set to True, then automatic rotation of the Managed Object is performed by the server when the difference between the *Initial Date* of the Managed Object and the current server time reaches or exceeds this value.

|  |  |
| --- | --- |
| Item | Encoding |
| Rotate Interval | Interval |

Table 208: Rotate Interval Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | When created or registered |
| Applies to Object Types | All Objects |

Table 209: Rotate Interval Attribute Rules

## Rotate Latest

If set to True, specifies the Managed Object is the most recent object of the set of rotated Managed Objects.

|  |  |
| --- | --- |
| Item | Encoding |
| Rotate Latest | Boolean |

Table 210: Rotate Latest Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | When object is rotated by the server |
| Applies to Object Types | All Objects |

Table 211: Rotate Latest Attribute Rules

## Rotate Name

The *Rotate Name* attribute is a text string used to identify a set of managed objects that have been rotated. This attribute is assigned by the client, and the *Rotate Name* is intended to be in a form that humans are able to interpret. The key management system MAY specify rules by which the client creates valid rotate names. Clients are informed of such rules by a mechanism that is not specified by this standard. Rotate Names MAY NOT be unique within a given key management server.

|  |  |
| --- | --- |
| Item | Encoding |
| Rotate Name | Text String |
|  |  |
|  |  |

Table 212: Rotate Name Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client |
| Modifiable by server | No |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| Applies to Object Types | All Objects |

Table 213: Rotate Name Attribute Rules

## Rotate Offset

When automatic rotation of the Managed Object is performed by the server, specifies the Offset value to use in the equivalent of the ReKey, ReKeyKeyPair or ReCertify operation performed by the server.

|  |  |
| --- | --- |
| Item | Encoding |
| Rotate Offset | Interval |

Table 214: Rotate Offset Attribute

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | When object is created or registered |
| Applies to Object Types | All Objects |

Table 215: Rotate Offset Attribute Rules

## Sensitive

If True then the server SHALL prevent the object value being retrieved (via the Get operation) unless it is wrapped by another key. The server SHALL set the value to False if the value is not provided by the client.

|  |  |
| --- | --- |
| Item | Encoding |
| Sensitive | Boolean |

Table 216: Sensitive Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes (but only from False to True) |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | When object is created or registered |
| Applies to Object Types | All Objects |

Table 217: Sensitive Attribute Rules

## Short Unique Identifier

The *Short Unique Identifier* is generated by the key management system to uniquely identify a Managed Object using a shorter identifier. It is only REQUIRED to be unique within the identifier space managed by a single key management system, however this identifier SHOULD be globally unique in order to allow for a key management server export of such objects. This attribute SHALL be assigned by the key management system upon creation or registration of a *Unique Identifier*, and then SHALL NOT be changed or deleted before the object is destroyed.

The *Short Unique Identifier* SHOULD be generated as a SHA-256 hash of the *Unique Identifier* and SHALL be a 32 byte *byte string*.

|  |  |
| --- | --- |
| Item | Encoding |
| Short Unique Identifier | Byte String |

Table 218: Unique Identifier Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 219: Short Unique Identifier Attribute Rules

## Split Key Polynomial

The Split Key Polynomial is specified by the client in order to determine the characteristics of how Galois Field (GF) split key with Polynomial Sharing is conducted by the Server. Given variants of 283 and 285 the value must be specified to be interoperable between systems.

|  |  |
| --- | --- |
| Item | Encoding |
| Split Key Polynomial | Enumeration |

Table 220: Split Key Polynomial Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Client |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create Split Key, Join Key |
| Applies to Object Types | All Objects |

Table 221: Split Key Polynomial Rules

## Split Key Method

The *Split Key Method* is specified by the client to indicate the method used to create the Split Key Parts

|  |  |
| --- | --- |
| Item | Encoding |
| Split Key Method | Enumeration |

Table 222: Split Key Method Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Client |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create Split Key |
| Applies to Object Types | Split Key |

Table 223: Split Key Method Rules

## Split Key Parts

The *Split Key Parts* is specified by the client to indicate the number of parts into which a Split Key will be split.

|  |  |
| --- | --- |
| Item | Encoding |
| Split Key Parts | Integer |

Table 224: Split Key Parts Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Client |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create Split Key |
| Applies to Object Types | Split Key |

Table 225: Split Key Parts Rules

## Split Key Threshold

The *Split Key Threshold* is specified by the client to indicate the number of parts required the minimum number of parts needed to reconstruct the entire key.

|  |  |
| --- | --- |
| Item | Encoding |
| Split Key Threshold | Integer |

Table 226: Split Key Threshold Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Client |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create Split Key |
| Applies to Object Types | Split Key |

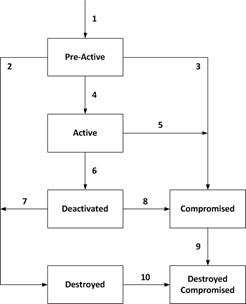
Table 227: Split Key Threshold Rules

## State

This attribute is an indication of the *State* of an object as known to the key management server. The State SHALL NOT be changed by using the Modify Attribute operation on this attribute. The State SHALL only be changed by the server as a part of other operations or other server processes. An object SHALL be in one of the following states at any given time.

Note: The states correspond to those described in [SP800-57-1].

Figure 1: Cryptographic Object States and Transitions



* *Pre-Active*: The object exists and SHALL NOT be used for any cryptographic purpose.
* *Active*: The object SHALL be transitioned to the *Active* state prior to being used for any cryptographic purpose. The object SHALL only be used for all cryptographic purposes that are allowed by its Cryptographic Usage Mask attribute. If a Process Start Date attribute is set, then the object SHALL NOT be used for cryptographic purposes prior to the Process Start Date. If a Protect Stop attribute is set, then the object SHALL NOT be used for cryptographic purposes after the Process Stop Date.
* *Deactivated*: The object SHALL NOT be used for applying cryptographic protection (e.g., encryption, signing, wrapping, MACing, deriving) . The object SHALL only be used for cryptographic purposes permitted by the Cryptographic Usage Mask attribute. The object SHOULD only be used to process cryptographically-protected information (e.g., decryption, signature verification, unwrapping, MAC verification under extraordinary circumstances and when special permission is granted.
* *Compromised*: The object SHALL NOT be used for applying cryptographic protection (e.g., encryption, signing, wrapping, MACing, deriving). The object SHOULD only be used to process cryptographically-protected information (e.g., decryption, signature verification, unwrapping, MAC verification in a client that is trusted to use managed objects that have been compromised. The object SHALL only be used for cryptographic purposes permitted by the Cryptographic Usage Mask attribute.
* *Destroyed*: The object SHALL NOT be used for any cryptographic purpose.
* *Destroyed Compromised*: The object SHALL NOT be used for any cryptographic purpose; however its compromised status SHOULD be retained for audit or security purposes.

State transitions occur as follows:

1. The transition from a non-existent key to the Pre-Active state is caused by the creation of the object. When an object is created or registered, it automatically goes from non-existent to Pre-Active. If, however, the operation that creates or registers the object contains an Activation Date that has already occurred, then the state immediately transitions from Pre-Active to Active. In this case, the server SHALL set the Activation Date attribute to the value specified in the request, or fail the request attempting to create or register the object, depending on server policy. If the operation contains an Activation Date attribute that is in the future, or contains no Activation Date, then the Cryptographic Object is initialized in the key management system in the Pre-Active state.
2. The transition from Pre-Active to Destroyed is caused by a client issuing a Destroy operation. The server destroys the object when (and if) server policy dictates.
3. The transition from Pre-Active to Compromised is caused by a client issuing a Revoke operation with a Revocation Reason containing a *Revocation Reason Code* of *Key Compromise* or *CA Compromise*.
4. The transition from Pre-Active to Active SHALL occur in one of three ways:

* The Activation Date is reached,
* A client successfully issues a Modify Attribute operation, modifying the Activation Date to a date in the past, or the current date, or
* A client issues an Activate operation on the object. The server SHALL set the Activation Date to the time the Activate operation is received.

1. The transition from Active to Compromised is caused by a client issuing a Revoke operation with a Revocation Reason containing a *Revocation Reason Code* of Compromised.
2. The transition from Active to Deactivated SHALL occur in one of three ways:

* The object's Deactivation Date is reached,
* The object’s Protect Stop Date is reached,
* The object’s Usage Limit is reached,
* A client issues a Deactivate operation, or
* The client successfully issues a Modify Attribute operation, modifying the Deactivation Date to a date in the past, or the current date.

1. The transition from Deactivated to Destroyed is caused by a client issuing a Destroy operation, or by a server, both in accordance with server policy. The server destroys the object when (and if) server policy dictates.
2. The transition from Deactivated to Compromised is caused by a client issuing a Revoke operation with a Revocation Reason containing a *Revocation Reason Code* of Compromised.
3. The transition from Compromised to Destroyed Compromised is caused by a client issuing a Destroy operation, or by a server, both in accordance with server policy. The server destroys the object when (and if) server policy dictates.
4. The transition from Destroyed to Destroyed Compromised is caused by a client issuing a *R*evoke operation with a Revocation Reason containing a *Revocation Reason Code* of Compromised.

Only the transitions described above are permitted.

|  |  |
| --- | --- |
| Item | Encoding |
| State | Enumeration |

Table 228: State Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | Yes |
| Modifiable by client | No, but only by the server in response to certain requests (see above) |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Activate, Deactivate, Revoke, Destroy, Certify, Re-certify, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 229: State Attribute Rules

## Unique Identifier

The *Unique Identifier* is generated by the key management system to uniquely identify a Managed Object. It is only REQUIRED to be unique within the identifier space managed by a single key management system, however this identifier SHOULD be globally unique in order to allow for a key management server export of such objects. This attribute SHALL be assigned by the key management system at creation or registration time, and then SHALL NOT be changed or deleted before the object is obliterated.

It is recommended that where possible servers SHOULD use a Universally Unique Identifier following one of the formats specified in [RFC9562].

When a Unique Identifier is used to refer to a managed object, generally the encoding should be specified as Reference (or Name Reference). When the Unique Identifier is in the context of the attributes of a specific managed object (i.e. when registering or creating a managed object or returning a managed object’s attributes), generally the encoding should be specified as Identifier.

Within protocol messages, the *Unique Identifier* may be referred with additional context in the form of *Private Key Unique Identifier*, *Public Key Unique Identifier*, *Certificate Request Unique Identifier*, or *Replaced Unique Identifier*. In all of these contexts, the same encoding descriptions apply.

|  |  |
| --- | --- |
| Encoding | Description |
| Identifier | Unique Identifier of a Managed Object. |
| Enumeration | When used outside the context of an attribute of an object (i.e. within protocol messages outside of the Attributes structure), Unique Identifier Enumeration |
| Integer | When used outside the context of an attribute of an object (i.e. within protocol messages outside of the Attributes structure), Zero based nth Unique Identifier in the response. If negative the count is backwards from the beginning of the current operation’s batch item. |
| Reference | When used outside the context of an attribute of an object (i.e. within protocol messages outside of the Attributes structure) |
| Name Reference | When used outside the context of an attribute of an object (i.e. within protocol messages outside of the Attributes structure) |

Table 230: Unique Identifier encoding descriptions

|  |  |
| --- | --- |
| Item | Encoding |
| Unique Identifier | Enumeration, Integer or Identifier |

Table 231: Unique Identifier Attribute

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key, Re-key Key Pair |
| Applies to Object Types | All Objects |

Table 232: Unique Identifier Attribute Rules

## Usage Limits

The *Usage Limits* attribute is a mechanism for limiting the usage of a Managed Object. It only applies to Managed Objects that are able to be used for applying cryptographic protection and it SHALL only reflect their usage for applying that protection (e.g., encryption, signing, etc.). This attribute does not necessarily exist for all Managed Cryptographic Objects, since some objects are able to be used without limit for cryptographically protecting data, depending on client/server policies. Usage for processing cryptographically protected data (e.g., decryption, verification, etc.) is not limited. The Usage Limits attribute contains the Usage Limit structure which has the three following fields:

|  |  |
| --- | --- |
| **Value** | **Description** |
| Usage Limits Total | The total number of Usage Limits Units allowed to be protected. This is the total value for the entire life of the object and SHALL NOT be changed once the object begins to be used for applying cryptographic protection. |
| Usage Limits Count | The currently remaining number of Usage Limits Units allowed to be protected by the object. |
| Usage Limits Unit | The type of quantity for which this structure specifies a usage limit (e.g., byte, object). |

Table 233: Usage Limits Descriptions

When the attribute is initially set (usually during object creation or registration), the Usage Limits Count is set to the Usage Limits Total value allowed for the useful life of the object, and are decremented when the object is used. The server SHALL ignore the Usage Limits Count value if the attribute is specified in an operation that creates a new object. Changes made via the Modify Attribute operation reflect corrections to the Usage Limits Total value, but they SHALL NOT be changed once the Usage Limits Count value has changed by a Get Usage Allocation operation. The Usage Limits Count value SHALL NOT be set or modified by the client via the Add Attribute or Modify Attribute operations.

|  |  |
| --- | --- |
| SHALL always have a value | No |
| Initially set by | Server (Usage Limits Total, Usage Limits Count, and Usage Limits Unit) or Client (Usage Limits Total and/or Usage Limits Unit only) |
| Modifiable by server | Yes |
| Modifiable by client | Yes (Usage Limits Total and/or Usage Limits Unit only, as long as Get Usage Allocation has not been performed) |
| Deletable by client | Yes, as long as Get Usage Allocation has not been performed |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Re-key, Re-key Key Pair, Get Usage Allocation |
| Applies to Object Types | All Objects |

Table 234: Usage Limits Attribute Rules

## Vendor Attribute

A vendor specific Attribute is a structure used for sending and receiving a Managed Object attribute. The *Vendor Identification* and *Attribute Name* are text-strings that are used to identify the attribute. The *Attribute Value* is either a primitive data type or structured object, depending on the attribute. Vendor identification values “x” and “y” are reserved for KMIP v2.0 and later implementations referencing KMIP v1.x Custom Attributes.

Vendor Attributes created by the client with Vendor Identification “x” are not created (provided during object creation), set, added, adjusted, modified or deleted by the server.

Vendor Attributes created by the server with Vendor Identification “y” are not created (provided during object creation), set, added, adjusted, modified or deleted by the client.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Attribute | Structure |  |
| Vendor Identification | Text String (with usage limited to alphanumeric, underscore and period – i.e. [A-Za-z0-9\_.]) | Yes |
| Attribute Name | Text String | Yes |
| Attribute Value | Varies, depending on attribute. | Yes, except for the Notify operation |

Table 235: Attribute Object Structure

## X.509 Certificate Identifier

The *X.509* *Certificate Identifier* attribute is a structure used to provide the identification of an X.509 public key certificate. The X.509 Certificate Identifier contains the Issuer Distinguished Name (i.e., from the Issuer field of the X.509 certificate) and the Certificate Serial Number (i.e., from the Serial Number field of the X.509 certificate). The X.509 Certificate Identifier SHALL be set by the server when the X.509 certificate is created or registered and then SHALL NOT be changed or deleted before the object is destroyed.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| X.509 Certificate Identifier | Structure |  |
| Issuer Distinguished Name | Byte String | Yes |
| Certificate Serial Number | Byte String | Yes |

Table 236: X.509 Certificate Identifier Attribute Structure

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Register, Certify, Re-certify |
| Applies to Object Types | X.509 Certificates |

Table 237: X.509 Certificate Identifier Attribute Rules

## X.509 Certificate Issuer

The *X.509 Certificate Issuer* attribute is a structure used to identify the issuer of a X.509 certificate, containing the Issuer Distinguished Name (i.e., from the Issuer field of the X.509 certificate). It MAY include one or more alternative names (e.g., email address, IP address, DNS name) for the issuer of the certificate (i.e., from the Issuer Alternative Name extension within the X.509 certificate). The server SHALL set these values based on the information it extracts from a X.509 certificate that is created as a result of a Certify or a Re-certify operation or is sent as part of a Register operation. These values SHALL NOT be changed or deleted before the object is destroyed.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| X.509 Certificate Issuer | Structure |  |
| Issuer Distinguished Name | Byte String | Yes |
| Issuer Alternative Name | Byte String, MAY be repeated | No |

Table 238: X.509 Certificate Issuer Attribute Structure

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Register, Certify, Re-certify |
| Applies to Object Types | X.509 Certificates |

Table 239: X.509 Certificate Issuer Attribute Rules

## X.509 Certificate Subject

The *X.509 Certificate Subject* attribute is a structure used to identify the subject of a X.509 certificate or a X.509 Certificate Request. The X.509 Certificate Subject contains the Subject Distinguished Name (i.e., from the Subject field of the X.509 certificate). It MAY include one or more alternative names (e.g., email address, IP address, DNS name) for the subject of the X.509 certificate (i.e., from the Subject Alternative Name extension within the X.509 certificate). The X.509 Certificate Subject SHALL be set by the server based on the information it extracts from the X.509 certificate that is created (as a result of a Certify or a Re-certify operation) or registered (as part of a Register operation) and SHALL NOT be changed or deleted before the object is destroyed.

If the Subject Alternative Name extension is included in the X.509 certificate and is marked critical within the X.509 certificate itself, then an X.509 certificate MAY be issued with the subject field left blank. Therefore an empty string is an acceptable value for the Subject Distinguished Name.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| X.509 Certificate Subject | Structure |  |
| Subject Distinguished Name | Byte String | Yes, but MAY be the empty string |
| Subject Alternative Name | Byte String, MAY be repeated | Yes, if the Subject Distinguished Name is an empty string. |

Table 240: X.509 Certificate Subject Attribute Structure

|  |  |
| --- | --- |
| SHALL always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Register, Certify, Re-certify |
| Applies to Object Types | X.509 Certificates |

Table 241: X.509 Certificate Subject Attribute Rules

# Attribute Data Structures

## Attributes

This structure is used in various operations to provide the desired attribute values in the request and to return the actual attribute values in the response.

The *Attributes* structure is defined as follows:

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Attributes | Structure |  |
| *Any attribute in §4 - Object Attributes* | *Any*, MAY be repeated | No |

Table 242: Attributes Definition

## Common Attributes

This structure is used in various operations to provide the desired attribute values in the request and to return the actual attribute values in the response.

The *Common* *Attributes* structure is defined as follows:

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Common Attributes | Structure |  |
| *Any attribute in §4 - Object Attributes* | *Any*, MAY be repeated | No |

Table 243: Common Attributes Definition

## Private Key Attributes

This structure is used in various operations to provide the desired attribute values in the request and to return the actual attribute values in the response.

The *Private Key* *Attributes* structure is defined as follows:

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Private Key Attributes | Structure |  |
| *Any attribute in §4 - Object Attributes* | *Any*, MAY be repeated | No |

*Table 244: Private Key Attributes Definition*

## Public Key Attributes

This structure is used in various operations to provide the desired attribute values in the request and to return the actual attribute values in the response.

The *Public Key* *Attributes* structure is defined as follows:

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Public Key Attributes | Structure |  |
| *Any attribute in §4 - Object Attributes* | *Any*, MAY be repeated | No |

*Table 245: Public Key Attributes Definition*

## Attribute Reference

These structures are used in various operations to provide reference to an attribute by name or by tag in a request or response.

The *Attribute Reference* definition is as follows:

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Attribute Reference | Structure |  |
| *Vendor Identification* | Text String (with usage limited to alphanumeric, underscore and period – i.e. [A-Za-z0-9\_.]) | Yes |
| *Attribute Name* | *Text String* | Yes |

OR

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Attribute Reference | Enumeration (Tag) | Yes |

Table 246: Attribute Reference Definition

## Current Attribute

Structure used in various operations to provide the *Current Attribute* value in the request.

The *Current Attribute* structure is defined identically as follows:

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Current Attribute | Structure |  |
| *Any attribute in §4 - Object Attributes* | *Any* | Yes |

Table 247: Current Attribute Definition

## New Attribute

Structure used in various operations to provide the *New Attribute* value in the request.

The *New Attribute* structure is defined identically as follows:

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| New Attribute | Structure |  |
| *Any attribute in §4 - Object Attributes* | *Any* | Yes |

Table 248: New Attribute Definition

# Operations

## Client-to-Server Operations

The following subsections describe the operations that MAY be requested by a key management client. Not all clients have to be capable of issuing all operation requests; however any client that issues a specific request SHALL be capable of understanding the response to the request. All Object Management operations are issued in requests from clients to servers, and results obtained in responses from servers to clients. Multiple operations MAY be combined within a batch, resulting in a single request/response message pair.

A number of the operations whose descriptions follow are affected by a mechanism referred to as the *ID Placeholder.*

The key management server SHALL implement a temporary variable called the ID Placeholder. This value consists of a single Unique Identifier. It is a variable stored inside the server that is only valid and preserved during the execution of a batch of operations. Once the batch of operations has been completed, the ID Placeholder value SHALL be discarded and/or invalidated by the server, so that subsequent requests do not find this previous ID Placeholder available. The Check operation clears the ID Placeholder if the requested Check fails.

The ID Placeholder is obtained from the Unique Identifier returned in response to any operations that successfully completes and returns a Unique Identifier. The server SHALL copy the Unique Identifier into the ID Placeholder variable, where it is held until the completion of the operations remaining in the batched request or until a subsequent operation in the batch causes the ID Placeholder to be replaced. Subsequent operations in the batched request MAY make use of the ID Placeholder by using a Unique Identifier enumeration value of IDPlaceholder. For operations other than Locate which return more than one Unique Identifier (either as a Unique Identifier, Private Key Unique Identifier, or Public Key Unique Identifier), the first value in the response is the Unique Identifier value with respect to ID Placeholder handling.

Requests MAY contain attribute values to be assigned to the object. This information is specified with zero or more individual attributes.

For any operations that operate on Managed Objects already stored on the server, any archived object SHALL first be made available by a Recover operation before they MAY be specified (i.e., as on-line objects).

Multi-part cryptographic operations (operations where a stream of data is provided across multiple requests from a client to a server) are optionally supported by those cryptographic operations that include the Correlation Value, Init Indicator and Final Indicator request parameters.

For multi-part cryptographic operations the following sequence is performed

1. On the first request
   1. Provide an Init Indicator with a value of True
   2. Provide any other required parameters
   3. Preserve the Correlation Value returned in the response for use in subsequent requests
   4. Use the Data output (if any) from the response
2. On subsequent requests
   1. Provide the Correlation Value from the response to the first request
   2. Provide any other required parameters
   3. Use the next block of Data output (if any) from the response
3. On the final request
   1. Provide the Correlation Value from the response to the first request
   2. Provide a Final Indicator with a value of True
   3. Use the final block of Data output (if any) from the response

Single-part cryptographic operations (operations where a single input is provided and a single response returned) the following sequence is performed:

1. On each request
   1. Do not provide an Init Indicator, Final Indicator or Correlation Value or provide an Init indicator and Final Indicator but no Correlation Value.
   2. Provide any other required parameters
   3. Use the Data output from the response

Data is always required in cryptographic operations except when either Init Indicator or Final Indicator is true.

The list of Result Reasons listed for each Operation is not exhaustive for each Operation. Any Result Reason listed in the Message Data Structures Section MAY be used. A Server SHALL return the most appropriate Result Reason for the given context.

### Activate

This operation requests the server to activate a Managed Object. The operation SHALL only be performed on an object in the Pre-Active state and has the effect of changing its state to Active, and setting its Activation Date to the current date and time.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | Determines the object being activated. |

Table 249: Activate Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |

Table 250: Activate Response Payload

#### Error Handling – Activate

This section details the specific Result Reasons that SHALL be returned for errors detected in a Activate Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Object Type, Object Not Found, Wrong Key Lifecycle State, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 251: Activate Errors

### Add Attribute

This operation requests the server to add a new attribute instance to be associated with a Managed Object and set its value. The request contains the Unique Identifier of the Managed Object to which the attribute pertains, along with the attribute name and value. For single-instance attributes, this creates the attribute value. For multi-instance attributes, this is how the first and subsequent values are created. Existing attribute values SHALL NOT be changed by this operation. Read-Only attributes SHALL NOT be added using the Add Attribute operation.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| New Attribute | Yes | Specifies the attribute to be added to the object. |

Table 252: Add Attribute Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |

Table 253: Add Attribute Response Payload

#### Error Handling - Add Attribute

This section details the specific Result Reasons that SHALL be returned for errors detected in a Add Attribute Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attribute Single Valued, Invalid Attribute, Invalid Message, Non Unique Name Attribute, Object Not Found, Read Only Attribute, Server Limit Exceeded, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large, Wrong Key Lifecycle State |

Table 254: Add Attribute Errors

### Adjust Attribute

This operation requests the server to adjust the value of an attribute. The request contains the Unique Identifier of the Managed Object to which the attribute pertains, along with the attribute reference and value. If the object did not have value for the attribute, the previous value is assumed to be a 0 for numeric types and intervals, or false for Boolean, otherwise an error is raised. If the object had exactly one instance, then it is modified. If it has more than one instance an error is raised. Read-Only attributes SHALL NOT be added or modified using this operation.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| Attribute Reference | Yes | The attribute to be adjusted. |
| Adjustment Type | Yes | The adjustment to be made. |
| Adjustment Value | No | The value for the adjustment |

Table 255: Adjust Attribute Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |

Table 256: Adjust Attribute Response Payload

#### Error Handling - Adjust Attribute

This section details the specific Result Reasons that SHALL be returned for errors detected in a Adjust Attribute Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Data Type, Item Not Found, Multi Valued Attribute, Numeric Range, Object Archived, Read Only Attribute, Unsupported Attribute, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large, Wrong Key Lifecycle State |

Table 257: Adjust Attribute Errors

### Archive

This operation is used to specify that a Managed Object MAY be archived. The actual time when the object is archived, the location of the archive, or level of archive hierarchy is determined by the policies within the key management system and is not specified by the client. The request contains the Unique Identifier of the Managed Object. This request is only an indication from a client that, from its point of view, the key management system MAY archive the object.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |

Table 258: Archive Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |

Table 259: Archive Response Payload

#### Error Handling – Archive

This section details the specific Result Reasons that SHALL be returned for errors detected in a Archive Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Object Archived, Object Not Found, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 260: Archive Errors

### Cancel

This operation requests the server to cancel an outstanding asynchronous operation. The correlation value of the original operation SHALL be specified in the request. The server SHALL respond with a *Cancellation Result.* The response to this operation is not able to be asynchronous.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Asynchronous Correlation Value | Yes | Specifies the request being canceled. |

Table 261: Cancel Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Asynchronous Correlation Value | Yes | Specified in the request. |
| Cancellation Result | Yes | Enumeration indicating the result of the cancellation. |

Table 262: Cancel Response Payload

This section details the specific Result Reasons that SHALL be returned for errors detected in a Cancel Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Asynchronous Correlation Value, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 263: Cancel Errors

### Certify

This request is used to generate a Certificate object for a public key or a Certificate Request. This request supports the certification of a new public key, as well as the certification of a public key that has already been certified (i.e., certificate update). Only a single certificate SHALL be requested at a time.

The Certificate Request Value MAY be omitted, in which case the public key or Certificate Request for which a Certificate object is generated SHALL be specified by its Unique Identifier only. If the Certificate Request Type and the Certificate Request Value objects are omitted from the request and the Unique Identifier does not refer to a Certificate Request, then the Certificate Type SHALL be specified using the Attributes object.

The Certificate Request MAY be passed as a Byte String, which allows multiple certificate request types for X.509 certificates (e.g., PKCS#10, PEM, etc.) to be submitted to the server.

For the public key, the server SHALL create a Certificate Link attribute pointing to the generated certificate. For the generated certificate, the server SHALL create a Public Key Link attribute pointing to the Public Key.

A client MAY indicate which Certification Authority should be used for the Certify operation by providing a X.509 Certificate Issuer value in the list of attributes.

If the information in the Certificate Request Value field conflicts with the attributes specified in the Attributes, then the information in the Certificate Request Value field takes precedence.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | No | The Unique Identifier of the Public Key or the Certificate Request being certified. |
| Certificate Request Type | No | An Enumeration object specifying the type of certificate request. It is REQUIRED if the Certificate Request Value is present. |
| Certificate Request Value | No | A Byte String object with the certificate request. |
| Attributes | No | Specifies desired object attributes. |
| Protection Storage Masks | No | Specifies all permissible Protection Storage Mask selections for the new object |

Table 264: Certify Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the generated Certificate object. |

Table 265: Certify Response Payload

#### Error Handling – Certify

This section details the specific Result Reasons that SHALL be returned for errors detected in a Certify Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid CSR, Invalid Object Type, Item Not Found, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Protection Storage Unavailable, Response Too Large |

Table 266: Certify Errors

### Check

This operation requests that the server check for the use of a Managed Object according to values specified in the request. This operation SHOULD only be used when placed in a batched set of operations, usually following a Locate, Create, Create Pair, Derive Key, Certify, Re-Certify, Re-key or Re-key Key Pair operation, and followed by a Get operation.

If the server determines that the client is allowed to use the object according to the specified attributes, then the server returns the Unique Identifier of the object.

If the server determines that the client is not allowed to use the object according to the specified attributes, then the server empties the ID Placeholder and does not return the Unique Identifier, and the operation returns the set of attributes specified in the request that caused the server policy denial. The only attributes returned are those that resulted in the server determining that the client is not allowed to use the object, thus allowing the client to determine how to proceed.

Only STOP or UNDO Batch Error Continuation Option values SHOULD be used by the client in such a batch. Additional attributes that MAY be specified in the request are limited to:

|  |  |
| --- | --- |
| **Value** | **Description** |
| Usage Limits Count | The request MAY contain the usage amount that the client deems necessary to complete its needed function. This does not require that any subsequent Get Usage Allocation operations request this amount. It only means that the client is ensuring that the amount specified is available. |
| Cryptographic Usage Mask | This is used to specify the cryptographic operations for which the client intends to use the object (see Section 3.19). This allows the server to determine if the policy allows this client to perform these operations with the object. Note that this MAY be a different value from the one specified in a Locate operation that precedes this operation. Locate, for example, MAY specify a Cryptographic Usage Mask requesting a key that MAY be used for both Encryption and Decryption, but the value in the Check operation MAY specify that the client is only using the key for Encryption at this time. |
| Lease Time | This specifies a desired lease time (see Section 3.20). The client MAY use this to determine if the server allows the client to use the object with the specified lease or longer. Including this attribute in the Check operation does not actually cause the server to grant a lease, but only indicates that the requested lease time value MAY be granted if requested by a subsequent, batched Obtain Lease operation. |

Table 267: Check value description

Note that these objects are not encoded in an Attribute structure.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| Usage Limits Count | No | Specifies the number of Usage Limits Units to be protected to be checked against server policy. |
| Cryptographic Usage Mask | No | Specifies the Cryptographic Usage for which the client intends to use the object. |
| Lease Time | No | Specifies a Lease Time value that the Client is asking the server to validate against server policy. |

Table 268: Check Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes, unless a failure, | The Unique Identifier of the object. |
| Usage Limits Count | No | Returned by the Server if the Usage Limits value specified in the Request Payload is larger than the value that the server policy allows. |
| Cryptographic Usage Mask | No | Returned by the Server if the Cryptographic Usage Mask specified in the Request Payload is rejected by the server for policy violation. |
| Lease Time | No | Returned by the Server if the Lease Time value in the Request Payload is larger than a valid Lease Time that the server MAY grant. |

Table 269: Check Response Payload

#### Error Handling – Check

This section details the specific Result Reasons that SHALL be returned for errors detected in a Check Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Illegal Object Type, Incompatible Cryptographic Usage Mask, Object Not Found, Usage Limit Exceeded, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 270: Check Errors

### Create

This operation requests the server to generate a new symmetric key or generate Secret Data as a Managed Cryptographic Object.

The request contains information about the type of object being created, and some of the attributes to be assigned to the object (e.g., Cryptographic Algorithm, Cryptographic Length, etc.).

The response contains the Unique Identifier of the created object.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Object Type | Yes | Determines the type of object to be created. |
| Attributes | Yes | Specifies desired attributes to be associated with the new object. |
| Protection Storage Masks | No | Specifies all permissible Protection Storage Mask selections for the new object |

Table 271: Create Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Object Type | Yes | Type of object created. |
| Unique Identifier | Yes | The Unique Identifier of the newly created object. |

Table 272: Create Response Payload

#### Error Handling - Create

This section details the specific Result Reasons that SHALL be returned for errors detected in a Create operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attribute Read Only, Attribute Single Valued, Cryptographic Failure, Invalid Attribute, Invalid Attribute Value, Invalid Object Type, Non Unique Name Attribute, Read Only Attribute, Server Limit Exceeded, Unsupported Attribute, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Protection Storage Unavailable, Response Too Large |

Table 273: Create Errors

### Create Credential

This operation requests the server to create a credential.

The request contains information about the attributes to be assigned to the object.

The response contains the Unique Identifier of the created object.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Credential Type | Yes | Determines the type of credential object to be created. |
| Attributes | Yes | Specifies desired attributes to be associated with the new object. |
| Password Credential or Device Credential or Hashed Password Credential or OTP Credential or Certificate |  |  |

Table 274: Create Credential Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the newly created object. |

Table 275: Create Credential Response Payload

#### Error Handling - Create Group

This section details the specific Result Reasons that SHALL be returned for errors detected in a Create Credential Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attribute Read Only, Attribute Single Valued, Invalid Attribute, Invalid Attribute Value, Non Unique Name Attribute, Read Only Attribute, Server Limit Exceeded, Unsupported Attribute, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 276: Create Credential Errors

### Create Group

This operation requests the server to create a new group.

The request contains information about the attributes to be assigned to the object.

The response contains the Unique Identifier of the created object.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Attributes | Yes | Specifies desired attributes to be associated with the new object. |

Table 277: Create Group Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the newly created object. |

Table 278: Create Group Response Payload

#### Error Handling - Create Group

This section details the specific Result Reasons that SHALL be returned for errors detected in a Create Group Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attribute Read Only, Attribute Single Valued, Invalid Attribute, Invalid Attribute Value, Non Unique Name Attribute, Read Only Attribute, Server Limit Exceeded, Unsupported Attribute, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 279: Create Group Errors

### Create Key Pair

This operation requests the server to generate a new public/private key pair and register the two corresponding new Managed Cryptographic Objects.

The request contains attributes to be assigned to the objects (e.g., Cryptographic Algorithm, Cryptographic Length, etc.). Attributes MAY be specified for both keys at the same time by specifying a Common Attributes object in the request. Attributes not common to both keys (e.g., Name, Cryptographic Usage Mask) MAY be specified using the Private Key Attributes and Public Key Attributes objects in the request, which take precedence over the Common Attributes object.

For the Private Key, the server SHALL create a Public Key Link attribute pointing to the Public Key. For the Public Key, the server SHALL create a Private Key Link attribute pointing to the Private Key. The response contains the Unique Identifiers of both created objects.

|  |  |  |
| --- | --- | --- |
| **Request Payload** | | |
| Item | REQUIRED | Description |
| Common Attributes | No | Specifies desired attributes to be associated with the new object that apply to both the Private and Public Key Objects. |
| Private Key Attributes | No | Specifies the attributes to be associated with the new object that apply to the Private Key Object. |
| Public Key Attributes | No | Specifies the attributes to be associated with the new object that apply to the Public Key Object. |
| Common Protection Storage Masks | No | Specifies all Protection  Storage Mask selections that are  permissible for the new Private Key  and Public Key objects. |
| Private Protection Storage Masks | No | Specifies all Protection  Storage Mask selections that are  permissible for the new Private Key  object. |
| Public Protection Storage Masks | No | Specifies all Protection  Storage Mask selections that are  permissible for the new Public Key  object. |

Table 280: Create Key Pair Request Payload

For multi-instance attributes, the union of the values found in the attributes of the Common, Private, and Public Key Attributes SHALL be used.

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Private Key Unique Identifier | Yes | The Unique Identifier of the newly created Private Key object. |
| Public Key Unique Identifier | Yes | The Unique Identifier of the newly created Public Key object. |

Table 281: Create Key Pair Response Payload

Table 282 indicates which attributes SHALL have the same value for the Private and Public Key.

|  |  |
| --- | --- |
| Attribute | SHALL contain the same value for both Private and Public Key |
| Cryptographic Algorithm | Yes |
| Cryptographic Length | Yes |
| Cryptographic Domain Parameters | Yes |
| Cryptographic Parameters | Yes |

Table 282: Create Key Pair Attribute Requirements

Setting the same Cryptographic Length value for both private and public key does not imply that both keys are of equal length. For RSA, Cryptographic Length corresponds to the bit length of the Modulus. For DSA and DH algorithms, Cryptographic Length corresponds to the bit length of parameter P, and the bit length of Q is set separately in the Cryptographic Domain Parameters attribute. For ECDSA, ECDH, and ECMQV algorithms, Cryptographic Length corresponds to the bit length of parameter Q.

#### Error Handling - Create Key Pair

This section details the specific Result Reasons that SHALL be returned for errors detected in a Create Key Pair Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attribute Read Only, Attribute Single Valued, Cryptographic Failure, Invalid Attribute, Invalid Attribute Value, Non Unique Name Attribute, Server Limit Exceeded, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Private Protection Storage Unavailable, Public Protection Storage Unavailable, Response Too Large |

Table 283: Create Key Pair Errors

### Create Split Key

This operation requests the server to generate a new split key and register all the splits as individual new Managed Cryptographic Objects.

The request contains attributes to be assigned to the objects (e.g., Split Key Parts, Split Key Threshold, Split Key Method, Cryptographic Algorithm, Cryptographic Length, etc.). The request MAY contain the Unique Identifier of an existing cryptographic object that the client requests be split by the server. If the attributes supplied in the request do not match those of the key supplied, the attributes of the key take precedence.

The response contains the Unique Identifiers of all created objects.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Object Type | Yes | Determines the type of object to be created. |
| Unique Identifier | No | The Unique Identifier of the key to be split (if applicable). |
| Split Key Parts | Yes | The total number of parts. |
| Split Key Threshold | Yes | The minimum number of parts needed to reconstruct the entire key. |
| Split Key Method | Yes |  |
| Prime Field Size | No |  |
| Attributes | Yes | Specifies desired object attributes. |
| Protection Storage Masks | No | Specifies all permissible Protection Storage Mask selections for the new object |

Table 284: Create Split Key Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes, MAY be repeated | The list of Unique Identifiers of the newly created objects. |

Table 285: Create Split Key Response Payload

#### Error Handling - Create Split Key

This section details the specific Result Reasons that SHALL be returned for errors detected in a Create Split Key Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Bad Cryptographic Parameters, Cryptographic Failure, Invalid Attribute, Invalid Attribute Value, Invalid Object type, Item Not Found, Non Unique Name Attribute, Server Limit Exceeded, Unsupported Cryptographic Parameters, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Protection Storage Unavailable, Response Too Large |

Table 286: Create Split Key Errors

### Create User

This operation requests the server to create a new user.

The request contains information about the attributes to be assigned to the object.

The response contains the Unique Identifier of the created object.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Attributes | Yes | Specifies desired attributes to be associated with the new object. |

Table 287: Create User Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the newly created object. |

Table 288: Create User Response Payload

#### Error Handling - Create Group

This section details the specific Result Reasons that SHALL be returned for errors detected in a Create User Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attribute Read Only, Attribute Single Valued, Invalid Attribute, Invalid Attribute Value, Non Unique Name Attribute, Read Only Attribute, Server Limit Exceeded, Unsupported Attribute, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 289: Create User Errors

### Deactivate

This operation requests the server to *Deactivate* a Managed Object. The request contains an optional reason for the deactivation. The Deactivate operation places an Object into a “deactivated” state, and the Deactivation Date is set to the current date and time. If the Deactivation Reason is not provided, then a server SHALL process the request as though the Deactivation Reason was specified as Unspecified.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object.. |
| Deactivation Reason | No | Specifies the reason for deactivation. |
| Deactivation Date | No | Specifies the date on which the deactivation occurred. |

Table 290: Deactivate Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |

Table 291: Deactivate Response Payload

#### Error Handling – Deactivate

This section details the specific Result Reasons that SHALL be returned for errors detected in a Revoke Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Field, Invalid Object Type, Object Not Found, Wrong Key Lifecycle State, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 292: Deactivate Errors

### Decrypt

This operation requests the server to perform a decryption operation on the provided data using a Managed Cryptographic Object as the key for the decryption operation.

The request contains information about the cryptographic parameters (mode and padding method), the data to be decrypted, and the IV/Counter/Nonce to use. The cryptographic parameters MAY be omitted from the request as they can be specified as associated attributes of the Managed Cryptographic Object. The initialization vector/counter/nonce MAY also be omitted from the request if the algorithm does not use an IV/Counter/Nonce.

The response contains the Unique Identifier of the Managed Cryptographic Object used as the key and the result of the decryption operation.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the Managed Cryptographic Object that is the key to use for the decryption operation. |
| Cryptographic Parameters | No | The Cryptographic Parameters (Block Cipher Mode, Padding Method) corresponding to the particular decryption method requested.  If there are no Cryptographic Parameters associated with the Managed Cryptographic Object and the algorithm requires parameters then the operation SHALL return with a Result Status of Operation Failed. |
| Data | Yes for single-part. No for multi-part. | The data to be decrypted. |
| IV/Counter/Nonce | No | The initialization vector, counter or nonce to be used (where appropriate). |
| Correlation Value | No | Specifies the existing stream or by-parts cryptographic operation (as returned from a previous call to this operation). |
| Init Indicator | No | Initial operation as Boolean |
| Final Indicator | No | Final operation as Boolean |
| Authenticated Encryption Additional Data | No | Additional data to be authenticated via the Authenticated Encryption Tag. If supplied in multi-part decryption, this data MUST be supplied on the initial Decrypt request |
| Authenticated Encryption Tag | No | Specifies the tag that will be needed to authenticate the decrypted data and the additional authenticated data. If supplied in multi-part decryption, this data MUST be supplied on the initial Decrypt request |

Table 293: Decrypt Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the Managed Cryptographic Object that is the key used for the decryption operation. |
| Data | No. | The decrypted data (as a Byte String). |
| Correlation Value | No | Specifies the stream or by-parts value to be provided in subsequent calls to this operation for performing cryptographic operations. |

Table 294: Decrypt Response Payload

#### Error Handling - Decrypt

This section details the specific Result Reasons that SHALL be returned for errors detected in a Decrypt Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Bad Cryptographic Parameters, Cryptographic Failure, Cryptographic Failure, Incompatible Cryptographic Usage Mask, Invalid Correlation Value, Invalid Object Type, Key Value Not Present, Missing Initialization Vector, Object Not Found, Unsupported Cryptographic Parameters, Wrong Key Lifecycle State, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 295: Decrypt Errors

### Delegated Login

This operation requests the server to allow future requests to be authenticated using Ticket data that is returned by this operation. Requests using the ticket MUST only be permitted to perform the operations specified in the Rights section.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Lease Time | No | The lease time Interval or Date Time for the ticket. |
| Request Count | No | The integer count of the number of requests that can be made with the ticket |
| Usage Limits | No | The usage limits for operations performed. |
| Rights | Yes | List of Rights granted to the ticket holder which may only perform operations allowed by at least one of the contained Right structures. |

Table 296: Delegated Login Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Ticket | Yes | The Ticket that is returned |

Table 297: Delegated Login Response Payload

#### Error Handling – Delegated Login

This section details the specific Result Reasons that SHALL be returned for errors detected in a Delegated Login Operation

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Field, Permission Denied, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 298: Delegated Login Errors

### Delete Attribute

This operation requests the server to delete an attribute associated with a Managed Object. The request contains the Unique Identifier of the Managed Object whose attribute is to be deleted, the Current Attribute of the attribute. Attributes that are always REQUIRED to have a value SHALL never be deleted by this operation. Attempting to delete a non-existent attribute or specifying an Current Attribute for which there exists no attribute value SHALL result in an error. If no Current Attribute is specified in the request, and an Attribute Reference is specified, then all instances of the specified attribute SHALL be deleted.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | Determines the object whose attributes are being deleted. |
| Current Attribute | No | Specifies the attribute associated with the object to be deleted. |
| Attribute Reference | No | Specifies the reference for the attribute associated with the object to be deleted. |

Table 299: Delete Attribute Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |

Table 300: Delete Attribute Response Payload

#### Error Handling - Delete Attribute

This section details the specific Result Reasons that SHALL be returned for errors detected in a Delete Attribute Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attribute Instance Not Found, Attribute Not Found, Attribute Read Only, Invalid Attribute, Object Not Found, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large, Wrong Key Lifecycle State |

Table 301: Delete Attribute Errors

### Derive Key

This request is used to derive a Symmetric Key or Secret Data object from keys or Secret Data objects that are already known to the key management system. The request SHALL only apply to Managed Objects that have the Derive Key bit set in the Cryptographic Usage Mask attribute of the specified Managed Object (i.e., are able to be used for key derivation). If the operation is issued for an object that does not have this bit set, then the server SHALL return an error. For all derivation methods, the client SHALL specify the desired length of the derived key or Secret Data object using the Cryptographic Length attribute. If a key is created, then the client SHALL specify both its Cryptographic Length and Cryptographic Algorithm. If the specified length exceeds the output of the derivation method, then the server SHALL return an error. Clients MAY derive multiple keys and IVs by requesting the creation of a Secret Data object and specifying a Cryptographic Length that is the total length of the derived object. If the specified length exceeds the output of the derivation method, then the server SHALL return an error.

The fields in the Derive Key request specify the Unique Identifiers of the keys or Secret Data objects to be used for derivation (e.g., some derivation methods MAY use multiple keys or Secret Data objects to derive the result), the method to be used to perform the derivation, and any parameters needed by the specified method.

The server SHALL perform the derivation function, and then register the derived object as a new Managed Object, returning the new Unique Identifier for the new object in the response

For the keys or Secret Data objects from which the key or Secret Data object is derived, the server SHALL create a Derived Object Link attribute pointing to the Symmetric Key or Secret Data object derived as a result of this operation. For the Symmetric Key or Secret Data object derived as a result of this operation, the server SHALL create a Derivation Base Object Link attribute pointing to the keys or Secret Data objects from which the key or Secret Data object is derived.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Object Type | Yes | Determines the type of object to be created. |
| Unique Identifier | Yes. MAY be repeated | Determines the object or objects to be used to derive a new key. |
| Derivation Method | Yes | An Enumeration object specifying the method to be used to derive the new key. |
| Derivation Parameters | Yes | A Structure object containing the parameters needed by the specified derivation method. |
| Attributes | Yes | Specifies desired attributes to be associated with the new object; the length and algorithm SHALL always be specified for the creation of a symmetric key. |

Table 302: Derive Key Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the newly derived key or Secret Data object. |

Table 303: Derive Key Response Payload

#### Error Handling - Derive Key

This section details the specific Result Reasons that SHALL be returned for errors detected in a Derive Key Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Bad Cryptographic parameters, Cryptographic Failure, Incompatible Cryptographic Usage Mask, Invalid Attribute, Invalid Field, Invalid Message, Invalid Object Type, Key Value Not Present, Non Unique Name Attribute, Object Not Found, Unsupported Cryptographic Parameters, Wrong Key Lifecycle State, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 304: Derive Key Errors

### Destroy

This operation is used to indicate to the server that the key material for the specified Managed Object SHALL be destroyed or rendered inaccessible. The meta-data for the key material SHALL be retained by the server. Objects SHALL only be destroyed if they are in either Pre-Active or Deactivated state.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | Determines the object being destroyed. |

Table 305: Destroy Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |

Table 306: Destroy Response Payload

#### Error Handling – Destroy

This section details the specific Result Reasons that SHALL be returned for errors detected in a Destroy Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Object Destroyed, Object Not Found, Wrong Key Lifecycle State, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 307: Destroy Errors

### Discover Versions

This operation is used by the client to determine a list of protocol versions that is supported by the server. The request payload contains an OPTIONAL list of protocol versions that is supported by the client. The protocol versions SHALL be ranked in decreasing order of preference.

The response payload contains a list of protocol versions that are supported by the server. The protocol versions are ranked in decreasing order of preference. If the client provides the server with a list of supported protocol versions in the request payload, the server SHALL return only the protocol versions that are supported by both the client and server. The server SHOULD list all the protocol versions supported by both client and server. If the protocol version specified in the request header is not specified in the request payload and the server does not support any protocol version specified in the request payload, the server SHALL return an empty list in the response payload. If no protocol versions are specified in the request payload, the server SHOULD return all the protocol versions that are supported by the server.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Protocol Version | No, MAY be Repeated | The list of protocol versions supported by the client ordered in decreasing order of preference. |

Table 308: Discover Versions Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Protocol Version | No, MAY be repeated | The list of protocol versions supported by the server ordered in decreasing order of preference. |

Table 309: Discover Versions Response Payload

#### Error Handling - Discover Versions

This section details the specific Result Reasons that SHALL be returned for errors detected in a Discover Versions Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 310: Discover Versions Errors

### Encrypt

This operation requests the server to perform an encryption operation on the provided data using a Managed Cryptographic Object as the key for the encryption operation.

The request contains information about the cryptographic parameters (mode and padding method), the data to be encrypted, and the IV/Counter/Nonce to use. The cryptographic parameters MAY be omitted from the request as they can be specified as associated attributes of the Managed Cryptographic Object. The IV/Counter/Nonce MAY also be omitted from the request if the cryptographic parameters indicate that the server shall generate a Random IV on behalf of the client or the encryption algorithm does not need an IV/Counter/Nonce. The server does not store or otherwise manage the IV/Counter/Nonce.

If the Managed Cryptographic Object referenced has a Usage Limits attribute then the server SHALL obtain an allocation from the current Usage Limits value prior to performing the encryption operation. If the allocation is unable to be obtained the operation SHALL return with a result status of Operation Failed and result reason of Permission Denied.

The response contains the Unique Identifier of the Managed Cryptographic Object used as the key and the result of the encryption operation.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the Managed Cryptographic Object that is the key to use for the encryption operation. |
| Cryptographic Parameters | No | The Cryptographic Parameters (Block Cipher Mode, Padding Method, RandomIV) corresponding to the particular encryption method requested.  If there are no Cryptographic Parameters associated with the Managed Cryptographic Object and the algorithm requires parameters then the operation SHALL return with a Result Status of Operation Failed. |
| Data | Yes for single-part. No for multi-part. | The data to be. |
| IV/Counter/Nonce | No | The initialization vector, counter or nonce to be used (where appropriate). |
| Correlation Value | No | Specifies the existing stream or by-parts cryptographic operation (as returned from a previous call to this operation). |
| Init Indicator | No | Initial operation as Boolean |
| Final Indicator | No | Final operation as Boolean |
| Authenticated Encryption Additional Data | No | Any additional data to be authenticated via the Authenticated Encryption Tag. If supplied in multi-part encryption, this data MUST be supplied on the initial Encrypt request |

Table 311: Encrypt Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the Managed Cryptographic Object that was the key used for the encryption operation. |
| Data | No. | The encrypted data (as a Byte String). |
| IV/Counter/Nonce | No | The value used if the Cryptographic Parameters specified Random IV and the IV/Counter/Nonce value was not provided in the request and the algorithm requires the provision of an IV/Counter/Nonce. |
| Correlation Value | No | Specifies the stream or by-parts value to be provided in subsequent calls to this operation for performing cryptographic operations. |
| Authenticated Encryption Tag | No | Specifies the tag that will be needed to authenticate the decrypted data (and any “additional data”). Only returned on completion of the encryption of the last of the plaintext by an authenticated encryption cipher. |

Table 312: Encrypt Response Payload

#### Error Handling – Encrypt

This section details the specific Result Reasons that SHALL be returned for errors detected in a Encrypt Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Bad Cryptographic Parameters, Cryptographic Failure, Incompatible Cryptographic Usage Mask, Invalid Correlation Value, Invalid Object Type, Key Value Not Present, Missing Initialization Vector, Object Not Found, Unsupported Cryptographic Parameters, Usage Limit Exceeded, Wrong Key Lifecycle State, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 313: Encrypt Errors

### Export

This operation requests that the server returns a Managed Object specified by its Unique Identifier, together with its attributes.

The Key Format Type, Key Wrap Type, Key Compression Type and Key Wrapping Specification SHALL have the same semantics as for the Get operation. If the Managed Object has been Destroyed then the key material for the specified managed object SHALL not be returned in the response.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| Key Format Type | No | Determines the key format type to be returned. |
| Key Wrap Type | No | Determines the Key Wrap Type of the returned key value. |
| Key Compression Type | No | Determines the compression method for elliptic curve public keys. |
| Key Wrapping Specification | No | Specifies keys and other information for wrapping the returned object. |

Table 314: Export Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Object Type | Yes | Type of object |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| Attributes | Yes | All of the object’s Attributes. |
| Any Object (Section 2) | Yes | The object value being returned, in the same manner as the Get operation. |

Table 315: Export Response Payload

#### Error Handling – Export

This section details the specific Result Reasons that SHALL be returned for errors detected in a Export Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Bad Cryptographic Parameters, Encoding Option Error, Encoding Option Error, Incompatible Cryptographic Usage Mask, Invalid Object Type, Key Compression Type Not Supported, Key Format Type Not Supported, Key Value Not Present, Key Wrap Type Not Supported, Object Not Found, Wrapping Object Archived, Wrapping Object Destroyed, Wrapping Object Not Found, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 316: Export Errors

### Get

This operation requests that the server returns the Managed Object specified by its Unique Identifier.

Only a single object is returned. The response contains the Unique Identifier of the object, along with the object itself, which MAY be wrapped using a wrapping key as specified in the request.

The following key format capabilities SHALL be assumed by the client; restrictions apply when the client requests the server to return an object in a particular format:

* If a client registered a key in a given format, the server SHALL be able to return the key during the Get operation in the same format that was used when the key was registered.
* Any other format conversion MAY be supported by the server.

If Key Format Type is specified to be PKCS#12 then the response payload shall be a PKCS#12 container as specified by [RFC7292]. The Unique Identifier shall be either that of a private key or certificate to be included in the response. The container shall be protected using the Secret Data object specified via the private key or certificate’s PKCS#12 Password Link. The current certificate chain shall also be included as determined by using the private key’s Public Key link to get the corresponding public key (where relevant), and then using that public key’s PKCS#12 Certificate Link to get the base certificate, and then using each certificate’s Certificate Link to build the certificate chain. It is an error if there is more than one valid certificate chain.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| Key Format Type | No | Determines the key format type to be returned. |
| Key Wrap Type | No | Determines the Key Wrap Type of the returned key value. |
| Key Compression Type | No | Determines the compression method for elliptic curve public keys. |
| Key Wrapping Specification | No | Specifies keys and other information for wrapping the returned object. |

Table 317: Get Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Object Type | Yes | Type of object. |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| Any Object (Section 2) | Yes | The object being returned. |

Table 318: Get Response Payload

#### Error Handling – Get

This section details the specific Result Reasons that SHALL be returned for errors detected in a Get Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Bad Cryptographic Parameters, Encoding Option Error, Encoding Option Error, Incompatible Cryptographic Usage Mask, Incompatible Cryptographic Usage Mask, Invalid Object Type, Key Compression Type Not Supported, Key Format Type Not Supported, Key Value Not Present, Key Wrap Type Not Supported, Not Extractable, Object Not Found, Sensitive, Wrapping Object Archived, Wrapping Object Destroyed, Wrapping Object Not Found, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 319: Get Errors

### Get Attributes

This operation requests one or more attributes associated with a Managed Object. The object is specified by its Unique Identifier, and the attributes are specified by their name in the request. If a specified attribute has multiple instances, then all instances are returned. If a specified attribute does not exist (i.e., has no value), then it SHALL NOT be present in the returned response. If none of the requested attributes exist, then the response SHALL consist only of the Unique Identifier. The same Attribute Reference SHALL NOT be present more than once in a request.

If no Attribute Reference is provided, the server SHALL return all attributes.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| Attribute Reference | No, MAY be repeated | Specifies an attribute associated with the object. |

Table 320: Get Attributes Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| Attributes | Yes | The requested attributes associated with the object. |

Table 321: Get Attributes Response Payload

#### Error Handling - Get Attributes

This section details the specific Result Reasons that SHALL be returned for errors detected in a Get Attributes Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Attribute, Object Not Found, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 322: Get Attributes Errors

### Get Attribute List

This operation requests a list of the attribute names associated with a Managed Object. The object is specified by its Unique Identifier.

If no Attribute Reference is provided, the server SHALL return all attributes.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |

Table 323: Get Attribute List Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| Attribute Reference | Yes, MAY be repeated | The attributes associated with the object. |

Table 324: Get Attribute List Response Payload

#### Error Handling - Get Attribute List

This section details the specific Result Reasons that SHALL be returned for errors detected in a Get Attribute List Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Object Not Found, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 325: Get Attribute List Errors

### Get Constraints

This operation instructs the server to return the constraints that are being applied to Managed Objects during operations.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
|  |  |  |

Table 326: Get Constraints Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Constraints | Yes | The set of Constraints that are being  applied during operations. |

Table 327: Get Constraints Response Payload

#### Error Handling - Get Constraints

This section details the specific Result Reasons that SHALL be returned for errors detected in a Get Constrains Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Field, Invalid Object Type, Attestation Failed,  Attestation Required, Feature Not Supported, Invalid  Field, Invalid Message, Operation Not Supported,  Permission Denied, Response Too Large |

Table 328: Get Constraints Errors

### Get Usage Allocation

This operation requests the server to obtain an allocation from the current Usage Limits value to allow the client to use the Managed Cryptographic Object for applying cryptographic protection. The allocation only applies to Managed Objects that are able to be used for applying protection (e.g., symmetric keys for encryption, private keys for signing, etc.) and is only valid if the Managed Object has a Usage Limits attribute. Usage for processing cryptographically protected information (e.g., decryption, verification, etc.) is not limited and is not able to be allocated. A Managed Object that has a Usage Limits attribute SHALL NOT be used by a client for applying cryptographic protection unless an allocation has been obtained using this operation. The operation SHALL only be requested during the time that protection is enabled for these objects (i.e., after the Activation Date and before the Protect Stop Date). If the operation is requested for an object that has no Usage Limits attribute, or is not an object that MAY be used for applying cryptographic protection, then the server SHALL return an error.

The field in the request specifies the number of units that the client needs to protect. If the requested amount is not available or if the Managed Object is not able to be used for applying cryptographic protection at this time, then the server SHALL return an error. The server SHALL assume that the entire allocated amount is going to be consumed. Once the entire allocated amount has been consumed, the client SHALL NOT continue to use the Managed Object for applying cryptographic protection until a new allocation is obtained.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| Usage Limits Count | Yes | The number of Usage Limits Units to be protected. |

Table 329: Get Usage Allocation Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |

Table 330: Get Usage Allocation Response Payload

#### Error Handling - Get Usage Allocation

This section details the specific Result Reasons that SHALL be returned for errors detected in a Get Usage Allocation Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attribute Not Found, Invalid Message, Invalid Object Type, Object Not Found, Usage Limit Exceeded, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 331: Get Usage Allocation Errors

### Hash

This operation requests the server to perform a hash operation on the data provided.

The request contains information about the cryptographic parameters (hash algorithm) and the data to be hashed.

The response contains the result of the hash operation.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Cryptographic Parameters | Yes | The Cryptographic Parameters (Hashing Algorithm) corresponding to the particular hash method requested. |
| Data | Yes for single-part. No for multi-part. | The data to be hashed . |
| Correlation Value | No | Specifies the existing stream or by-parts cryptographic operation (as returned from a previous call to this operation). |
| Init Indicator | No | Initial operation as Boolean |
| Final Indicator | No | Final operation as Boolean |

Table 332: Hash Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Data | Yes for single-part. No for multi-part. | The hashed data (as a Byte String). |
| Correlation Value | No | Specifies the stream or by-parts value to be provided in subsequent calls to this operation for performing cryptographic operations. |

Table 333: Hash Response Payload

#### Error Handling - HASH

This section details the specific Result Reasons that SHALL be returned for errors detected in a Hash Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Cryptographic Failure, Invalid Correlation Value, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 334: HASH Errors

### Import

This operation requests the server to Import a Managed Object specified by its Unique Identifier. The request specifies the object being imported and all the attributes to be assigned to the object. The attribute rules for each attribute for “Initially set by” and “When implicitly set” SHALL NOT be enforced as all attributes MUST be set to the supplied values rather than any server generated values.

The response contains the Unique Identifier provided in the request or assigned by the server.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object to be imported |
| Object Type | Yes | Determines the type of object being imported. |
| Replace Existing | No | A Boolean. If specified and true then any existing object with the same Unique Identifier SHALL be replaced by this operation. If absent or false and an object exists with the same Unique Identifier then an error SHALL be returned. |
| Key Wrap Type | If and only if the key object is wrapped. | If Not Wrapped then the server SHALL unwrap the object before storing it, and return an error if the wrapping key is  not available. Otherwise the server  SHALL store the object as provided. |
| Attributes | Yes | Specifies object attributes to be associated with the new object. |
| Any Object (Section 2) | Yes | The object being imported. The object and attributes MAY be wrapped. |

Table 335: Import Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the newly imported object. |

Table 336: Import Response Payload

#### Error Handling – Import

This section details the specific Result Reasons that SHALL be returned for errors detected in a Import Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attribute Read Only, Attribute Single Valued, Encoding Option Error, Invalid Attribute, Invalid Attribute Value, Invalid Field, Non Unique Name Attribute, Object Already Exists, Server Limit Exceeded, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Protection Storage Unavailable, Response Too Large |

Table 337: Import Errors

### Interop

This operation informs the server about the status if interop tests. It SHALL NOT be available in a production server. The Interop Operation uses three Interop Functions (Begin, End and Reset).

An Interop Identifier of “\*” is reserved for use during interoperability testing to indicate that the server should perform a cleanup for the currently authenticated user so that testing may be repeated. This allows for repeated testing without manual intervention.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Interop Function | Yes | The function to be performed |
| Interop Identifier | Yes | The identifier if the test case to be submitted as a TextString |

Table 338: Interop Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
|  |  |  |

Table 339: Interop Response Payload

#### Error Handling – Interop

This section details the specific Result Reasons that SHALL be returned for errors detected in an Interop Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Field, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 340: Interop Errors

### Join Split Key

This operation requests the server to combine a list of Split Keys into a single Managed Cryptographic Object. The number of Unique Identifiers in the request SHALL be at least the value of the Split Key Threshold defined in the Split Keys.

The request contains the Object Type of the Managed Cryptographic Object that the client requests the Split Key Objects be combined to form. If the Object Type formed is Secret Data, the client MAY include the Secret Data Type in the request.

The response contains the Unique Identifier of the object obtained by combining the Split Keys.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Object Type | Yes | Determines the type of object to be created. |
| Unique Identifier | Yes, MAY be repeated | Determines the Split Keys to be combined to form the object returned by the server. The minimum number of identifiers is specified by the Split Key Threshold field in each of the Split Keys. |
| Secret Data Type | No | Determines which Secret Data type the Split Keys form. |
| Attributes | No | Specifies desired object attributes. |
| Protection Storage Masks | No | Specifies all permissible Protection Storage Mask selections for the new object |

Table 341: Join Split Key Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object obtained by combining the Split Keys. |

Table 342: Join Split Key Response Payload

#### Error Handling - Join Split Key

This section details the specific Result Reasons that SHALL be returned for errors detected in a Join Split Key Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attribute Read Only, Attribute Single Valued, Bad Cryptographic Parameters, Cryptographic Failure, Cryptographic Failure, Invalid Attribute, Invalid Attribute Value, Invalid Object Type, Non Unique Name Attribute, Object Not Found, Server Limit Exceeded, Unsupported Cryptographic Parameters, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Protection Storage Unavailable, Response Too Large |

Table 343: Join Split Key Errors

### Locate

This operation requests that the server search for one or more Managed Objects, depending on the attributes specified in the request. All attributes are allowed to be used. The request MAY contain a *Maximum Items* field, which specifies the maximum number of objects to be returned. If the Maximum Items field is omitted, then the server MAY return all objects matched, or MAY impose an internal maximum limit due to resource limitations.

The request MAY contain an *Offset Items* field, which specifies the number of objects to skip that satisfy the identification criteria specified in the request. An *Offset Items* field of 0 is the same as omitting the *Offset Items* field. If both *Offset Items* and *Maximum Items* are specified in the request, the server skips *Offset Items* objects and returns up to *Maximum Items* objects.

If more than one object satisfies the identification criteria specified in the request, then the response MAY contain Unique Identifiers for multiple Managed Objects. Responses containing Unique Identifiers for multiple objects SHALL be returned in descending order of object creation (most recently created object first). Returned objects SHALL match all of the attributes in the request. If no objects match, then an empty response payload is returned. If no attribute is specified in the request, any object SHALL be deemed to match the Locate request. The response MAY include *Located Items* which is the count of all objects that satisfy the identification criteria.

The server returns a list of Unique Identifiers of the found objects, which then MAY be retrieved using the Get operation. If the objects are archived, then the Recover and Get operations are REQUIRED to be used to obtain those objects. If a single Unique Identifier is returned to the client, then the server SHALL copy the Unique Identifier returned by this operation into the ID Placeholder variable. If the Locate operation matches more than one object, and the Maximum Items value is omitted in the request, or is set to a value larger than one, then the server SHALL empty the ID Placeholder, causing any subsequent operations that are batched with the Locate, and which do not specify a Unique Identifier explicitly, to fail. This ensures that these batched operations SHALL proceed only if a single object is returned by Locate.

The Date attributes in the Locate request (e.g., Initial Date, Activation Date, etc.) are used to specify a time or a time range for the search. If a single instance of a given Date attribute is used in the request (e.g., the Activation Date), then objects with the same Date attribute are considered to be matching candidate objects. If two instances of the same Date attribute are used (i.e., with two different values specifying a range), then objects for which the Date attribute is inside or at a limit of the range are considered to be matching candidate objects. If a Date attribute is set to its largest possible value, then it is equivalent to an undefined attribute.

When the Cryptographic Usage Mask attribute is specified in the request, candidate objects are compared against this field via an operation that consists of a logical AND of the requested mask with the mask in the candidate object, and then a comparison of the resulting value with the requested mask. For example, if the request contains a mask value of 10001100010000, and a candidate object mask contains 10000100010000, then the logical AND of the two masks is 10000100010000, which is compared against the mask value in the request (10001100010000) and the match fails. This means that a matching candidate object has all of the bits set in its mask that are set in the requested mask, but MAY have additional bits set.

When the Usage Limits attribute is specified in the request, matching candidate objects SHALL have a Usage Limits Count and Usage Limits Total equal to or larger than the values specified in the request.

The Storage Status Mask field is used to indicate whether on-line objects (not archived or destroyed), archived objects, destroyed objects or any combination of the above are to be searched. The server SHALL NOT return unique identifiers for objects that are destroyed unless the Storage Status Mask field includes the Destroyed Storage indicator. The server SHALL NOT return unique identifiers for objects that are archived unless the Storage Status Mask field includes the Archived Storage indicator.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Maximum Items | No | An Integer object that indicates the maximum number of object identifiers the server MAY return. |
| Offset Items | No | An Integer object that indicates the number of object identifiers to skip that satisfy the identification criteria specified in the request. |
| Storage Status Mask | No | An Integer object (used as a bit mask) that indicates whether only on-line objects, only archived objects, destroyed objects or any combination of these, are to be searched. If omitted, then only on-line objects SHALL be returned. |
| Object Class Mask | No | An Integer object (used as a bit mask) that indicates which classes of objects are to be searched. If omitted, then only User Objects SHALL be returned. |
| Attributes | Yes | Specifies an attribute and its value(s) that are REQUIRED to match those in a candidate object (according to the matching rules defined above).  Note: the Attributes structure MAY be empty indicating all objects should match. |

Table 344: Locate Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Located Items | No | An Integer object that indicates the number of object identifiers that satisfy the identification criteria specified in the request. A server MAY elect to omit this value from the Response if it is unable or unwilling to determine the total count of matched items.  A server MAY elect to return the Located Items value even if Offset Items is not present in the Request. |
| Unique Identifier | No, MAY be repeated | The Unique Identifier of the located objects. |

Table 345: Locate Response Payload

#### Error Handling – Locate

This section details the specific Result Reasons that SHALL be returned for errors detected in a Locate Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Attribute, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 346: Locate Errors

### Log

This operation requests the server to log a message to the server log. The response payload returned is empty.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Log Message | Yes | The message to log |

Table 347: Log Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |

Table 348: Log Response Payload

#### Error Handling – Log

This section details the specific Result Reasons that SHALL be returned for errors detected in a Query Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 349: Log Errors

### Login

This operation requests the server to allow future requests ti be authenticated using a ticket that is returned by this operation.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Lease Time | No | The lease time Interval or Date Time for the ticket |
| Request Count | No | The integer count of the number of requests that can be made with the ticket |
| Usage Limits | No | The usage limits for the operations performed |

Table 350: Login Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Ticket | Yes | The ticket that is returned |

Table 351: Login Response Payload

#### Error Handling - Login

This section details the specific Result Reasons that SHALL be returned for errors detected in an Login Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Field, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 352: Login Errors

### Logout

This operation requests the server to terminate the Login and prevent future unauthenticated sessions being created without the ticket.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Ticket | Yes | The ticket to be invalidated |

Table 353: Logout Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |

Table 354: Logout Response Payload

#### Error Handling - Logout

This section details the specific Result Reasons that SHALL be returned for errors detected in an Logout Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Ticket, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 355: Logout Errors

### MAC

This operation requests the server to perform message authentication code (MAC) operation on the provided data using a Managed Cryptographic Object as the key for the MAC operation.

The request contains information about the cryptographic parameters (cryptographic algorithm) and the data to be MACed. The cryptographic parameters MAY be omitted from the request as they can be specified as associated attributes of the Managed Cryptographic Object.

The response contains the Unique Identifier of the Managed Cryptographic Object used as the key and the result of the MAC operation.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the Managed Cryptographic Object that is the key to use for the MAC operation. |
| Cryptographic Parameters | No | The Cryptographic Parameters (Cryptographic Algorithm) corresponding to the particular MAC method requested. If there are no Cryptographic Parameters associated with the Managed Cryptographic Object and the algorithm requires parameters then the operation SHALL return with a Result Status of Operation Failed. |
| Data | Yes for single-part. No for multi-part. | The data to be MACed . |
| Correlation Value | No | Specifies the existing stream or by-parts cryptographic operation (as returned from a previous call to this operation). |
| Init Indicator | No | Initial operation as Boolean |
| Final Indicator | No | Final operation as Boolean |

Table 356: MAC Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the Managed Cryptographic Object that is the key used for the MAC operation. |
| MAC Data | Yes for single-part. No for multi-part | The data MACed (as a Byte String). |
| Correlation Value | No | Specifies the stream or by-parts value to be provided in subsequent calls to this operation for performing cryptographic operations. |

Table 357: MAC Response Payload

#### Error Handling - MAC

This section details the specific Result Reasons that SHALL be returned for errors detected in a MAC Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Bad Cryptographic Parameters, Cryptographic Failure, Incompatible Cryptographic Usage Mask, Invalid Correlation Value, Invalid Object Type, Key Value Not Present, Object Not Found, Usage Limit Exceeded, Wrong Key Lifecycle State, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 358: MAC Errors

### MAC Verify

This operation requests the server to perform message authentication code (MAC) verify operation on the provided data using a Managed Cryptographic Object as the key for the MAC verify operation.

The request contains information about the cryptographic parameters (cryptographic algorithm) and the data to be MAC verified and MAY contain the data that was passed to the MAC operation (for those algorithms which need the original data to verify a MAC). The cryptographic parameters MAY be omitted from the request as they can be specified as associated attributes of the Managed Cryptographic Object.

The response contains the Unique Identifier of the Managed Cryptographic Object used as the key and the result of the MAC verify operation. The validity of the MAC is indicated by the Validity Indicator field.

The response message SHALL include the Validity Indicator for single-part MAC Verify operations and for the final part of a multi-part MAC Verify operation. Non-Final parts of multi-part MAC Verify operations SHALL NOT include the Validity Indicator.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the Managed Cryptographic Object that is the key to use for the MAC verify operation. |
| Cryptographic Parameters | No | The Cryptographic Parameters (Cryptographic Algorithm) corresponding to the particular MAC method requested. If there are no Cryptographic Parameters associated with the Managed Cryptographic Object and the algorithm requires parameters then the operation SHALL return with a Result Status of Operation Failed. |
| Data | No | The data that was MACed . |
| MAC Data | Yes for single-part. No for multi-part. | The data to be MAC verified (as a Byte String). |
| Correlation Value | No | Specifies the existing stream or by-parts cryptographic operation (as returned from a previous call to this operation). |
| Init Indicator | No | Initial operation as Boolean |
| Final Indicator | No | Final operation as Boolean |

Table 359: MAC Verify Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the Managed Cryptographic Object that is the key used for the verification operation. |
| Validity Indicator | Yes for single-part. No for multi-part. | An Enumeration object indicating whether the MAC is valid, invalid, or unknown. |
| Correlation Value | No | Specifies the stream or by-parts value to be provided in subsequent calls to this operation for performing cryptographic operations. |

Table 360: MAC Verify Response Payload

#### Error Handling - MAC Verify

This section details the specific Result Reasons that SHALL be returned for errors detected in a MAC Verify Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Bad Cryptographic Parameters, Cryptographic Failure, Incompatible Cryptographic Usage Mask, Invalid Correlation Value, Invalid Object Type, Key Value Not Present, Object Not Found, Permission Denied, Wrong Key Lifecycle State, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 361: MAC Verify Errors

### Modify Attribute

This operation requests the server to modify the value of an existing attribute instance associated with a Managed Object. The request contains the Unique Identifier of the Managed Object whose attribute is to be modified, the OPTIONAL Current Attribute existing value and the New Attribute new value. If no Current Attribute is specified in the request, then if there is only a single instance of the Attribute it SHALL be selected as the attribute instance to be modified to the New Attribute value, and if there are multiple instances of the Attribute an error SHALL be returned (as the specific instance of the attribute is unable to be determined).. Only existing attributes MAY be changed via this operation. Only the specified instance of the attribute SHALL be modified. Specifying a Current Attribute for which there exists no Attribute associated with the object SHALL result in an error.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| Current Attribute | No | Specifies the existing attribute value associated with the object to be modified. |
| New Attribute | Yes | Specifies the new value for the attribute associated with the object . |

Table 362: Modify Attribute Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |

Table 363: Modify Attribute Response Payload

#### Error Handling - Modify Attribute

This section details the specific Result Reasons that SHALL be returned for errors detected in a Modify Attribute Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attribute Instance Not Found, Attribute Not Found, Attribute Read Only, Non Unique Name Attribute, Object Not Found, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large, Wrong Key Lifecycle State |

Table 364: Modify Attribute Errors

### Obliterate

This operation requests the server to remove the Managed Object. All meta-data SHALL also be removed from the server. Any attempt to reference an object with the Unique Identifier SHALL return an Object Not Found error. After this operation has been completed, another object with the same Unique Identifier may be created, imported or registered without needing to set the Replace Existing parameter. The Response Payload SHALL be empty.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | Determines the object being obliterated. |

Table 365: Obliterate Request Payload

|  |  |  |
| --- | --- | --- |
| Reponse Payload | | |
| Item | REQUIRED | Description |
|  |  |  |

Table 366: Obliterate Response Payload

#### Error Handling – Obliterate

This section details the specific Result Reasons that SHALL be returned for errors detected in an Obliterate Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Object Not Found, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Response Too Large |

Table 367: Obliterate Errors

### Obtain Lease

This operation requests the server to obtain a new *Lease Time* for a specified Managed Object. The Lease Time is an interval value that determines when the client's internal cache of information about the object expires and needs to be renewed. If the returned value of the lease time is zero, then the server is indicating that no lease interval is effective, and the client MAY use the object without any lease time limit. If a client's lease expires, then the client SHALL NOT use the associated cryptographic object until a new lease is obtained. If the server determines that a new lease SHALL NOT be issued for the specified cryptographic object, then the server SHALL respond to the Obtain Lease request with an error.

The response payload for the operation contains the current value of the Last Change Date attribute for the object. This MAY be used by the client to determine if any of the attributes cached by the client need to be refreshed, by comparing this time to the time when the attributes were previously obtained.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | Determines the object for which the lease is being obtained. |

Table 368: Obtain Lease Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| Lease Time | Yes | An interval (in seconds) that specifies the amount of time that the object MAY be used until a new lease needs to be obtained. |
| Last Change Date | Yes | The date and time indicating when the latest change was made to the contents or any attribute of the specified object. |

Table 369: Obtain Lease Response Payload

#### Error Handling - Obtain Lease

This section details the specific Result Reasons that SHALL be returned for errors detected in a Obtain Lease Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Object Not Found, Usage Limit Exceeded, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 370: Obtain Lease Errors

### Ping

This operation is used to determine if a server is alive and responding. The server MAY treat the *Ping* operation as a non-logged operation.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
|  |  |  |

Table 371: Ping Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
|  |  |  |

Table 372: Ping Response Payload

### PKCS#11

This operation enables the server to perform a PKCS#11 operation.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| PKCS#11 Interface | No | The name of the interface. If absent, the default V3.0 interface which defines the functions supported. |
| PKCS#11 Function | Yes | The function to perform. An Enumeration for PKCS#11 defined functions or an Integer for vendor defined function. |
| Correlation Value | No | Must be returned to the server if provided in a previous response. |
| PKCS#11 Input Parameters | No | The parameters to the function. The format is specified in the PKCS#11 Profile and the **[PKCS#11]** standard document |

Table 373: PKCS#11 Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| PKCS#11 Interface | No | The name of the interface. If absent,  the default V3.0 interface is used. |
| PKCS#11 Function | Yes | The function that was performed. An  Enumeration for PKCS#11 defined  functions or an Integer for vendor  defined function. |
| Correlation Value | No | Server defined Byte String that the  client must provide in the next request. |
| PKCS#11 Output Parameters | No | The parameters output from the  function. The format is specified in the  PKCS#11 Profile [KMIP-Prof] and the **[PKCS#11]**  standard document. |
| PKCS#11 Return Code | Yes | The PKCS#11 return code as specified  in the CK\_RV values in **[PKCS#11]** |

Table 374: PKCS#11 Response Payload

#### Error Handling – PKCS#11

This section details the specific Result Reasons that SHALL be returned for errors detected in a PKCS#11 Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Asynchronous Correlation Value, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large, PKCS#11 Codec Error, PKCS#11 Invalid Function, PKCS#11 Invalid Interface |

Table 375: PKCS#11 Errors

### Poll

This operation is used to poll the server in order to obtain the status of an outstanding asynchronous operation. The correlation value of the original operation SHALL be specified in the request. The response to this operation SHALL NOT be asynchronous.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Asynchronous Correlation Value | Yes | Specifies the request being polled. |

Table 376: Poll Request Payload

The server SHALL reply with one of two responses:

If the operation has not completed, the response SHALL contain no payload and a Result Status of Pending.

If the operation has completed, the response SHALL contain the appropriate payload for the operation. This response SHALL be identical to the response that would have been sent if the operation had completed synchronously.

#### Error Handling – Poll

This section details the specific Result Reasons that SHALL be returned for errors detected in a Poll Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Asynchronous Correlation Value, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 377: Poll Errors

### Process

This operation requests the server to modify its processing of a previously-submitted asynchronous request such that the next Poll for that asynchronous request SHALL NOT return a “pending” status, effectively changing the processing mode for that batch item to that resembling synchronous processing (note that this may have processing implications for other items in that same.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Asynchronous Correlation Value | Yes | The value of the Asynchronous Correlation Value for the Batch Item to be made “synchronous” |

Table 378: Process Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
|  |  |  |

Table 379: Process Response Payload

#### Error Handling – Process

This section details the specific Result Reasons that SHALL be returned for errors detected in a Process Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Asynchronous Correlation Value, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 380: Process Request Errors

### Query

This operation is used by the client to interrogate the server to determine its capabilities and/or protocol mechanisms.

For each Query Function specified in the request, the corresponding items SHALL be returned in the response.

|  |  |
| --- | --- |
| **Value** | **Description** |
| Query Operations | A list of Operation enumerated values, which SHALL list all the operations that the server supports. |
| Query Object Type | A list of Object Type enumerated values, which SHALL list all the object types that the server supports. |
| Query Server Information | A Server Information structure containing vendor-specific fields and/or substructures. |
| Query Application Namespace | A list of Application Namespace text strings that the server SHALL generate values for if requested by the client. |
| Query Extension List | A list of Extension List structure containing the descriptions of Objects with Item Tag values in the Extensions range that are supported by the server. If the request contains a Query Extension List and/or Query Extension Map value in the Query Function field, then the Extensions Information fields SHALL be returned in the response. |
| Query Extension Map |
| Query Attestation Types | A list of Attestation Type enumerated values, which SHALL list all the attestation types that the server supports. |
| Query RNGs | A list of RNG Parameters structures containing the RNGs supported. The response SHALL list all the Random Number Generators that the server supports. If the request contains a Query RNGs value in the Query Function field, then this field SHALL be returned in the response. |
| Query Validations | A list if Validation Information structures containing details of each formal validation which the server asserts. If the request contains a Query Validations value, then zero or more Validation Information fields SHALL be returned in the response. A server MAY elect to return no validation information in the response. |
| Query Profiles | A list of Profile Information structures containing details of the profiles that a server supports including potentially how it supports that profile. If the request contains a *Query Profiles* value in the *Query Function* field, then this field SHALL be returned in the response if the server supports any Profiles. |
| Query Capability Information | A Capability Information structure containing details of the capability of the server. |
| Query Client Registration Methods | A list of *Client Registration Method* enumerated values, which SHALL list all the client registration methods that the server supports. If the request contains a *Query Client Registration Method* value in the Query Function field, then this field SHALL be returned in the response if the server supports any *Client Registration Methods*. |
| Query Defaults Information | A Defaults Information structure containing Object Defaults structures, which list the default values that the server SHALL use on Cryptographic Objects if the client omits them. |
| Query Protection Storage Masks | An ordered list of ProtectionStorageMask enumerated values for the alternatives that a server supports |
| Query Credential Information | A Credential Information structure containing details of the Credential types that a server supports. |

Table 381: Query Functions

If both Query Extension List and Query Extension Map are specified in the request, then only the response to Query Extension Map SHALL be returned and the Query Extension List SHALL be ignored.

If the Query Function RNG Parameters is specified in the request and If the server is unable to specify details of the RNG then it SHALL return an RNG Parameters with the RNG Algorithm enumeration of Unspecified.

Note that the response payload is empty if there are no values to return.

The Query Function field in the request SHALL contain one or more of the following items:

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Query Function | Yes, MAY be Repeated | Determines the information being queried. |

Table 382: Query Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Operation | No, MAY be repeated | Specifies an Operation that is supported by the server. |
| Object Type | No, MAY be repeated | Specifies a Managed Object Type that is supported by the server. |
| Vendor Identification | No | SHALL be returned if Query Server Information is requested. The Vendor Identification SHALL be a text string that uniquely identifies the vendor. |
| Server Information | No | Contains vendor-specific information possibly be of interest to the client. |
| Application Namespace | No, MAY be repeated | Specifies an Application Namespace supported by the server. |
| Extension Information | No, MAY be repeated | SHALL be returned if Query Extension List or Query Extension Map is requested and supported by the server. |
| Attestation Type | No, MAY be repeated | Specifies an Attestation Type that is supported by the server. |
| RNG Parameters | No, MAY be repeated | Specifies the RNG that is supported by the server. |
| Profile Information | No, MAY be repeated | Specifies the Profiles that are supported by the server. |
| Validation Information | No, MAY be repeated | Specifies the validations that are supported by the server. |
| Capability Information | No | Specifies the capabilities that are supported by the server. |
| Client Registration Method | No, MAY be repeated | Specifies a Client Registration Method that is supported by the server. |
| Defaults Information | No | Specifies the defaults that the server will use if the client omits them. |
| Protection Storage Masks | Yes , if Protection Storage Mask is contained in the Query Function. | Specifies the list of Protection Storage Mask values supported by the server. A server MAY elect to provide an empty list in the Response if it is unable or unwilling to provide this information. |
| Credential Information | Yes, if Credential Information is contained in the Query Function | Specifies the list of Credential types supported. |

Table 383: Query Response Payload

#### Error Handling – Query

This section details the specific Result Reasons that SHALL be returned for errors detected in a Query Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 384: Query Errors

### Query Asynchronous Requests

This operation requests the server to report on any asynchronous requests that have been made for which results have not yet been obtained via the normal Poll (or less-normal Cancel) operation.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Asynchronous Correlation Values | No | Structure containing zero or more Asynchronous Correlation Value. |
| Operations | No | Structure Containing zero or more *Operation*. |

Table 385: Query Asynchronous Requests Request Payload

If no parameters are passed on this operation, the server SHOULD report any asynchronous requests whose results are still outstanding. If Asynchronous Correlation Values are passed, the server SHOULD report the outstanding asynchronous requests whose values match. If Operations are passed, the server SHOULD report the outstanding asynchronous requests whose Operation matches one of those contained in Operations.

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Asynchronous Request | No, MAY be repeated | The details regarding a particular asynchronous request, |

Table 386: PKCS#11 Response Payload

#### Error Handling – Query Asynchronous Requests

This section details the specific Result Reasons that SHALL be returned for errors detected in a Query Asynchronous Requests Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attestation Failed, Attestation Required, Feature Not Supported, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 387: Query Asynchronous Requests Errors

### Recover

This operation is used to obtain access to a Managed Object that has been archived. This request MAY need asynchronous polling to obtain the response due to delays caused by retrieving the object from the archive. Once the response is received, the object is now on-line, and MAY be obtained (e.g., via a Get operation). Special authentication and authorization SHOULD be enforced to perform this request (see [KMIP-UG]).

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | Determines the object being recovered. |

Table 388: Recover Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |

Table 389: Recover Response Payload

#### Error Handling – Recover

This section details the specific Result Reasons that SHALL be returned for errors detected in a Recover Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Object Not Found, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 390: Recover Errors

### Register

This operation requests the server to register a Managed Object that was created by the client or obtained by the client through some other means, allowing the server to manage the object. The arguments in the request are similar to those in the Create operation, but contain the object itself for storage by the server.

The request contains information about the type of object being registered and attributes to be assigned to the object (e.g., Cryptographic Algorithm, Cryptographic Length, etc.). This information SHALL be specified by the use of an Attributes object.

If the Managed Object being registered is wrapped, the server SHALL create a Wrapping Key Link attribute pointing to the Managed Object with which the Managed Object being registered is wrapped.

If the client provides a Unique Identifier value in the set of attributes, the server SHALL use the provided Unique Identifier value unless the Unique Identifier value is already in use within the server (and in which case the server SHALL return a Result Reason of Object Already Exists). A server SHALL accept Unique Identifier values specified as universally unique identifiers represented as 32 hexadecimal (base-16) digits, formatted in five groups of digits separated by hyphens, in the form 8-4-4-4-12 for a total of 36 characters (32 hexadecimal characters and 4 hyphens). A server MAY also accept other formats of Unique Identifier values.

The response contains the registered object’s Unique Identifier as assigned by the server or specified by the client. The Initial Date attribute of the object SHALL be set to the current time.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Object Type | Yes | Determines the type of object being registered. |
| Attributes | Yes | Specifies desired object attributes to be associated with the new object. |
| Any Object (Section 2) | Yes | The object being registered. The object and attributes MAY be wrapped. |
| Protection Storage Masks | No | Specifies all permissible Protection Storage Mask selections for the new object |

Table 391: Register Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the newly registered object. |

Table 392: Register Response Payload

If a Managed Cryptographic Object is registered, then the following attributes SHALL be included in the Register request.

|  |  |
| --- | --- |
| Attribute | REQUIRED |
| Cryptographic Algorithm | Yes, MAY be omitted only if this information is encapsulated in the Key Block. Does not apply to Secret Data. If present, then Cryptographic Length below SHALL also be present. |
| Cryptographic Length | Yes, MAY be omitted only if this information is encapsulated in the Key Block. Does not apply to Secret Data. If present, then Cryptographic Algorithm above SHALL also be present. |
| Certificate Length | Yes. Only applies to Certificates. |
| Cryptographic Usage Mask | Yes. |
| Digital Signature Algorithm | Yes, MAY be omitted only if this information is encapsulated in the Certificate object. Only applies to Certificates. |

Table 393: Register Attribute Requirements

#### Error Handling – Register

This section details the specific Result Reasons that SHALL be returned for errors detected in a Register Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Attribute Read Only, Attribute Single Valued, Bad Password, Encoding Option Error, Invalid Attribute, Invalid Attribute Value, Invalid Object Type, Non Unique Name Attribute, Server Limit Exceeded, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Object Already Exists, Operation Not Supported, Permission Denied, Protection Storage Unavailable, Response Too Large |

Table 394: Register Errors

### Revoke

This operation requests the server to revoke a Managed Cryptographic Object or an Opaque Object. The request contains an option reason for the revocation (e.g., “key compromise”, “CA compromise”, etc.). The operation places the object into the “compromised” state; the Compromise Date is set to the current date and time; and the Compromise Occurrence Date is set to the value in the Revoke request and if a value is not provided in the Revoke request then Compromise Occurrence Date SHOULD be set to the Initial Date for the object. If the Revocation Reason is not provided, then a server SHALL process the request as though the Revocation Reason was specified as Unspecified.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | Determines the object being revoked. |
| Revocation Reason | No | Specifies the reason for revocation. |
| Compromise Occurrence Date | No | Specifies the date on which the compromise occurred if known. |

Table 395: Revoke Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |

Table 396: Revoke Response Payload

#### Error Handling – Revoke

This section details the specific Result Reasons that SHALL be returned for errors detected in a Revoke Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Field, Invalid Object Type, Object Not Found, Wrong Key Lifecycle State, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 397: Revoke Errors

### Re-certify

This request is used to renew an existing Certificate object for the same key pair. Only a single certificate SHALL be renewed at a time.

The Certificate Request Value object MAY be omitted, in which case the public key for which a Certificate object is generated or a Certificate Request object SHALL be specified by its Unique Identifier only. If the Certificate Request Type and the Certificate Request Value objects are omitted from the request and the Unique Identifier does not refer to a Certificate Request, then the Certificate Type SHALL be specified using the Attributes object in the request, then the Certificate Type of the new certificate SHALL be the same as that of the existing certificate.

The Certificate Request is passed as a Byte String, which allows multiple certificate request types for X.509 certificates (e.g., PKCS#10, PEM, etc.) to be submitted to the server.

If the information in the Certificate Request Value field in the request conflicts with the attributes specified in the Attributes, then the information in the Certificate Request Value field takes precedence.

As the new certificate takes over the Name attribute of the existing certificate, Re-certify SHOULD only be performed once on a given (existing) certificate.

For the existing certificate, the server SHALL create a Replacement Object Link attribute pointing to the new certificate. For the new certificate, the server SHALL create a Replaced Object Link attribute pointing to the existing certificate. For the public key, the server SHALL change the Certificate Link attribute to point to the new certificate.

An *Offset* MAY be used to indicate the difference between the Initial Date and the Activation Date of the new certificate. If no Offset is specified, the Activation Date and Deactivation Date values are copied from the existing certificate. If Offset is set and dates exist for the existing certificate, then the dates of the new certificate SHALL be set based on the dates of the existing certificate as follows:

|  |  |
| --- | --- |
| Attribute in Existing Certificate | Attribute in New Certificate |
| Initial Date (*IT1*) | Initial Date (*IT2*) > *IT1* |
| Activation Date (*AT1*) | Activation Date (*AT2*) =  *IT2+ Offset* |
| Deactivation Date (*DT1*) | Deactivation Date = *DT1*+(*AT2*- *AT1*) |

Table 398: Computing New Dates from Offset during Re-certify

Attributes that are not copied from the existing certificate and that are handled in a specific way for the new certificate are:

|  |  |
| --- | --- |
| Attribute | Action |
| Initial Date | Set to current time. |
| Destroy Date | Not set. |
| Revocation Reason | Not set. |
| Unique Identifier | New value generated. |
| Name | Set to the name(s) of the existing certificate; all Name attributes are removed from the existing certificate. |
| State | Set based on attributes values, such as dates. |
| Digest | Recomputed from the new certificate value. |
| Link | Set to point to the existing certificate as the replaced certificate. |
| Last Change Date | Set to current time. |

Table 399: Re-certify Attribute Requirements

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the Certificate being renewed. |
| Certificate Request Unique Identifier | No | The Unique Identifier of the Certificate Request. |
| Certificate Request Type | No | An Enumeration object specifying the type of certificate request. It is REQUIRED if the Certificate Request is present. |
| Certificate Request Value | No | A Byte String object with the certificate request. |
| Offset | No | An Interval object indicating the difference between the Initial Date of the new certificate and the Activation Date of the new certificate. |
| Attributes | No | Specifies desired object attributes. |
| Protection Storage Masks | No | Specifies all permissible Protection Storage Mask selections for the new object |

Table 400: Re-certify Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the new certificate. |

Table 401: Re-certify Response Payload

#### Error Handling - Re-certify

This section details the specific Result Reasons that SHALL be returned for errors detected in a Re-certify Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid CSR, Invalid Message, Invalid Object Type, Object Not Found, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Protection Storage Unavailable, Response Too Large |

Table 402: Re-certify Errors

### Re-key

This request is used to generate a replacement key for an existing symmetric key. It is analogous to the Create operation, except that attributes of the replacement key are copied from the existing key, with the exception of the attributes listed in *Re-key Attribute Requirements*.

As the replacement key takes over the name attribute of the existing key, Re-key SHOULD only be performed once on a given key.

For the existing key, the server SHALL create a Replacement Object Link attribute pointing to the replacement key. For the replacement key, the server SHALL create a Replaced Object Link attribute pointing to the existing key.

An *Offset* MAY be used to indicate the difference between the Initial Date and the Activation Date of the replacement key. If no Offset is specified, the Activation Date, Process Start Date, Protect Stop Date and Deactivation Date values are copied from the existing key. If Offset is set and dates exist for the existing key, then the dates of the replacement key SHALL be set based on the dates of the existing key as follows:

|  |  |
| --- | --- |
| Attribute in Existing Key | Attribute in Replacement Key |
| Initial Date (*IT1*) | Initial Date (*IT2*) > *IT1* |
| Activation Date (*AT1*) | Activation Date (AT2) = IT2+ Offset |
| Process Start Date (*CT1*) | Process Start Date = *CT1*+(*AT2*- *AT1*) |
| Protect Stop Date (*TT1*) | Protect Stop Date = *TT1*+(*AT2*- *AT1*) |
| Deactivation Date (*DT1*) | Deactivation Date = *DT1*+(*AT2*- *AT1*) |

Table 403: Computing New Dates from Offset during Re-key

Attributes requiring special handling when creating the replacement key are:

|  |  |
| --- | --- |
| Attribute | Action |
| Initial Date | Set to the current time |
| Destroy Date | Not set |
| Compromise Occurrence Date | Not set |
| Compromise Date | Not set |
| Revocation Reason | Not set |
| Unique Identifier | New value generated |
| Usage Limits | The Total value is copied from the existing key, and the Count value in the existing key is set to the Total value. |
| Name | Set to the name(s) of the existing key; all Name attributes are removed from the existing key. |
| State | Set based on attributes values, such as dates. |
| Digest | Recomputed from the replacement key value |
| Link | Set to point to the existing key as the replaced key |
| Last Change Date | Set to current time |
| Random Number Generator | Set to the random number generator used for creating the new managed object. Not copied from the original object. |

Table 404: Re-key Attribute Requirements

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | Determines the existing Symmetric Key being re-keyed. |
| Offset | No | An Interval object indicating the difference between the Initial Date and the Activation Date of the replacement key to be created. |
| Attributes | No | Specifies desired object attributes. |
| Protection Storage Masks | No | Specifies all permissible Protection Storage Mask selections for the new object |

Table 405: Re-key Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the newly-created replacement Symmetric Key. |

Table 406: Re-key Response Payload

#### Error Handling - Re-key

This section details the specific Result Reasons that SHALL be returned for errors detected in a Re-key Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Cryptographic Failure, Invalid Field, Invalid Message, Invalid Object Type, Key Value Not Present, Object Not Found, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Protection Storage Unavailable, Response Too Large |

Table 407: Re-key Errors

### Re-key Key Pair

This request is used to generate a replacement key pair for an existing public/private key pair. It is analogous to the Create Key Pair operation, except that attributes of the replacement key pair are copied from the existing key pair, with the exception of the attributes listed in *Re-key Key Pair Attribute Requirements tor.*

As the replacement of the key pair takes over the name attribute for the existing public/private key pair, Re-key Key Pair SHOULD only be performed once on a given key pair.

For both the existing public key and private key, the server SHALL create a Replacement Object Link attribute pointing to the replacement public and private key, respectively. For both the replacement public and private key, the server SHALL create a Replaced Object Link attribute pointing to the existing public and private key, respectively.

An *Offset* MAY be used to indicate the difference between the Initial Date and the Activation Date of the replacement key pair. If no Offset is specified, the Activation Date and Deactivation Date values are copied from the existing key pair. If Offset is set and dates exist for the existing key pair, then the dates of the replacement key pair SHALL be set based on the dates of the existing key pair as follows

|  |  |
| --- | --- |
| Attribute in Existing Key Pair | Attribute in Replacement Key Pair |
| Initial Date (*IT1*) | Initial Date (*IT2*) > *IT1* |
| Activation Date (*AT1*) | Activation Date (*AT2*) =  *IT2+ Offset* |
| Deactivation Date (*DT1*) | Deactivation Date = *DT1*+(*AT2*- *AT1*) |

Table 408: Computing New Dates from Offset during Re-key Key Pair

Attributes for the replacement key pair that are not copied from the existing key pair and which are handled in a specific way are:

|  |  |
| --- | --- |
| Attribute | Action |
| Private Key Unique Identifier | New value generated |
| Public Key Unique Identifier | New value generated |
| Name | Set to the name(s) of the existing public/private keys; all Name attributes of the existing public/private keys are removed. |
| Digest | Recomputed for both replacement public and private keys from the new public and private key values |
| Usage Limits | The Total Bytes/Total Objects value is copied from the existing key pair, while the Byte Count/Object Count values are set to the Total Bytes/Total Objects. |
| State | Set based on attributes values, such as dates. |
| Initial Date | Set to the current time |
| Destroy Date | Not set |
| Compromise Occurrence Date | Not set |
| Compromise Date | Not set |
| Revocation Reason | Not set |
| Link | Set to point to the existing public/private keys as the replaced public/private keys |
| Last Change Date | Set to current time |
| Random Number Generator | Set to the random number generator used for creating the new managed object. Not copied from the original object. |

Table 409: Re-key Key Pair Attribute Requirements

|  |  |  |
| --- | --- | --- |
| **Request Payload** | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | Determines the existing Asymmetric key pair to be re-keyed. |
| Offset | No | An Interval object indicating the difference between the Initial Date and the Activation Date of the replacement key pair to be created. |
| Common Attributes | No | Specifies desired attributes that apply to both the Private and Public Key Objects. |
| Private Key Attributes | No | Specifies attributes that apply to the Private Key Object. |
| Public Key Attributes | No | Specifies attributes that apply to the Public Key Object. |
| Common Protection Storage Masks | No | Specifies all Protection Storage Mask selections that are permissible for the new Private Key and new Public Key objects |
| Private Protection Storage Masks | No | Specifies all Protection  Storage Mask selections that are  permissible for the new Private Key  object. |
| Public Protection Storage Masks | No | Specifies all Protection  Storage Mask selections that are  permissible for the new Public Key  object. |

Table 410: Re-key Key Pair Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Private Key Unique Identifier | Yes | The Unique Identifier of the newly created replacement Private Key object. |
| Public Key Unique Identifier | Yes | The Unique Identifier of the newly created replacement Public Key object. |

Table 411: Re-key Key Pair Response Payload

#### Error Handling - Re-key Key Pair

This section details the specific Result Reasons that SHALL be returned for errors detected in a Re-key Key Pair Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Cryptographic Failure, Invalid Field, Invalid Message, Invalid Object Type, Key Value Not Present, Object Not Found, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Private Protection Storage Unavailable, Public Protection Storage Unavailable, Response Too Large |

Table 412: Re-key Key Pair Errors

### Re-Provision

This request is used to generate a replacement client link level credential from an existing client link level credential. The client requesting re-provisioning SHALL provide a certificate signing request, or a certificate, or no parameters if the server will create the client credential .

If the client provides a certificate signing request, the server SHALL process the certificate signing request and assign the new certificate to the be the client link level credential. The server SHALL return the unique identifier for the signed certificate stored on the server.

If the client provides a certificate, the server SHALL associate the certificate with the client as the client’s link level credential. The server SHALL return the unique identifier for the certificate stored on the server.

Where no parameters are provided, the server shall generate a key pair and certificate associated with the client. The server SHALL return the unique identifier for the private key. The client may then subsequently retrieve the private key via a Get operation.

The current client credential SHALL be made invalid and cannot be used in future KMIP requests.

Re-Provision SHALL be called by the client that requires new credentials

Re-Provision SHOULD fail if the certificate that represents the client credential has expired.

Re-Provision SHALL fail if the certificate that represents the client credential has been Revoked.

Re-Provision SHALL fail if the certificate that represents the client credential has been compromised.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Certificate Request | No | The certificate request to be signed |
| Certificate | No | The certificate to replace the existing certificate |

Table 413: Re-Provision Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | No | The Certificate or Private Key unique identifier |

Table 414: Re-Provision Response Payload

#### Error Handling – Re-Provision

This section details the specific Result Reasons that SHALL be returned for errors detected in a Re-Provision Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Cryptographic Failure, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 415: RNG Retrieve Errors

### RNG Retrieve

This operation requests the server to return output from a Random Number Generator (RNG).

The request contains the quantity of output requested.

The response contains the RNG output.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Data Length | Yes | The amount of random number generator output to be returned (in bytes). |

Table 416: RNG Retrieve Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Data | Yes | The random number generator output. |

Table 417: RNG Retrieve Response Payload

#### Error Handling - RNG Retrieve

This section details the specific Result Reasons that SHALL be returned for errors detected in a RNG Retrieve Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Cryptographic Failure, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 418: RNG Retrieve Errors

### RNG Seed

This operation requests the server to seed a Random Number Generator.

The request contains the seeding material.

The response contains the amount of seed data used.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

The server MAY elect to ignore the information provided by the client (i.e. not accept the seeding material) and MAY indicate this to the client by returning zero as the value in the Data Length response. A client SHALL NOT consider a response from a server which does not use the provided data as an error.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Data | Yes | The data to be provided as a seed to the random number generator. |

Table 419: RNG Seed Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Data Length | Yes | The amount of seed data used (in bytes). |

Table 420: RNG Seed Response Payload

#### Error Handling - RNG Seed

This section details the specific Result Reasons that SHALL be returned for errors detected in a RNG Seed Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Cryptographic Failure, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 421: RNG Seed Errors

### Set Attribute

This operation requests the server to either add or modify an attribute. The request contains the Unique Identifier of the Managed Object to which the attribute pertains, along with the attribute and value. If the object did not have any instances of the attribute, one is created. If the object had exactly one instance, then it is modified. If it has more than one instance an error is raised. Read-Only attributes SHALL NOT be added or modified using this operation.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| New Attribute | Yes | Specifies the new value for the attribute associated with the object. |

Table 422: Set Attribute Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the Object. |

Table 423: Set Attribute Response Payload

#### Error Handling - Set Attribute

This section details the specific Result Reasons that SHALL be returned for errors detected in a Add Attribute Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Attribute Value, Invalid Attribute Value, Multi Valued Attribute, Non Unique Name Attribute, Object Not Found, Read Only Attribute, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large, Wrong Key Lifecycle State |

Table 424: Set Attribute Errors

### Set Constraints

This operation instructs the server to set the constraints that will be applied to Managed Objects during

operations.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Constraints | Yes | The set of Constraints to apply during  operations. |

Table 425: Set Constraints Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
|  |  |  |

Table 426: Set Constraints Response Payload

#### Error Handling - Set Constraints

This section details the specific Result Reasons that SHALL be returned for errors detected in a Set

Constraints Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Field, Invalid Object Type, Attestation Failed,  Attestation Required, Feature Not Supported, Invalid  Field, Invalid Message, Operation Not Supported,  Permission Denied, Response Too Large |

Table 427: Set Constraints Errors

### Set Defaults

This operation instructs the server to set the default attributes that will be applied to Managed Objects during factory operations if the client does not supply values for mandatory attributes.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Defaults Information | No | The set of Object Defaults to begin using. If no Defaults Information is  supplied, the semantic is to remove all Object Defaults from the server. |

Table 428: Set Defaults Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
|  |  |  |

Table 429: Set Defaults Response Payload

#### Error Handling - Set Defaults

This section details the specific Result Reasons that SHALL be returned for errors detected in a Set

Defaults Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Field, Invalid Object Type, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 430: Set Defaults Errors

### Set Endpoint Role

This operation requests specifying the role of server for subsequent requests and responses over the current client-to-server communication channel. After successful completion of the operation the server assumes the client role, and the client assumes the server role, but the communication channel remains as established.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Endpoint Role | Yes | The endpoint role for the server to apply. |

Table 431: Set Endpoint Role Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Endpoint Role | Yes | The accepted endpoint role as applied by the server. |

Table 432: Set Endpoint Role Response Payload

#### Error Handling - Set Endpoint Role

This section details the specific Result Reasons that SHALL be returned for errors detected in a Set Endpoint Role Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Permission Denied, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 433: Set Endpoint Role Errors

### Sign

This operation requests the server to perform a signature operation on the provided data using a Managed Cryptographic Object as the key for the signature operation.

The request contains information about the cryptographic parameters (digital signature algorithm or cryptographic algorithm and hash algorithm) and the data to be signed. The cryptographic parameters MAY be omitted from the request as they can be specified as associated attributes of the Managed Cryptographic Object.

If the Managed Cryptographic Object referenced has a Usage Limits attribute then the server SHALL obtain an allocation from the current Usage Limits value prior to performing the signing operation. If the allocation is unable to be obtained the operation SHALL return with a result status of Operation Failed and result reason of Permission Denied.

The response contains the Unique Identifier of the Managed Cryptographic Object used as the key and the result of the signature operation.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the Managed Cryptographic Object that is the key to use for the signature operation. |
| Cryptographic Parameters | No | The Cryptographic Parameters (Digital Signature Algorithm or Cryptographic Algorithm and Hashing Algorithm) corresponding to the particular signature generation method requested. If there are no Cryptographic Parameters associated with the Managed Cryptographic Object and the algorithm requires parameters then the operation SHALL return with a Result Status of Operation Failed. |
| Data | Yes for single-part, unless Digested Data is supplied.. No for multi-part. | The data to be. |
| Digested Data | No | The digested data to be signed (as a Byte String). |
| Correlation Value | No | Specifies the existing stream or by-parts cryptographic operation (as returned from a previous call to this operation). |
| Init Indicator | No | Initial operation as Boolean |
| Final Indicator | No | Final operation as Boolean |

Table 434: Sign Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the Managed Cryptographic Object that is the key used for the signature operation. |
| Signature Data | Yes for single-part. No for multi-part. | The signed data (as a Byte String). |
| Correlation Value | No | Specifies the stream or by-parts value to be provided in subsequent calls to this operation for performing cryptographic operations. |

Table 435: Sign Response Payload

#### Error Handling - Sign

This section details the specific Result Reasons that SHALL be returned for errors detected in a sign Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Bad Cryptographic Parameters, Cryptographic Failure, Incompatible Cryptographic Usage Mask, Invalid Correlation Value, Invalid Object Type, Invalid Object Type, Key Value Not Present, Object Not Found, Unsupported Cryptographic Parameters, Usage Limit Exceeded, Wrong Key Lifecycle State, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 436: Sign Errors

### Signature Verify

This operation requests the server to perform a signature verify operation on the provided data using a Managed Cryptographic Object as the key for the signature verification operation.

The request contains information about the cryptographic parameters (digital signature algorithm or cryptographic algorithm and hash algorithm) and the signature to be verified and MAY contain the data that was passed to the signing operation (for those algorithms which need the original data to verify a signature).

The cryptographic parameters MAY be omitted from the request as they can be specified as associated attributes of the Managed Cryptographic Object.

The response contains the Unique Identifier of the Managed Cryptographic Object used as the key and the OPTIONAL data recovered from the signature (for those signature algorithms where data recovery from the signature is supported). The validity of the signature is indicated by the Validity Indicator field.

The response message SHALL include the Validity Indicator for single-part Signature Verify operations and for the final part of a multi-part Signature Verify operation. Non-Final parts of multi-part Signature Verify operations SHALL NOT include the Validity Indicator.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the Managed Cryptographic Object that is the key to use for the signature verify operation. |
| Cryptographic Parameters | No | The Cryptographic Parameters (Digital Signature Algorithm or Cryptographic Algorithm and Hashing Algorithm) corresponding to the particular signature verification method requested.  If there are no Cryptographic Parameters associated with the Managed Cryptographic Object and the algorithm requires parameters then the operation SHALL return with a Result Status of Operation Failed. |
| Data | No | The data that was. |
| Digested Data | No | The digested data to be verified (as a Byte String) |
| Signature Data | Yes for single-part. No for multi-part. | The signature to be verified (as a Byte String). |
| Correlation Value | No | Specifies the existing stream or by-parts cryptographic operation (as returned from a previous call to this operation). |
| Init Indicator | No | Initial operation as Boolean |
| Final Indicator | No | Final operation as Boolean |

Table 437: Signature Verify Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the Managed Cryptographic Object that is the key used for the verification operation. |
| Validity Indicator | Yes for single-part. No for multi-part. | An Enumeration object indicating whether the signature is valid, invalid, or unknown. |
| Data | No | The OPTIONAL recovered data (as a Byte String) for those signature algorithms where data recovery from the signature is supported. |
| Correlation Value | No | Specifies the stream or by-parts value to be provided in subsequent calls to this operation for performing cryptographic operations. |

Table 438: Signature Verify Response Payload

#### Error Handling - Signature Verify

This section details the specific Result Reasons that SHALL be returned for errors detected in a signature Verify Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Bad Cryptographic Parameters, Cryptographic Failure, Incompatible Cryptographic Usage Mask, Invalid Correlation Value, Invalid Object Type, Invalid Object Type, Key Value Not Present, Object Not Found, Unsupported Cryptographic Parameters, Wrong Key Lifecycle State, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 439: Signature Verify Errors

### Validate

This operation requests the server to validate a certificate chain and return information on its validity. Only a single certificate chain SHALL be included in each request.

The request MAY contain a list of certificate objects, and/or a list of Unique Identifiers that identify Managed Certificate objects. Together, the two lists compose a certificate chain to be validated. The request MAY also contain a date for which all certificates in the certificate chain are REQUIRED to be valid.

The method or policy by which validation is conducted is a decision of the server and is outside of the scope of this protocol. Likewise, the order in which the supplied certificate chain is validated and the specification of trust anchors used to terminate validation are also controlled by the server.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Certificate | No, MAY be repeated | One or more Certificates. |
| Unique Identifier | No, MAY be repeated | One or more Unique Identifiers of Certificate Objects. |
| Validity Date | No | A Date-Time object indicating when the certificate chain needs to be valid. If omitted, the current date and time SHALL be assumed. |

Table 440: Validate Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Validity Indicator | Yes | An Enumeration object indicating whether the certificate chain is valid, invalid, or unknown. |

Table 441: Validate Response Payload

#### Error Handling – Validate

This section details the specific Result Reasons that SHALL be returned for errors detected in a Validate Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Invalid Field, Invalid Object Type, Object Not Found, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 442: Validate Errors

## Server-to-Client Operations

Server-to-client operations are used by servers to send information or Managed Objects to clients via means outside of the normal client-server request-response mechanism. These operations are used to send Managed Objects directly to clients without a specific request from the client.

### Discover Versions

This operation is used by the server to determine a list of protocol versions that is supported by the client. The request payload contains an OPTIONAL list of protocol versions that is supported by the server. The protocol versions SHALL be ranked in decreasing order of preference.

The response payload contains a list of protocol versions that are supported by the client. The protocol versions are ranked in decreasing order of preference. If the server provides the client with a list of supported protocol versions in the request payload, the client SHALL return only the protocol versions that are supported by both the client and server. The client SHOULD list all the protocol versions supported by both client and server. If the protocol version specified in the request header is not specified in the request payload and the client does not support any protocol version specified in the request payload, the client SHALL return an empty list in the response payload. If no protocol versions are specified in the request payload, the client SHOULD return all the protocol versions that are supported by the client.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Protocol Version | No, MAY be Repeated | The list of protocol versions supported by the server ordered in decreasing order of preference. |

Table 443: Discover Versions Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Protocol Version | No, MAY be repeated | The list of protocol versions supported by the client ordered in decreasing order of preference. |

#### Error Handling – Discover Versions

This section details the specific Result Reasons that SHALL be returned for errors detected in a Discover Versions Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Permission Denied, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 444: Discover Versions Errors

### Notify

This operation is used to notify a client of events that resulted in changes to attributes of an object. This operation is only ever sent by a server to a client via means outside of the normal client request/response protocol, using information known to the server via unspecified configuration or administrative mechanisms. It contains the Unique Identifier of the object to which the notification applies, and a list of the attributes whose changed values or deletion have triggered the notification. The client SHALL send a response in the form of a Response containing no payload, unless both the client and server have prior knowledge (obtained via out-of-band mechanisms) that the client is not able to respond.

|  |  |  |
| --- | --- | --- |
| Message Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| Attributes | No | The attributes that have changed. This includes at least the Last Change Date attribute. |
| Attribute Reference | No, may be repeated | The attributes that have been deleted. |

#### Error Handling – Notify

This section details the specific Result Reasons that SHALL be returned for errors detected in a Notify Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Permission Denied, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 445: Notify Message Errors

### Put

This operation is used to “push” Managed Objects to clients. This operation is only ever sent by a server to a client via means outside of the normal client request/response protocol, using information known to the server via unspecified configuration or administrative mechanisms. It contains the Unique Identifier of the object that is being sent, and the object itself. The client SHALL send a response in the form of a Response Message containing no payload, unless both the client and server have prior knowledge (obtained via out-of-band mechanisms) that the client is not able to respond.

The *Put Function* field indicates whether the object being “pushed” is a new object, or is a replacement for an object already known to the client (e.g., when pushing a certificate to replace one that is about to expire, the Put Functionfield would be set to indicate replacement, and the Unique Identifier of the expiring certificate would be placed in the *Replaced Unique Identifier* field). The Put Function SHALL contain one of the following values:

* *New* – which indicates that the object is not a replacement for another object.
* *Replace* – which indicates that the object is a replacement for another object, and that the Replaced Unique Identifier field is present and contains the identification of the replaced object. In case the object with the Replaced Unique Identifier does not exist at the client, the client SHALL interpret this as if the Put Function contained the value New.

The Attribute field contains one or more attributes that the server is sending along with the object. The server MAY include the attributes associated with the object.

|  |  |  |
| --- | --- | --- |
| Message Payload | | |
| Item | REQUIRED | Description |
| Unique Identifier | Yes | The Unique Identifier of the object. |
| Put Function | Yes | Indicates function for Put message. |
| Replaced Unique Identifier | No | Unique Identifier of the replaced object. SHALL be present if the *Put Function* is *Replace.* |
| All Objects | Yes | The object being sent to the client. |
| Attributes | No | The additional attributes that the server wishes to send with the object. |

#### Error Handling – Put

This section details the specific Result Reasons that SHALL be returned for errors detected in a Put Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Permission Denied, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 446: Put Errors

### Query

This operation is used by the server to interrogate the client to determine its capabilities and/or protocol mechanisms. The *Query Function* field in the request SHALL contain one or more of the following items:

* Query Operations
* Query Objects
* Query Server Information
* Query Extension List
* Query Extension Map
* Query Attestation Types
* Query RNGs
* Query Validations
* Query Profiles
* Query Capabilities
* Query Client Registration Methods

The *Operation* fields in the response contain Operation enumerated values, which SHALL list all the operations that the client supports. If the request contains a Query Operations value in the Query Function field, then these fields SHALL be returned in the response.

The *Object Type* fields in the response contain Object Type enumerated values, which SHALL list all the object types that the client supports. If the request contains a *Query Objects* value in the Query Function field, then these fields SHALL be returned in the response.

The *Server Information* field in the response is a structure containing vendor-specific fields and/or substructures. If the request contains a *Query Server Information* value in the Query Function field, then this field SHALL be returned in the response.

The *Extension Information* fields in the response contain the descriptions of Objects with Item Tag values in the Extensions range that are supported by the server. If the request contains a *Query Extension List* and/or *Query Extension Map* value in the Query Function field, then the Extensions Information fields SHALL be returned in the response. If the Query Function field contains the Query Extension Map value, then the Extension Tag and Extension Type fields SHALL be specified in the Extension Information values. If both Query Extension List and Query Extension Map are specified in the request, then only the response to Query Extension Map SHALL be returned and the Query Extension List SHALL be ignored.

The *Attestation Type* fields in the response contain Attestation Type enumerated values, which SHALL list all the attestation types that the client supports. If the request contains a *Query Attestation Types* value in the Query Function field, then this field SHALL be returned in the response if the server supports any Attestation Types.

The *RNG Parameters* fields in the response SHALL list all the Random Number Generators that the client supports. If the request contains a *Query RNGs* value in the Query Function field, then this field SHALL be returned in the response. If the server is unable to specify details of the RNG then it SHALL return an *RNG Parameters* with the *RNG Algorithm* enumeration of *Unspecified*.

The *Validation Information* field in the response is a structure containing details of each formal validation which the client asserts. If the request contains a *Query Validations* value, then zero or more *Validation Information* fields SHALL be returned in the response. A client MAY elect to return no validation information in the response.

A *Profile Information* field in the response is a structure containing details of the profiles that a client supports including potentially how it supports that profile. If the request contains a *Query Profiles* value in the Query Function field, then this field SHALL be returned in the response if the client supports any Profiles.

The *Capability Information* fields in the response contain details of the capability of the client.

The *Client Registration Method* fields in the response contain Client Registration Method enumerated values, which SHALL list all the client registration methods that the client supports. If the request contains a *Query Client Registration Methods* value in the Query Function field, then this field SHALL be returned in the response if the server supports any Client Registration Methods.

Note that the response payload is empty if there are no values to return.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Query Function | Yes, MAY be Repeated | Determines the information being queried. |

Table 447: Query Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Operation | No, MAY be repeated | Specifies an Operation that is supported by the client. |
| Object Type | No, MAY be repeated | Specifies a Managed Object Type that is supported by the client. |
| Vendor Identification | No | SHALL be returned if Query Server Information is requested. The Vendor Identification SHALL be a text string that uniquely identifies the vendor. |
| Server Information | No | Contains vendor-specific information in response to the Query. |
| Extension Information | No, MAY be repeated | SHALL be returned if Query Extension List or Query Extension Map is requested and supported by the client. |
| Attestation Type | No, MAY be repeated | Specifies an Attestation Type that is supported by the client. |
| RNG Parameters | No, MAY be repeated | Specifies the RNG that is supported by the client. |
| Profile Information | No, MAY be repeated | Specifies the Profiles that are supported by the client. |
| Validation Information | No, MAY be repeated | Specifies the validations that are supported by the client. |
| Capability Information | No, MAY be repeated | Specifies the capabilities that are supported by the client. |
| Client Registration Method | No, MAY be repeated | Specifies a Client Registration Method that is supported by the client. |

#### Error Handling – Query

This section details the specific Result Reasons that SHALL be returned for errors detected in a Query Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Permission Denied, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 448: Query Errors

### Set Endpoint Role

This operation requests specifying the role of server for subsequent requests and responses over the current client-to-server communication channel. After successful completion of the operation the server assumes the client role, and the client assumes the server role, but the communication channel remains as established.

|  |  |  |
| --- | --- | --- |
| Request Payload | | |
| Item | REQUIRED | Description |
| Endpoint Role | Yes | The endpoint role for the client to apply. |

Table 449: Set Endpoint Role Request Payload

|  |  |  |
| --- | --- | --- |
| Response Payload | | |
| Item | REQUIRED | Description |
| Endpoint Role | Yes | The accepted endpoint role as applied by the client. |

Table 450: Set Endpoint Role Response Payload

#### Error Handling - Set Endpoint Role

This section details the specific Result Reasons that SHALL be returned for errors detected in a Set Endpoint Role Operation.

|  |  |
| --- | --- |
| **Result Status** | **Result Reason** |
| Operation Failed | Permission Denied, Attestation Failed, Attestation Required, Feature Not Supported, Invalid Field, Invalid Message, Operation Not Supported, Permission Denied, Response Too Large |

Table 451: Set Endpoint Role Errors

# Operations Data Structures

Common structure used across multiple operations

## Asynchronous Correlation Values

A list of Asynchronous Correlation Values.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Asynchronous Correlation Values | Structure |  |
| Asynchronous Correlation Value | Byte String | No |

Table 452 Asynchronous Correlation Values Structure

## Asynchronous Request

The Asynchronous Request structure contains details of an asynchronous request whose status has not yet been obtained by the submitter.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Asynchronous Request | Structure |  |
| Asynchronous Correlation Value | Byte String | Yes |
| Operation | Enumeration | Yes |
| Submission Date | Date Time Extended | Yes |
| Processing Stage | Enumeration | Yes |

Table 453 Asynchronous Request Structure

## Authenticated Encryption Additional Data

The Authenticated Encryption Additional Data object is used in authenticated encryption and decryption operations that require the transmission of that data between client and server.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Authenticated Encryption Additional Data | Byte String | No |

Table 454 Authenticated Encryption Additional Data

## Authenticated Encryption Tag

The Authenticated Encryption Tag object is used to validate the integrity of the data encrypted and decrypted in “Authenticated Encryption” mode. See [SP800-38D].

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Authenticated Encryption Tag | Byte String | No |

Table 455 Authenticated Encryption Tag

## Capability Information

The *Capability Information* base object is a structure that contains details of the supported capabilities.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Capability Information | Structure |  |
| Streaming Capability | Boolean | No |
| Asynchronous Capability | Boolean | No |
| Attestation Capability | Boolean | No |
| Batch Undo Capability | Boolean | No |
| Batch Continue Capability | Boolean | No |
| Unwrap Mode | Enumeration | No |
| Destroy Action | Enumeration | No |
| Shredding Algorithm | Enumeration | No |
| RNG Mode | Enumeration | No |
| Quantum Safe Capability | Boolean | No |

Table 456: Capability Information Structure

## Constraint

The *Constraint* is a structure that contains details of a constraint that is applied to operations that create

Managed Objects.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Constraint | Structure | YES |
| Object Types | Structure | No |
| Object Groups | Structure | No |
| Attributes | Structure | No |

Table 457: Constraint Structure

## Constraints

A set of Constraint structures.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Constraints | Structure | YES |
| Constraint | Structure | No, May be repeated. |

Table 458: Constraints Structure

## Correlation Value

The *Correlation Value* is used in requests and responses in cryptographic operations that support multi-part (streaming) operations. This is generated by the server and returned in the first response to an operation that is being performed across multiple requests. Note: the server decides which operations are supported for multi-part usage. A server-generated correlation value SHALL be specified in any subsequent cryptographic operations that pertain to the original operation.

|  |  |
| --- | --- |
| Object | Encoding |
| Correlation Value | Byte String |

Table 459: Correlation Value Structure

## Credential Information

The Credential Information operations data object is a structure that contains a list of *Credential Types*. A server SHALL return at least one Credential Type within the structure.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | Required |
| Credential Information | Structure |  |
| Credential Type | Enumeration, may be repeated | Yes |

Table 460: Credential Information Structure

## Data

The *Data* object is used in requests and responses in cryptographic operations that pass data between the client and the server.

|  |  |
| --- | --- |
| Encoding | Description |
| Byte String | The Data |
| Enumeration | Data Enumeration |
| Integer | Zero based nth Data in the response. If negative the count is backwards from the beginning of the current operation’s batch item. |

Table 461: Data encoding descriptions

|  |  |
| --- | --- |
| Object | Encoding |
| Data | Byte String, Enumeration or Integer |

Table 462: Data

## Data Length

The *Data Length* is used in requests in cryptographic operations to indicate the amount of data expected in a response.

|  |  |
| --- | --- |
| Object | Encoding |
| Data Length | Integer |

Table 463: Data Length Structure

## Defaults Information

The *Defaults Information* is a structure used in Query responses for values that servers will use if clients omit them from factory operations requests.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Defaults Information | Structure |  |
| Object Defaults | Structure, may be repeated | No |

Table 464: Defaults Information Structure

## Derivation Parameters

The Derivation Parameters for all derivation methods consist of the following parameters.

|  |  |  |
| --- | --- | --- |
| **Object** | **Encoding** | **REQUIRED** |
| Derivation Parameters | Structure | Yes. |
| Cryptographic Parameters, | Structure | No, depends on the PRF. |
| Initialization Vector | Byte String | No, depends on the PRF (if different than those defined in **[PKCS#5]**) and mode of operation: an empty IV is assumed if not provided. |
| Derivation Data | Byte String | Yes, unless the Unique Identifier of a Secret Data object is provided. May be repeated. |
| Salt | Byte String | Yes if Derivation method is PBKDF2. |
| Iteration Count | Integer | Yes if Derivation method is PBKDF2. |

Table 465: Derivation Parameters Structure

Cryptographic Parameters identify the Pseudorandom Function (PRF) or the mode of operation of the PRF (e.g., if a key is to be derived using the HASH derivation method, then clients are REQUIRED to indicate the hash algorithm inside Cryptographic Parameters; similarly, if a key is to be derived using AES in CBC mode, then clients are REQUIRED to indicate the Block Cipher Mode).

If a key is derived using HMAC, then the attributes of the derivation key provide enough information about the PRF, and the Cryptographic Parameters are ignored.

Derivation Data is either the data to be encrypted, hashed, or HMACed. For the NIST SP 800-108 methods [SP800-108], Derivation Data is Label||{0x00}||Context, where the all-zero byte is optional.

Most derivation methods (e.g., Encrypt) REQUIRE a derivation key and the derivation data to be used. The HASH derivation method REQUIRES either a derivation key or derivation data. Derivation data MAY either be explicitly provided by the client with the Derivation Data field or implicitly provided by providing the Unique Identifier of a Secret Data object. If both are provided, then an error SHALL be returned.

For the AWS Signature Version 4 derivation method, the Derivation Data is (in order) the Date, Region, and Service.

For the HKDF derivation method, the Input Key Material is provided by the specified managed object, the salt is provided in the Salt field of the Derivation Parameters, the optional information is provided in the Derivation Data field of the Derivation Parameters, the output length is specified in the Cryptographic Length attribute provided in the Attributes request parameter. The default hash function is SHA-256 and may be overridden by specifying a Hashing Algorithm in the Cryptographic Parameters field of the Derivation Parameters.

## Extension Information

An *Extension Information* object is a structure describing Objects with Item Tag values in the Extensions range. The Extension Name is a Text String that is used to name the Object. The Extension Tag is the Item Tag Value of the Object. The Extension Type is the Item Type Value of the Object.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Extension Information | Structure |  |
| Extension Name | Text String | Yes |
| Extension Tag | Integer | No |
| Extension Type | Enumeration (Item Type) | No |
| Extension Enumeration | Integer | No |
| Extension Attribute | Boolean | No |
| Extension Parent Structure Tag | Integer | No |
| Extension Description | Text String | No |

Table 466: Extension Information Structure

## Final Indicator

The *Final Indicator* is used in requests in cryptographic operations that support multi-part (streaming) operations. This is provided in the final (last) request with a value of True to an operation that is being performed across multiple requests.

|  |  |
| --- | --- |
| Object | Encoding |
| Final Indicator | Boolean |

Table 467: Final Indicator Structure

## Interop Function

The *IInterop Functrion* is used in requests and responses in to indicate the commencement or completion of an interop operation.

|  |  |
| --- | --- |
| Object | Encoding |
| Interop Function | Enumeration |

Table 468: Interop Function Structure

## Interop Identifier

The *Interop Identifier* is used in requests and responses in to indicate that which interop test is being performed.

|  |  |
| --- | --- |
| Object | Encoding |
| Interop identifier | TextString |

Table 469: Interop Function Structure

## Init Indicator

The *Init Indicator* is used in requests in cryptographic operations that support multi-part (streaming) operations. This is provided in the first request with a value of True to an operation that is being performed across multiple requests.

|  |  |
| --- | --- |
| Object | Encoding |
| Init Indicator | Boolean |

Table 470: Init Indicator Structure

## Key Wrapping Specification

This is a separate structure that is defined for operations that provide the option to return wrapped keys. The *Key Wrapping Specification* SHALL be included inside the operation request if clients request the server to return a wrapped key. If Cryptographic Parameters are specified in the Encryption Key Information and/ortheMAC/Signature Key Information of the Key Wrapping Specification, then the server SHALL verify that they match one of the instances of the Cryptographic Parameters attribute of the corresponding key.. If the corresponding key does not have any Cryptographic Parameters attribute, or if no match is found, then an error is returned.

This structure contains:

* A Wrapping Method that indicates the method used to wrap the Key Value*.*
* Encryption Key Information with the Unique Identifier value of the encryption key and associated cryptographic parameters.
* MAC/Signature Key Information with the Unique Identifier value of the MAC/signature key and associated cryptographic parameters.
* Zero or more Attribute Names to indicate the attributes to be wrapped with the key material.
* An Encoding Option, specifying the encoding of the Key Value before wrapping. If No Encoding is specified, then the Key Value SHALL NOT contain any attributes

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Key Wrapping Specification | Structure |  |
| Wrapping Method | Enumeration | Yes |
| Encryption Key Information | Structure | No, SHALL be present if MAC/Signature Key Information is omitted |
| MAC/Signature Key Information | Structure | No, SHALL be present if Encryption Key Information is omitted |
| Attribute Reference | Text String, MAY be repeated | No |
| Encoding Option | Enumeration | No. If Encoding Option is not present, the wrapped Key Value SHALL be TTLV encoded. |

Table 471: Key Wrapping Specification Object Structure

## Log Message

The *Log Message* is used in the Log operation.

|  |  |
| --- | --- |
| Object | Encoding |
| Log Message | Text String |

Table 472: Log Message Structure

## MAC Data

The *MAC Data* is used in requests and responses in cryptographic operations that pass MAC data between the client and the server.

|  |  |
| --- | --- |
| Object | Encoding |
| MAC Data | Byte String |

Table 473: MAC Data Structure

## Objects

A list of Object Unique Identifiers.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Objects | Structure |  |
| Unique Identifier | Identifier, Unique Identifier, Enumeration or Integer | No, May be repeated. |

Table 474: Objects Structure

## Object Defaults

The *Object Defaults* is a structure that details the values that the server will use if the client omits them on factory methods for objects. The structure list the Attributes and their values by Object Type enumeration, as well as the Object Group(s) for which such defaults pertain (if not pertinent to ALL Object Group values).

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Object Defaults | Structure |  |
| Object Type | ObjectTypes | Enumeration | Structure | Yes |
| Attributes | Structure | Yes |
| Object Groups | Structure | No |

Table 475: Object Defaults Structure

## Object Groups

The *Object Groups* is a structure that lists the relevant Object Group Attributes and their values.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Object Groups | Structure |  |
| Group Link | Reference or Name Reference, May be repeated | No |

Table 476: Object Groups Structure

## Object Types

The *Object Types* is a list of Object Type(s).

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Object Types | Structure |  |
| Object Type | Enumeration | No, May be repeated. |

Table 477: ObjectTypes Structure

## Operations

A list of Operations.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Operations | Structure |  |
| Operation | Enumeration | No, May be repeated. |

Table 478: Operations Structure

## PKCS#11 Function

The *PKCS#11* *Function* structure contains details of the PKCS#11 Function. Specific fields MAY pertain only to certain types of profiles.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| PKCS#11 Function | Structure |  |
| PKCS#11 Function | Enumeration | Yes |

Table 479: PKCS#11 Function Structure

## PKCS#11 Input Parameters

The *PKCS#11* *Input Parameters* structure contains details of the PKCS#11 Input Parameters. Specific fields MAY pertain only to certain types of profiles.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| PKCS#11 Input Parameters | Structure |  |
| PKCS#11 Input Parameters | ByteString | No |

Table 480: PKCS#11 Input Parameters Structure

## PKCS#11 Interface

The *PKCS#11* *Interface* structure contains details of the PKCS#11 Interface. Specific fields MAY pertain only to certain types of profiles.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| PKCS#11 Interface | Structure |  |
| PKCS#11 Interface | TextString | No |

Table 481: PKCS#11 Interface Structure

## PKCS#11 Output Parameters

The *PKCS#11* *Output Parameters* structure contains details of the PKCS#11 Output Parameters. Specific fields MAY pertain only to certain types of profiles.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| PKCS#11 Output Parameters | Structure |  |
| PKCS#11 Output Parameters | ByteString | No |

Table 482: PKCS#11 Output Parameters Structure

## PKCS#11 Return Code

The *PKCS#11* *Return Code* structure contains details of the PKCS#11 Return Code. Specific fields MAY pertain only to certain types of profiles.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| PKCS#11 Return Code | Structure |  |
| PKCS#11 Return Code | Enumeration | Yes |

Table 483: PKCS#11 Return Code Structure

## Profile Information

The *Profile Information* structure contains details of the supported profiles. Specific fields MAY pertain only to certain types of profiles.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Profile Information | Structure |  |
| Profile Name | Enumeration | Yes |
| Profile Version | Structure | No |
| Server URI | Text String | No |
| Server Port | Integer | No |

Table 484: Profile Information Structure

## Profile Version

The *Profile Version* structure contains the version number of the profile, ensuring that the profile is fully understood by both communicating parties. The version number SHALL be specified in two parts, major and minor.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Profile Version | Structure |  |
| Profile Version Major | Integer | Yes |
| Profile Version Minor | Integer | Yes |

Table 485: Profile Version Structure

## Protection Storage Masks

The Protection Storage Masks operations data object is a structure that contains an ordered collection of

*Protection Storage Mask* selections acceptable to the client. The server MAY service the request with

ANY *Storage Protection Mask* that the client passes, but SHALL return an error if no single *Storage*

*Protection Mask* can be satisfied in its entirety (all bits match). When more than one *Protection Storage*

*Mask* is specified by a Client, then all are acceptable alternatives. Note that there are also variants of

*Protection Storage Masks* to deal with the operations that deal with asymmetric pairs (*Common*

*Protection Storage Masks*, *Private Protection Storage Masks* and *Public Protection Storage Masks*), as

the client may have different requirements on the parts of the pair, but the layout of those variants is

identical to the one expressed here.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Protection Storage Masks | Structure |  |

Table 486: Protection Storage Mask Structure

## Right

The Right base object is a structure that defines a right to perform specific numbers of specific operations on specific managed objects. If any field is omitted, then that aspect is unrestricted..

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Right | Structure |  |
| Usage Limits | Structure | No |
| Operations | Structure | No |
| Objects | Structure | No |
| Object Groups | Structure | No |

Table 487: Right Structure

## Rights

A list of Rights.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Rights | Structure |  |
| Right | Structure | No, May be repeated. |

Table 488: Rights Structure

## RNG Parameters

The *RNG Parameters* base object is a structure that contains a mandatory RNG Algorithm and a set of OPTIONAL fields that describe a Random Number Generator. Specific fields pertain only to certain types of RNGs.

The RNG Algorithm SHALL be specified and if the algorithm implemented is unknown or the implementation does not want to provide the specific details of the RNG Algorithm then the Unspecified enumeration SHALL be used.

If the cryptographic building blocks used within the RNG are known they MAY be specified in combination of the remaining fields within the RNG Parameters structure.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| RNG Parameters | Structure |  |
| RNG Algorithm | Enumeration | Yes |
| Cryptographic Algorithm | Enumeration | No |
| Cryptographic Length | Integer | No |
| Hashing Algorithm | Enumeration | No |
| DRBG Algorithm | Enumeration | No |
| Recommended Curve | Enumeration | No |
| FIPS186 Variation | Enumeration | No |
| Prediction Resistance | Boolean | No |

Table 489: RNG Parameters Structure

## Server Information

The *Server Information*  base object is a structure that contains a set of OPTIONAL fields that describe server information. Where a server supports returning information in a vendor-specific field for which there is an equivalent field within the structure, the server SHALL provide the standardized version of the field.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Server Information | Structure |  |
| Server name | Text String | No |
| Server serial number | Text String | No |
| Server version | Text String | No |
| Server load | Text String | No |
| Product name | Text String | No |
| Build level | Text String | No |
| Build date | Text String | No |
| Cluster info | Text String | No |
| Alternative failover endpoints | Text String, MAY be repeated | No |
| *Vendor-Specific* | *Any,* MAY be repeated | No |

Table 490: Server Information Structure

## Signature Data

The *Signature Data* is used in requests and responses in cryptographic operations that pass signature data between the client and the server.

|  |  |
| --- | --- |
| Object | Encoding |
| Signature Data | Byte String |

Table 491: Signature Data Structure

## Ticket

The ticket structure used to specify a *Ticket*

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Ticket | Structure |  |
| Ticket Type | Enumeration | Yes |
| Ticket Value | Byte String | Yes |

Table 492: Ticket Structure

## Usage Limits

The *Usage Limits* structure is used to limit the number of operations that may be performed.

|  |  |  |
| --- | --- | --- |
| Item | Encoding | REQUIRED |
| Usage Limits | Structure |  |
| Usage Limits Total | Long Integer | Yes |
| Usage Limits Count | Long Integer | Yes |
| Usage Limits Unit | Enumeration | Yes |

Table 493: Usage limits Structure

## Validation Information

The *Validation Information* base object is a structure that contains details of a formal validation. Specific fields MAY pertain only to certain types of validations.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Validation Information | Structure |  |
| Validation Authority Type | Enumeration | Yes |
| Validation Authority Country | Text String | No |
| Validation Authority URI | Text String | No |
| Validation Version Major | Integer | Yes |
| Validation Version Minor | Integer | No |
| Validation Type | Enumeration | Yes |
| Validation Level | Integer | Yes |
| Validation Certificate Identifier | Text String | No |
| Validation Certificate URI | Text String | No |
| Validation Vendor URI | Text String | No |
| Validation Profile | Text String, MAY be repeated | No |

Table 494: Validation Information Structure

The Validation Authority along with the Validation Version Major, Validation Type and Validation Level SHALL be provided to uniquely identify a validation for a given validation authority. If the Validation Certificate URI is not provided the server SHOULD include a Validation Vendor URI from which information related to the validation is available.

The Validation Authority Country is the two letter ISO country code.

# Messages

The messages in the protocol consist of a message header, one or more batch items (which contain OPTIONAL message payloads), and OPTIONAL message extensions. The message headers contain fields whose presence is determined by the protocol features used (e.g., asynchronous responses). The field contents are also determined by whether the message is a request or a response. The message payload is determined by the specific operation being requested or to which is being replied.

The message headers are structures that contain some of the following objects.

Messages contain the following objects and fields. All fields SHALL appear in the order specified.

Batched operations SHALL be executed in the order in which they appear within the request.

If the client is capable of accepting asynchronous responses, then it MAY set the Asynchronous Indicator in the header of a batched request. The batched responses MAY contain a mixture of synchronous and asynchronous responses only if the Asynchronous Indicator is present in the header.

## Requests

### Request Message

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Request Message | Structure |  |
| Request Header | Structure | Yes |
| Batch Item | Structure, MAY be repeated | Yes |

Table 495: Request Message Structure

### Request Header

|  |  |  |
| --- | --- | --- |
| Request Header | | |
| **Object** | **REQUIRED in Message** | **Comment** |
| Request Header | Yes | Structure |
| Protocol Version | Yes |  |
| Maximum Response Size | No |  |
| Client Correlation Value | No |  |
| Server Correlation Value | No |  |
| Asynchronous Indicator | No |  |
| Attestation Capable Indicator | No |  |
| Attestation Type | No, MAY be repeated |  |
| Authentication | No |  |
| Batch Error Continuation Option | No | If omitted, then Stop is assumed |
| Time Stamp | No |  |

Table 496: Request Header Structure

### Request Batch Item

|  |  |  |
| --- | --- | --- |
| **Request Batch Item** | | |
| **Object** | **REQUIRED in Message** | **Comment** |
| Batch Item | Yes | Structure |
| Operation | Yes |  |
| Ephemeral | No | Ephemeral Enumeration.  Specifies how the Response Payload should be modified prior to return to the client in terms of which fields should be omitted. |
| Request Payload | Yes | Structure, contents depend on the Operation |
| Message Extension | No, MAY be repeated |  |

Table 497: Request Batch Item Structure

## Responses

### Response Message

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Response Message | Structure |  |
| Response Header | Structure | Yes |
| Batch Item | Structure, MAY be repeated | Yes |

Table 498: Response Message Structure

### Response Header

|  |  |  |
| --- | --- | --- |
| Response Header | | |
| **Object** | **REQUIRED in Message** | **Comment** |
| Response Header | Yes | Structure |
| Protocol Version | Yes |  |
| Time Stamp | Yes |  |
| Nonce | No |  |
| Server Hashed Password | Yes, if Hashed Password credential was used | Hash(Timestamp || S1 || Hash(S2)), where S1, S2 and the Hash algorithm are defined in the Hashed Password credential.  The client MUST check this value is correct prior to otherwise using the results from the server. |
| Attestation Type | No, MAY be repeated | REQUIRED in *Attestation Required* error message if client set Attestation Capable Indicator to True in the request |
| Client Correlation Value | No |  |
| Server Correlation Value | No |  |

Table 499: Response Header Structure

### Response Batch Item

|  |  |  |
| --- | --- | --- |
| **Response Batch Item** | | |
| **Object** | **REQUIRED in Message** | **Comment** |
| Batch Item | Yes | Structure |
| Operation | Yes, if specified in Request Batch Item |  |
| Result Status | Yes |  |
| Result Reason | Yes, if Result Status is *Failure* | REQUIRED if Result Status is *Failure*, otherwise OPTIONAL |
| Result Message | No | OPTIONAL if Result Status is not *Pending* or *Success* |
| Asynchronous Correlation Value | No | REQUIRED if Result Status is *Pending* |
| Response Payload | Yes, if not a failure | Structure, contents depend on the Operation |
| Message Extension | No |  |

Table 500: Response Batch Item Structure

# Message Data Structures

Data structures passed within request and response messages.

## Asynchronous Correlation Value

This is returned in the immediate response to an operation that is pending and that requires asynchronous polling. Note: the server decides which operations are performed synchronously or asynchronously. A server-generated correlation value SHALL be specified in any subsequent Poll or Cancel operations that pertain to the original operation.

|  |  |
| --- | --- |
| Object | Encoding |
| Asynchronous Correlation Value | Byte String |

Table 501: Asynchronous Correlation Value in Response Batch Item

## Asynchronous Indicator

This Enumeration indicates whether the client is able to accept an asynchronous response. If not present in a request, then Prohibited is assumed. If the value is Prohibited, the server SHALL process the request synchronously.

|  |  |
| --- | --- |
| Object | Encoding |
| Asynchronous Indicator | Enumeration |

Table 502: Asynchronous Indicator in Message Request Header

## Attestation Capable Indicator

The *Attestation Capable Indicator* flag indicates whether the client is able to create an Attestation Credential object. It SHALL have Boolean value True if the client is able to create an Attestation Credential object, and the value False otherwise. If not present, the value False is assumed. If a client indicates that it is not able to create an Attestation Credential Object, and the client has issued an operation that requires attestation such as Get, then the server SHALL respond to the request with a failure.

|  |  |
| --- | --- |
| Object | Encoding |
| Attestation Capable Indicator | Boolean |

Table 503: Attestation Capable Indicator in Message Request Header

## Authentication

This is used to authenticate the requester. It is an OPTIONAL information item, depending on the type of request being issued and on server policies. Servers MAY require authentication on no requests, a subset of the requests, or all requests, depending on policy. Queryoperations used to interrogate server features and functions SHOULD NOT require authentication. The Authentication structure SHALL contain one or more Credential structures. If multiple Credential structures are provided then they must ALL be satisfied.

Specific authentication mechanisms are specified in [KMIP-Prof].

|  |  |
| --- | --- |
| Object | Encoding |
| Authentication | Structure |
| Credential, MAY be repeated | Structure |

Table 504: Authentication Structure in Message Header

## Batch Error Continuation Option

This option SHALL have one of three values (*Undo*, *Stop* or *Continue).* If not specified, then Stop is assumed.

|  |  |
| --- | --- |
| Object | Encoding |
| Batch Error Continuation Option | Enumeration |

Table 505: Batch Error Continuation Option in Message Request Header

## Batch Item

This field consists of a structure that holds the individual requests or responses in a batch, and is REQUIRED. The contents of the batch items are described in Sections 8.1.3 and above8.2.3.

|  |  |
| --- | --- |
| Object | Encoding |
| Batch Item | Structure |

Table 506: Batch Item in Message

## Correlation Value (Client)

The Client Correlation Value is a string that MAY be added to messages by clients to provide additional

information to the server. It need not be unique. The server SHOULD log this information.

For client to server operations, the Client Correlation Value is provided in the request. For server to client operations the Client Correlation Value is provided in the response.

|  |  |
| --- | --- |
| Object | Encoding |
| Client Correlation Value | Text String |

Table 507: Client Correlation Value in Message Request Header

## Correlation Value (Server)

The Server Correlation Value SHOULD be provided by the server and SHOULD be globally unique, and SHOULD be logged by the server with each request.

For client to server operations the Server Correlation Value is provided in the response. For server to client operations, the Server Correlation Value is provided in the request.

|  |  |
| --- | --- |
| Object | Encoding |
| Server Correlation Value | Text String |

Table 508: Server Correlation Value in Message Request Header

## Credential

A *Credential* is a structure used for client identification purposes and is not managed by the key management system (e.g., user id/password pairs, Kerberos tokens, etc.). It MAY be used for authentication purposes as indicated in [KMIP-Prof].

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Credential | Structure |  |
| Credential Type | Enumeration | Yes |
| Credential Value | Varies based on Credential Type. | Yes |

Table 509: Credential Object Structure

If the Credential Type in the Credential is *Username and Password*, then Credential Value is a structure. The Username field identifies the client, and the Password field is a secret that authenticates the client.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Credential Value | Structure |  |
| Username | Text String | Yes |
| Password | Text String | No |

Table 510: Credential Value Structure for the Username and Password Credential

If the Credential Type in the Credential is *Device*, then Credential Value is a structure. One or a combination of the *Device Serial Number*, *Network Identifier*, *Machine Identifier*, and *Media Identifier* SHALL be unique. Server implementations MAY enforce policies on uniqueness for individual fields. A shared secret or password MAY also be used to authenticate the client. The client SHALL provide at least one field.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Credential Value | Structure |  |
| Device Serial Number | Text String | No |
| Password | Text String | No |
| Device Identifier | Text String | No |
| Network Identifier | Text String | No |
| Machine Identifier | Text String | No |
| Media Identifier | Text String | No |

Table 511: Credential Value Structure for the Device Credential

If the Credential Type in the Credential is *Attestation*, then Credential Value is a structure. The *Nonce Value* is obtained from the key management server in a Nonce Object. The Attestation Credential Object can contain a measurement from the client or an assertion from a third party if the server is not capable or willing to verify the attestation data from the client. Neither type of attestation data (*Attestation Measurement* or *Attestation Assertion*) is necessary to allow the server to accept either. However, the client SHALL provide attestation data in either the *Attestation Measurement* or *Attestation Assertion* fields.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Credential Value | Structure |  |
| Nonce | Structure | Yes |
| Attestation Type | Enumeration | Yes |
| Attestation Measurement | Byte String | No |
| Attestation Assertion | Byte String | No |

Table 512: Credential Value Structure for the Attestation Credential

If the Credential Type in the Credential is One Time Password, then Credential Value is a structure. The Username field identifies the client, and the Password field is a secret that authenticates the client. The One Time Password field contains a one time password (OTP) which may only be used for a single authentication.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Credential Value | Structure |  |
| Username | Text String | Yes |
| Password | Text String | No |
| One Time Password | Text String | Yes |

Table 513: Credential Value Structure for the One Time Password Credential

If the Credential Type in the Credential is Hashed Password, then Credential Value is a structure. The Username field identifies the client. The timestamp is the current timestamp used to produce the hash and SHALL monotonically increase. The Hashing Algorithm SHALL default to SHA 256. The Hashed Password is define as

Hashed Password = Hash(S1 || Timestamp) || S2

Where

S1 = Hash(Username || Password)

S2 = Hash(Password || Username)

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Credential Value | Structure |  |
| Username | Text String | Yes |
| Timestamp | Date Time Extended | Yes |
| Hashing Algorithm | Enumeration | No |
| Hashed Password | Byte String | Yes |

Table 514: Credential Value Structure for the Hashed Password Credential

If the Credential Type in the Credential is Ticket, then Credential Value is a structure.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Credential Value | Structure |  |
| Ticket | Structure | Yes |

Table 515: Credential Value Structure for the Ticket

## Maximum Response Size

This is an OPTIONAL field contained in a request message, and is used to indicate the maximum size of a response, in bytes, that the requester SHALL be able to handle. It SHOULD only be sent in requests that possibly return large replies.

|  |  |
| --- | --- |
| Object | Encoding |
| Maximum Response Size | Integer |

Table 516: Maximum Response Size in Message Request Header

## Message Extension

The *Message Extension* is an OPTIONAL structure that MAY be appended to any Batch Item. It is used to extend protocol messages for the purpose of adding vendor-specified extensions. The Message Extension is a structure that SHALL contain the Vendor Identification, Criticality Indicator, and Vendor Extension fields. The *Vendor Identification* SHALL be a text string that uniquely identifies the vendor, allowing a client to determine if it is able to parse and understand the extension. If a client or server receives a protocol message containing a message extension that it does not understand, then its actions depend on the *Criticality Indicator*. If the indicator is True (i.e., Critical), and the receiver does not understand the extension, then the receiver SHALL reject the entire message. If the indicator is False (i.e., Non-Critical), and the receiver does not understand the extension, then the receiver MAY process the rest of the message as if the extension were not present. The *Vendor Extension* structure SHALL contain vendor-specific extensions.

|  |  |
| --- | --- |
| Object | Encoding |
| Message Extension | Structure |
| Vendor Identification | Text String (with usage limited to alphanumeric, underscore and period – i.e. [A-Za-z0-9\_.]) |
| Criticality Indicator | Boolean |
| Vendor Extension | Structure |

Table 517: Message Extension Structure in Batch Item

## Nonce

A *Nonce* object is a structure used by the server to send a random value to the client. The Nonce Identifier is assigned by the server and used to identify the Nonce object. The Nonce Value consists of the random data created by the server.

|  |  |  |
| --- | --- | --- |
| Object | Encoding | REQUIRED |
| Nonce | Structure |  |
| Nonce ID | Byte String | Yes |
| Nonce Value | Byte String | Yes |

Table 518: Nonce Structure

## Operation

This field indicates the operation being requested or the operation for which the response is being returned.

|  |  |
| --- | --- |
| Object | Encoding |
| Operation | Enumeration |

Table 519: Operation in Batch Item

## Protocol Version

This field contains the version number of the protocol, ensuring that the protocol is fully understood by both communicating parties. The version number SHALL be specified in two parts, major and minor. Servers and clients SHALL support backward compatibility with versions of the protocol with the same major version. Support for backward compatibility with different major versions is OPTIONAL.

|  |  |
| --- | --- |
| Object | Encoding |
| Protocol Version | Structure |
| Protocol Version Major | Integer |
| Protocol Version Minor | Integer |

Table 520: Protocol Version Structure in Message Header

## Result Message

This field MAY be returned in a response. It contains a more descriptive error message, which MAY be provided to an end user or used for logging/auditing purposes.

|  |  |
| --- | --- |
| Object | Encoding |
| Result Message | Text String |

Table 521: Result Message in Response Batch Item

## Result Reason

This field indicates a reason for failure or a modifier for a partially successful operation and SHALL be present in responses that return a Result Status of Failure. In such a case, the Result Reason SHALL be set as specified. It SHALL NOT be present in any response that returns a Result Status of Success.

|  |  |
| --- | --- |
| Object | Encoding |
| Result Reason | Enumeration |

Table 522: Result Reason in Response Batch Item

## Result Status

This is sent in a response message and indicates the success or failure of a request. The following values MAY be set in this field:

* *Success* – The requested operation completed successfully.
* *Operation Pending* – The requested operation is in progress, and it is necessary to obtain the actual result via asynchronous polling. The asynchronous correlation value SHALL be used for the subsequent polling of the result status.
* *Operation Undone* – The requested operation was performed, but had to be undone (i.e., due to a failure in a batch for which the Error Continuation Option was set to Undo).
* *Operation Failed* – The requested operation failed.

|  |  |
| --- | --- |
| Object | Encoding |
| Result Status | Enumeration |

Table 523: Result Status in Response Batch Item

## Time Stamp

This is an OPTIONAL field contained in a client request. It is REQUIRED in a server request and response. It is used for time stamping, and MAY be used to enforce reasonable time usage at a client (e.g., a server MAY choose to reject a request if a client's time stamp contains a value that is too far off the server’s time). Note that the time stamp MAY be used by a client that has no real-time clock, but has a countdown timer, to obtain useful “seconds from now” values from all of the Date attributes by performing a subtraction.

|  |  |
| --- | --- |
| Object | Encoding |
| Time Stamp | Date-Time |

Table 524: Time Stamp in Message Header

# Message Protocols

## TTLV

In order to minimize the resource impact on potentially low-function clients, one encoding mechanism to be used for protocol messages is a simplified TTLV (Tag, Type, Length, Value) scheme.

The scheme is designed to minimize the CPU cycle and memory requirements of clients that need to encode or decode protocol messages, and to provide optimal alignment for both 32-bit and 64-bit processors. Minimizing bandwidth over the transport mechanism is considered to be of lesser importance.

### Tag

An Item Tag is a three-byte binary unsigned integer, transmitted big endian, which contains the *Tag* *Enumeration Value (using only the three least significant bytes of the enumeration)*.

### Type

An Item Type is a byte containing a coded value that indicates the data type of the data object using the specified *Item Type Enumeration* (using only the least significant byte of the enumeration).

|  |  |
| --- | --- |
| **Value** | **Description** |
| Structure | Encoded as the concatenated encodings of the elements of the structure. All structures defined in this specification SHALL have all of their fields encoded in the order in which they appear in their respective structure descriptions |
| Integer | Encoded as four-byte long (32 bit) binary signed numbers in 2's complement notation, transmitted big-endian. |
| Long Integer | Encoded as eight-byte long (64 bit) binary signed numbers in 2's complement notation, transmitted big-endian. |
| Big Integer | Encoded as a sequence of eight-bit bytes, in two's complement notation, transmitted big-endian. If the length of the sequence is not a multiple of eight bytes, then Big Integers SHALL be padded with the minimal number of leading sign-extended bytes to make the length a multiple of eight bytes. These padding bytes are part of the Item Value and SHALL be counted in the Item Length. |
| Enumeration | Encoded as four-byte long (32 bit) binary unsigned numbers transmitted big-endian. Extensions, which are permitted, but are not defined in this specification, contain the value 8 hex in the first nibble of the first byte. |
| Boolean | Encoded as an eight-byte hex value 0000000000000000, indicating the Boolean value False, or the hex value 0000000000000001, indicating the Boolean value True, transmitted big-endian. |
| Text String | Sequences of bytes that encode character values according to **[RFC3629]** the UTF-8 encoding standard. |
| Byte String | Sequences of bytes containing individual eight-bit binary values. |
| Date Time | Encoded as eight-byte long (64 bit) binary signed numbers in 2's complement notation, transmitted big-endian. |
| Interval | Encoded as four-byte long (32 bit) binary unsigned numbers, transmitted big-endian. |
| Date Time Extended | Encoded as eight-byte long (64 bit) binary signed numbers in 2's complement notation, transmitted big-endian. |
| Identifier | Sequences of bytes that encode character values according to  **[RFC3629]** the UTF-8 encoding standard. |
| Reference | Sequences of bytes that encode character values according to  **[RFC3629]** the UTF-8 encoding standard. |
| Name Reference | Sequences of bytes that encode character values according to **[RFC3629]** the UTF-8 encoding standard. |

Table 525: Item Types

### Length

An Item Length is a 32-bit binary integer, transmitted big-endian, containing the number of bytes in the Item Value. The allowed values are:

|  |  |
| --- | --- |
| **Data Type** | **Length** |
| Structure | Varies, multiple of 8 |
| Integer | 4 |
| Long Integer | 8 |
| Big Integer | Varies, multiple of 8 |
| Enumeration | 4 |
| Boolean | 8 |
| Text String | Varies |
| Byte String | Varies |
| Date Time | 8 |
| Interval | 4 |
| Date Time Extended | 8 |
| Identifier | Varies |
| Reference | Varies |
| Name Reference | Varies |

Table 526: Allowed Item Length Values

### Value

The item value is a sequence of bytes containing the value of the data item, depending on the type.

### Padding

If the Item Type is Structure, then the Item Length is the total length of all of the sub-items contained in the structure, including any padding. If the Item Type is Integer, Enumeration, Text String, Byte String, Identifier, Reference, Name Reference, or Interval, then the Item Length is the number of bytes excluding the padding bytes. Text Strings, Byte Strings, Identifiers, References and Name References SHALL be padded with the minimal number of bytes following the Item Value to obtain a multiple of eight bytes. Integers, Enumerations, and Intervals SHALL be padded with four bytes following the Item Value.

## Other Message Protocols

In addition to the mandatory TTLV messaging protocol, a number of optional message-encoding mechanisms to support different transport protocols and different client capabilities.

### HTTPS

The HTTPs messaging protocol is specified in [KMIP-Prof].

### JSON

The JSON messaging protocol is specified in [KMIP-Prof].

### XML

The XML messaging protocol is specified in [KMIP-Prof].

## Authentication

The mechanisms used to authenticate the client to the server and the server to the client are not part of the message definitions, and are external to the protocol. The KMIP Server SHALL support authentication as defined in [KMIP-Prof].

## Transport

KMIP Servers and Clients SHALL establish and maintain channel confidentiality and integrity, and provide assurance of authenticity for KMIP messaging as specified in [KMIP-Prof].

# Enumerations

The following tables define the values for enumerated lists. Values not listed (outside the range 80000000 to 8FFFFFFF) are reserved for future KMIP versions.

Implementations SHALL NOT use Tag Values marked as Reserved.

## Adjustment Type Enumeration

The *Adjustment Type* enumerations are:

|  |  |
| --- | --- |
| **Value** | **Description** |
| Increment | Add the Adjustment Parameter to the value. Applies to Integer, Long Integers, Big Integer, Interval, Date Time, and Date Time Extended. The default is parameter is 1 for numeric types, 1 second for Date Time, and 1 microsecond for Date Time Extended. |
| Decrement | Subtract the Adjustment Parameter to the value. Applies to Integer, Long Integers, Big Integer, Interval, Date Time, and Date Time Extended. The default is parameter is 1 for numeric types, 1 second for Date Time, and 1 microsecond for Date Time Extended. |
| Negate | Negate the value. Applies to Integer, Long Integers, Big Integer and Boolean types. |

Table 527: Adjustment Type Descriptions

|  |  |
| --- | --- |
| **Adjustment Type** | |
| **Name** | **Value** |
| Increment | 00000001 |
| Decrement | 00000002 |
| Negate | 00000003 |
| Extensions | 8XXXXXXX |

Table 528: Adjustment Type Enumeration

## Alternative Name Type Enumeration

|  |  |
| --- | --- |
| **Alternative Name Type** | |
| **Name** | **Value** |
| Uninterpreted Text String | 00000001 |
| URI | 00000002 |
| Object Serial Number | 00000003 |
| Email Address | 00000004 |
| DNS Name | 00000005 |
| X.500 Distinguished Name | 00000006 |
| IP Address | 00000007 |
| Extensions | 8XXXXXXX |

Table 529: Alternative Name Type Enumeration

## Asynchronous Indicator Enumeration

*Asynchronous Indicator* enumerations are:

|  |  |
| --- | --- |
| **Value** | **Description** |
| Mandatory | The server SHALL process all batch items in the request asynchronously (returning an Asynchronous Correlation Value for each batch item). |
| Optional | The server MAY process each batch item in the request either asynchronously (returning an Asynchronous Correlation Value for a batch item) or synchronously.  The method or policy by which the server determines whether or not to process an individual batch item asynchronously is a decision of the server and is outside of the scope of this protocol. |
| Prohibited | The server SHALL NOT process any batch item asynchronously. All batch items SHALL be processed synchronously. |

Table 530: Asynchronous Indicator Descriptions

|  |  |
| --- | --- |
| **Asynchronous Indicator** | |
| **Name** | **Value** |
| Mandatory | 00000001 |
| Optional | 00000002 |
| Prohibited | 00000003 |
| Extensions | 8XXXXXXX |

Table 531: Asynchronous Indicator Enumeration

## Attestation Type Enumeration

|  |  |
| --- | --- |
| **Attestation Type** | |
| **Name** | **Value** |
| TPM Quote | 00000001 |
| TCG Integrity Report | 00000002 |
| SAML Assertion | 00000003 |
| Extensions | 8XXXXXXX |

Table 532: Attestation Type Enumeration

## Batch Error Continuation Option Enumeration

*Batch Error Continuation Option* enumerations are:

|  |  |
| --- | --- |
| **Value** | **Description** |
| Undo | If any operation in the request fails, then the server SHALL undo all the previous operations.  Batch item fails and Result Status is set to Operation Failed. Responses to batch items that have already been processed are returned normally. Responses to batch items that have not been processed are not returned. |
| Stop | If an operation fails, then the server SHALL NOT continue processing subsequent operations in the request. Completed operations SHALL NOT be undone.  Batch item fails and Result Status is set to Operation Failed. Responses to other batch items are returned normally. |
| Continue | Return an error for the failed operation, and continue processing subsequent operations in the request.  Batch item fails and Result Status is set to Operation Failed. Batch items that had been processed have been undone and their responses are returned with Undone result status. |

Table 533: Batch Error Continuation Option Descriptions

|  |  |
| --- | --- |
| **Batch Error Continuation** | |
| **Name** | **Value** |
| Continue | 00000001 |
| Stop | 00000002 |
| Undo | 00000003 |
| Extensions | 8XXXXXXX |

Table 534: Batch Error Continuation Option Enumeration

## Block Cipher Mode Enumeration

|  |  |
| --- | --- |
| **Block Cipher Mode** | |
| **Name** | **Value** |
| CBC | 00000001 |
| ECB | 00000002 |
| PCBC | 00000003 |
| CFB | 00000004 |
| OFB | 00000005 |
| CTR | 00000006 |
| CMAC | 00000007 |
| CCM | 00000008 |
| GCM | 00000009 |
| CBC-MAC | 0000000A |
| XTS | 0000000B |
| AESKeyWrapPadding | 0000000C |
| NISTKeyWrap | 0000000D |
| X9.102 AESKW | 0000000E |
| X9.102 TDKW | 0000000F |
| X9.102 AKW1 | 00000010 |
| X9.102 AKW2 | 00000011 |
| AEAD | 00000012 |
| Extensions | 8XXXXXXX |

Table 535: Block Cipher Mode Enumeration

## Cancellation Result Enumeration

A *Cancellation Result* enumerations are:

|  |  |
| --- | --- |
| **Value** | **Description** |
| Canceled | The cancel operation succeeded in canceling the pending operation. |
| Unable to Cancel | The cancel operation is unable to cancel the pending operation. |
| Completed | The pending operation completed successfully before the cancellation operation was able to cancel it. |
| Failed | The pending operation completed with a failure before the cancellation operation was able to cancel it. |
| Unavailable | Unavailable – The specified correlation value did not match any recently pending or completed asynchronous operations. |

Table 536: Cancellation Result Enumeration Descriptions

|  |  |
| --- | --- |
| **Cancellation Result** | |
| **Name** | **Value** |
| Canceled | 00000001 |
| Unable to Cancel | 00000002 |
| Completed | 00000003 |
| Failed | 00000004 |
| Unavailable | 00000005 |
| Extensions | 8XXXXXXX |

Table 537: Cancellation Result Enumeration

## Certificate Request Type Enumeration

|  |  |
| --- | --- |
| **Certificate Request Type** | |
| **Name** | **Value** |
| CRMF | 00000001 |
| PKCS#10 | 00000002 |
| PEM | 00000003 |
| (Reserved) | 00000004 |
| Extensions | 8XXXXXXX |

Table 538: Certificate Request Type Enumeration

## Certificate Type Enumeration

|  |  |
| --- | --- |
| **Certificate Type** | |
| **Name** | **Value** |
| X.509 | 00000001 |
| PGP | 00000002 |
| Extensions | 8XXXXXXX |

Table 539: Certificate Type Enumeration

## Client Registration Method Enumeration

*Client Registration Method* enumerations are:

|  |  |
| --- | --- |
| **Value** | **Description** |
| Server Pre-Generated | The server has pre-generated the client’s private key. The returned PKCS#12 is protected with HEX(SHA256(Username || Password)). |
| Server On-Demand | The server generates the client’s private key on demand. The returned PKCS#12 is protected with HEX(SHA256(Username || Password)). |
| Client Generated | The client generates the private key and sends a Certificate Signing Request to the server to generate the certificate. The returned PKCS#12 is protected with HEX(SHA256(Username || Password)). |
| Client Registered | The client generates the private key and the certificates and registers the certificate with the server. |

Table 540: Client Registration Method Enumeration Descriptions

|  |  |
| --- | --- |
| **Client Registration Method** | |
| **Name** | **Value** |
| Unspecified | 00000001 |
| Server Pre-Generated | 00000002 |
| Server On-Demand | 00000003 |
| Client Generated | 00000004 |
| Client Registered | 00000005 |
| Extensions | 8XXXXXXX |

Table 541: Client Registration Method Enumerations

## Credential Type Enumeration

|  |  |
| --- | --- |
| **Credential Type** | |
| **Name** | **Value** |
| Username and Password | 00000001 |
| Device | 00000002 |
| Attestation | 00000003 |
| One Time Password | 00000004 |
| Hashed Password | 00000005 |
| Ticket | 00000006 |
| Password | 00000007 |
| Certificate | 00000008 |
| Extensions | 8XXXXXXX |

Table 542: Credential Type Enumeration

## Cryptographic Algorithm Enumeration

|  |  |
| --- | --- |
| **Cryptographic Algorithm** | |
| **Name** | **Value** |
| DES | 00000001 |
| 3DES | 00000002 |
| AES | 00000003 |
| RSA | 00000004 |
| DSA | 00000005 |
| ECDSA | 00000006 |
| HMAC-SHA1 | 00000007 |
| HMAC-SHA224 | 00000008 |
| HMAC-SHA256 | 00000009 |
| HMAC-SHA384 | 0000000A |
| HMAC-SHA512 | 0000000B |
| HMAC-MD5 | 0000000C |
| DH | 0000000D |
| ECDH | 0000000E |
| ECMQV | 0000000F |
| Blowfish | 00000010 |
| Camellia | 00000011 |
| CAST5 | 00000012 |
| IDEA | 00000013 |
| MARS | 00000014 |
| RC2 | 00000015 |
| RC4 | 00000016 |
| RC5 | 00000017 |
| SKIPJACK | 00000018 |
| Twofish | 00000019 |
| EC | 0000001A |
| One Time Pad | 0000001B |
| ChaCha20 | 0000001C |
| Poly1305 | 0000001D |
| ChaCha20Poly1305 | 0000001E |
| SHA3-224 | 0000001F |
| SHA3-256 | 00000020 |
| SHA3-384 | 00000021 |
| SHA3-512 | 00000022 |
| HMAC-SHA3-224 | 00000023 |
| HMAC-SHA3-256 | 00000024 |
| HMAC-SHA3-384 | 00000025 |
| HMAC-SHA3-512 | 00000026 |
| SHAKE-128 | 00000027 |
| SHAKE-256 | 00000028 |
| ARIA | 00000029 |
| SEED | 0000002A |
| SM2 | 0000002B |
| SM3 | 0000002C |
| SM4 | 0000002D |
| GOST R 34.10-2012 | 0000002E |
| GOST R 34.11-2012 | 0000002F |
| GOST R 34.13-2015 | 00000030 |
| GOST 28147-89 | 00000031 |
| XMSS | 00000032 |
| SPHINCS-256 | 00000033 |
| McEliece | 00000034 |
| McEliece-6960119 | 00000035 |
| McEliece-8192128 | 00000036 |
| Ed25519 | 00000037 |
| Ed448 | 00000038 |
| ML-KEM-512 | 00000039 |
| ML-KEM-768 | 0000003A |
| ML-KEM-1024 | 0000003B |
| ML-DSA-44 | 0000003C |
| ML-DSA-65 | 0000003D |
| ML-DSA-87 | 0000003E |
| SLH-DSA-SHA2-128s | 0000003F |
| SLH-DSA-SHA2-128f | 00000040 |
| SLH-DSA-SHA2-192s | 00000041 |
| SLH-DSA-SHA2-192f | 00000042 |
| SLH-DSA-SHA2-256s | 00000043 |
| SLH-DSA-SHA2-256f | 00000044 |
| SLH-DSA-SHAKE-128s | 00000045 |
| SLH-DSA-SHAKE-128f | 00000046 |
| SLH-DSA-SHAKE-192s | 00000047 |
| SLH-DSA-SHAKE-192f | 00000048 |
| SLH-DSA-SHAKE-256s | 00000049 |
| SLH-DSA-SHAKE-256f | 0000004A |
| Extensions | 8XXXXXXX |

Table 543: Cryptographic Algorithm Enumeration

## Data Enumeration

|  |  |
| --- | --- |
| **Data** | |
| **Name** | **Value** |
| Decrypt | 00000001 |
| Encrypt | 00000002 |
| Hash | 00000003 |
| MAC MAC Data | 00000004 |
| RNG Retrieve | 00000005 |
| Sign Signature Data | 00000006 |
| Signature Verify | 00000007 |
| Extensions | 8XXXXXXX |

Table 544: Data Enumeration

## Deactivation Reason Code Enumeration

|  |  |
| --- | --- |
| **Deactivation Reason Code** | |
| **Name** | **Value** |
| Unspecified | 00000001 |
| Deactivation Date | 00000002 |
| Protect Stop Date | 00000003 |
| Usage Limit | 00000004 |
| Extensions | 8XXXXXXX |

Table 545: Deactivation Reason Code Enumeration

## Derivation Method Enumeration

The *Derivation Method* enumerations are:

|  |  |  |
| --- | --- | --- |
| **Item** | **Description** | **Mapping** |
| PBKDF2 | This method is used to derive a symmetric key from a password or pass phrase. | [PKCS#5] and [RFC2898] |
| HASH | This method derives a key by computing a hash over the derivation key or the derivation data. |  |
| HMAC | This method derives a key by computing an HMAC over the derivation data. |  |
| ENCRYPT | This method derives a key by encrypting the derivation data. |  |
| NIST800-108-C | This method derives a key by computing the KDF in Counter Mode | [SP800-108] |
| NIST800-108-F | This method derives a key by computing the KDF in Feedback Mode | [SP800-108] |
| NIST800-108-DPI | This method derives a key by computing the KDF in Double-Pipeline Iteration Mode | [SP800-108] |
| Asymmetric Key | This method derives a key using asymmetric key agreement between a private and public key. |  |
| AWS Signature Version 4 | As defined in Amazon Web Services Signature Version 4. | **[AWS-SIGV4]** |
| HKDF | HMAC-based Extract-and-Expand Key Derivation Function | **[RFC5869]** |

Table 546: Derivation Method Enumeration Descriptions

|  |  |
| --- | --- |
| **Derivation Method** | |
| **Name** | **Value** |
| PBKDF2 | 00000001 |
| HASH | 00000002 |
| HMAC | 00000003 |
| ENCRYPT | 00000004 |
| NIST800-108-C | 00000005 |
| NIST800-108-F | 00000006 |
| NIST800-108-DPI | 00000007 |
| Asymmetric Key | 00000008 |
| AWS Signature Version 4 | 00000009 |
| HKDF | 0000000A |
| Extensions | 8XXXXXXX |

Table 547: Derivation Method Enumeration

## Destroy Action Enumeration

|  |  |
| --- | --- |
| **Destroy Action Type** | |
| **Name** | **Value** |
| Unspecified | 00000001 |
| Key Material Deleted | 00000002 |
| Key Material Shredded | 00000003 |
| Meta Data Deleted | 00000004 |
| Meta Data Shredded | 00000005 |
| Deleted | 00000006 |
| Shredded | 00000007 |
| Extensions | 8XXXXXXX |

Table 548: Destroy Action Enumeration

## Digital Signature Algorithm Enumeration

|  |  |
| --- | --- |
| **Digital Signature Algorithm** | |
| **Name** | **Value** |
| MD2 with RSA Encryption | 00000001 |
| MD5 with RSA Encryption | 00000002 |
| SHA-1 with RSA Encryption | 00000003 |
| SHA-224 with RSA Encryption | 00000004 |
| SHA-256 with RSA Encryption | 00000005 |
| SHA-384 with RSA Encryption | 00000006 |
| SHA-512 with RSA Encryption | 00000007 |
| RSASSA-PSS | 00000008 |
| DSA with SHA-1 | 00000009 |
| DSA with SHA224 | 0000000A |
| DSA with SHA256 | 0000000B |
| ECDSA with SHA-1 | 0000000C |
| ECDSA with SHA224 | 0000000D |
| ECDSA with SHA256 | 0000000E |
| ECDSA with SHA384 | 0000000F |
| ECDSA with SHA512 | 00000010 |
| SHA3-256 with RSA Encryption | 00000011 |
| SHA3-384 with RSA Encryption | 00000012 |
| SHA3-512 with RSA Encryption | 00000013 |
| Extensions | 8XXXXXXX |

Table 549: Digital Signature Algorithm Enumeration

## DRBG Algorithm Enumeration

|  |  |
| --- | --- |
| **DRBG Algorithm** | |
| **Name** | **Value** |
| Unspecified | 00000001 |
| Dual-EC | 00000002 |
| Hash | 00000003 |
| HMAC | 00000004 |
| CTR | 00000005 |
| Extensions | 8XXXXXXX |

Table 550: DRGB Algorithm Enumeration

## Encoding Option Enumeration

The following encoding options are currently defined:

|  |  |
| --- | --- |
| **Value** | **Description** |
| No Encoding | the wrapped un-encoded value of the Byte String Key Material field in the Key Value structure |
| TTLV Encoding | the wrapped TTLV-encoded Key Value structure |

Table 551: Encoding Option Description

|  |  |
| --- | --- |
| **Encoding Option** | |
| **Name** | **Value** |
| No Encoding | 00000001 |
| TTLV Encoding | 00000002 |
| Extensions | 8XXXXXXX |

Table 552: Encoding Option Enumeration

## Endpoint Role Enumeration

The following endpoint roles are currently defined:

|  |  |
| --- | --- |
| **Value** | **Description** |
| Client | The endpoint that sends requests and receives responses. |
| Server | The endpoint that receives requests and sends responses. |

Table 553: Endpoint Role Description

|  |  |
| --- | --- |
| **Encoding Option** | |
| **Name** | **Value** |
| Client | 00000001 |
| Server | 00000002 |
| Extensions | 8XXXXXXX |

Table 554: Endpoint Role Enumeration

## Ephemeral Enumeration

The following ephemeral options are currently defined:

|  |  |
| --- | --- |
| **Value** | **Description** |
| Data | Only the Data tag is omitted from the Response Payload |
| Empty | The Response Payload is returned as empty (all fields are omitted) |
| Unique Identifier | All fields in the Response Payload other than the Unique Identifier are omitted. |

Table 555: Ephemeral Description

|  |  |
| --- | --- |
| **Ephemeral** | |
| **Name** | **Value** |
| Data | 00000001 |
| Empty | 00000002 |
| Unique Identifier | 00000003 |
| Extensions | 8XXXXXXX |

Table 556 Ephemeral Enumeration

## FIPS186 Variation Enumeration

|  |  |
| --- | --- |
| **FIPS186 Variation** | |
| **Name** | **Value** |
| Unspecified | 00000001 |
| GP x-Original | 00000002 |
| GP x-Change Notice | 00000003 |
| x-Original | 00000004 |
| x-Change Notice | 00000005 |
| k-Original | 00000006 |
| k-Change Notice | 00000007 |
| Extensions | 8XXXXXXX |

Table 557: FIPS186 Variation Enumeration

Note: the user should be aware that a number of these algorithms are no longer recommended for general use and/or are deprecated. They are included for completeness.

## Hashing Algorithm Enumeration

|  |  |
| --- | --- |
| **Hashing Algorithm** | |
| **Name** | **Value** |
| MD2 | 00000001 |
| MD4 | 00000002 |
| MD5 | 00000003 |
| SHA-1 | 00000004 |
| SHA-224 | 00000005 |
| SHA-256 | 00000006 |
| SHA-384 | 00000007 |
| SHA-512 | 00000008 |
| RIPEMD-160 | 00000009 |
| Tiger | 0000000A |
| Whirlpool | 0000000B |
| SHA-512/224 | 0000000C |
| SHA-512/256 | 0000000D |
| SHA3-224 | 0000000E |
| SHA3-256 | 0000000F |
| SHA3-384 | 00000010 |
| SHA3-512 | 00000011 |
| Extensions | 8XXXXXXX |

Table 558: Hashing Algorithm Enumeration

## Interop Function Enumeration

*Interop Function* enumerations are:

|  |  |
| --- | --- |
| Function | Description |
| Begin | A specified test is about to begin |
| End | A specified test has ended |
| Reset | Resets the server to the state it would be in at the beginning of an interop session |

*Table 559: Interop Function Descriptions*

|  |  |
| --- | --- |
| **Interop Function** | |
| **Name** | **Value** |
| Begin | 00000001 |
| End | 00000002 |
| Reset | 00000003 |
| Extensions | 8XXXXXXX |

Table 560: Interop Function Enumeration

## Item Type Enumeration

*Item Type* enumerations are:

|  |  |
| --- | --- |
| **Value** | **Description** |
| Structure | The ordered concatenation of items. |
| Integer | Four-byte long (32 bit) signed numbers |
| Long Integer | Eight-byte long (64 bit) signed numbers. |
| Big Integer | A sequence of eight-bit bytes |
| Enumeration | Four-byte long (32 bit) unsigned numbers |
| Boolean | The value True or False. |
| Text String | Sequences of character values. |
| Byte String | Sequences of bytes containing individual unspecified eight-bit binary values |
| Date Time | Eight-byte long (64 bit) POSIX Time values in seconds. . |
| Interval | Four-byte long (32 bit) unsigned numbers in seconds |
| Date Time Extended | Eight-byte long (64 bit) POSIX Time values in micro-seconds. |
| Identifier | Sequences of character values. |
| Reference | Sequences of character values. |
| Name Reference | Sequence of character values. |

*Table 561: Item Type Descriptions*

|  |  |
| --- | --- |
| **Item Type** | |
| **Name** | **Value** |
| Structure | 00000001 |
| Integer | 00000002 |
| Long Integer | 00000003 |
| Big Integer | 00000004 |
| Enumeration | 00000005 |
| Boolean | 00000006 |
| Text String | 00000007 |
| Byte String | 00000008 |
| Date Time | 00000009 |
| Interval | 0000000A |
| Date Time Extended | 0000000B |
| Identifier | 0000000C |
| Reference | 0000000D |
| Name Reference | 0000000E |

Table 562: Item Type Enumeration

## Key Compression Type Enumeration

|  |  |
| --- | --- |
| **Key Compression Type** | |
| **Name** | **Value** |
| EC Public Key Type Uncompressed | 00000001 |
| EC Public Key Type X9.62 Compressed Prime | 00000002 |
| EC Public Key Type X9.62 Compressed Char2 | 00000003 |
| EC Public Key Type X9.62 Hybrid | 00000004 |
| Extensions | 8XXXXXXX |

Table 563: Key Compression Type Enumeration values

## Key Format Type Enumeration

A *Key Block* contains a Key Value of one of the following *Key Format Types*:

|  |  |
| --- | --- |
| **Value** | **Description** |
| Raw | A key that contains only cryptographic key material, encoded as a string of bytes. |
| Opaque | an encoded key for which the encoding is unknown to the key management system. It is encoded as a string of bytes. |
| PKCS1 | an encoded private key, expressed as a DER-encoded ASN.1 PKCS#1 object. |
| PKCS8 | An encoded private key, expressed as a DER-encoded ASN.1 PKCS#8 object, supporting both the RSAPrivateKey syntax and EncryptedPrivateKey |
| X.509 | An encoded object, expressed as a DER-encoded ASN.1 X.509 object. |
| ECPrivateKey | An ASN.1 encoded elliptic curve private key. |
| Several Transparent Key types | algorithm-specific structures containing defined values for the various key types. |
| Extensions | Vendor-specific extensions to allow for proprietary or legacy key formats. |

Table 564: Key Format Types Description

|  |  |
| --- | --- |
| **Key Format Type** | |
| **Name** | **Value** |
| Raw | 00000001 |
| Opaque | 00000002 |
| PKCS#1 | 00000003 |
| PKCS#8 | 00000004 |
| X.509 | 00000005 |
| ECPrivateKey | 00000006 |
| Transparent Symmetric Key | 00000007 |
| Transparent DSA Private Key | 00000008 |
| Transparent DSA Public Key | 00000009 |
| Transparent RSA Private Key | 0000000A |
| Transparent RSA Public Key | 0000000B |
| Transparent DH Private Key | 0000000C |
| Transparent DH Public Key | 0000000D |
| (Reserved) | 0000000E |
| (Reserved) | 0000000F |
| (Reserved) | 00000010 |
| (Reserved) | 00000011 |
| (Reserved) | 00000012 |
| (Reserved) | 00000013 |
| Transparent EC Private Key | 00000014 |
| Transparent EC Public Key | 00000015 |
| PKCS#12 | 00000016 |
| PKCS#10 | 00000017 |
| Extensions | 8XXXXXXX |

Table 565: Key Format Type Enumeration

## Key Role Type Enumeration

|  |  |
| --- | --- |
| Key Role Type | |
| **Name** | **Value** |
| BDK | 00000001 |
| CVK | 00000002 |
| DEK | 00000003 |
| MKAC | 00000004 |
| MKSMC | 00000005 |
| MKSMI | 00000006 |
| MKDAC | 00000007 |
| MKDN | 00000008 |
| MKCP | 00000009 |
| MKOTH | 0000000A |
| KEK | 0000000B |
| MAC16609 | 0000000C |
| MAC97971 | 0000000D |
| MAC97972 | 0000000E |
| MAC97973 | 0000000F |
| MAC97974 | 00000010 |
| MAC97975 | 00000011 |
| ZPK | 00000012 |
| PVKIBM | 00000013 |
| PVKPVV | 00000014 |
| PVKOTH | 00000015 |
| DUKPT | 00000016 |
| IV | 00000017 |
| TRKBK | 00000018 |
| Extensions | 8XXXXXXX |

Table 566: Key Role Type Enumeration

Note that while the set and definitions of key role types are chosen to match [X9 TR-31] there is no necessity to match binary representations.

## Key Value Location Type Enumeration

|  |  |
| --- | --- |
| **Key Value Location Type** | |
| **Name** | **Value** |
| Uninterpreted Text String | 00000001 |
| URI | 00000002 |
| Extensions | 8XXXXXXX |

Table 567: Key Value Location Type Enumeration

## Key Wrap Type Enumeration

|  |  |
| --- | --- |
| **Key Wrap Type** | |
| **Name** | **Value** |
| Not Wrapped | 00000001 |
| As Registered | 00000002 |
| Extensions | 8XXXXXXX |

Table 568: Key Wrap Enumeration

## Mask Generator Enumeration

|  |  |
| --- | --- |
| **Mask Generator** | |
| **Name** | **Value** |
| MFG1 | 00000001 |
| Extensions | 8XXXXXXX |

Table 569: Name Type Enumeration

## NIST Key Type Enumeration

|  |  |
| --- | --- |
| **NIST Key Type Enumeration** | |
| **Name** | **Value** |
| Private signature key | 00000001 |
| Public signature verification key | 00000002 |
| Symmetric authentication key | 00000003 |
| Private authentication key | 00000004 |
| Public authentication key | 00000005 |
| Symmetric data encryption key | 00000006 |
| Symmetric key wrapping key | 00000007 |
| Symmetric random number generation key | 00000008 |
| Symmetric master key | 00000009 |
| Private key transport key | 0000000A |
| Public key transport key | 0000000B |
| Symmetric key agreement key | 0000000C |
| Private static key agreement key | 0000000D |
| Public static key agreement key | 0000000E |
| Private ephemeral key agreement key | 0000000F |
| Public ephemeral key agreement key | 00000010 |
| Symmetric authorization key | 00000011 |
| Private authorization key | 00000012 |
| Public authorization key | 00000013 |
| Extensions | 8XXXXXXX |

Table 570: NIST Key Type Enumeration

## Object Class Enumeration

|  |  |
| --- | --- |
| **Object Class** | |
| **Name** | **Value** |
| User | 00000001 |
| System | 00000002 |
| Extensions | 8XXXXXXX |

Table 571: Object Class Enumeration

## Object Type Enumeration

|  |  |
| --- | --- |
| **Object Type** | |
| **Name** | **Value** |
| Certificate | 00000001 |
| Symmetric Key | 00000002 |
| Public Key | 00000003 |
| Private Key | 00000004 |
| Split Key | 00000005 |
| (Reserved) | 00000006 |
| Secret Data | 00000007 |
| Opaque Object | 00000008 |
| PGP Key | 00000009 |
| Certificate Request | 0000000A |
| User | 0000000B |
| Group | 0000000C |
| Password Credential | 0000000D |
| Device Credential | 0000000E |
| One Time Password Credential | 0000000F |
| Hashed Password Credential | 00000010 |
| Extensions | 8XXXXXXX |

Table 572: Object Type Enumeration

## Opaque Data Type Enumeration

|  |  |
| --- | --- |
| **Opaque Data Type** | |
| **Name** | **Value** |
| Extensions | 8XXXXXXX |

Table 573: Opaque Data Type Enumeration

## Operation Enumeration

|  |  |
| --- | --- |
| **Operation** | |
| **Name** | **Value** |
| Create | 00000001 |
| Create Key Pair | 00000002 |
| Register | 00000003 |
| Re-key | 00000004 |
| Derive Key | 00000005 |
| Certify | 00000006 |
| Re-certify | 00000007 |
| Locate | 00000008 |
| Check | 00000009 |
| Get | 0000000A |
| Get Attributes | 0000000B |
| Get Attribute List | 0000000C |
| Add Attribute | 0000000D |
| Modify Attribute | 0000000E |
| Delete Attribute | 0000000F |
| Obtain Lease | 00000010 |
| Get Usage Allocation | 00000011 |
| Activate | 00000012 |
| Revoke | 00000013 |
| Destroy | 00000014 |
| Archive | 00000015 |
| Recover | 00000016 |
| Validate | 00000017 |
| Query | 00000018 |
| Cancel | 00000019 |
| Poll | 0000001A |
| Notify | 0000001B |
| Put | 0000001C |
| Re-key Key Pair | 0000001D |
| Discover Versions | 0000001E |
| Encrypt | 0000001F |
| Decrypt | 00000020 |
| Sign | 00000021 |
| Signature Verify | 00000022 |
| MAC | 00000023 |
| MAC Verify | 00000024 |
| RNG Retrieve | 00000025 |
| RNG Seed | 00000026 |
| Hash | 00000027 |
| Create Split Key | 00000028 |
| Join Split Key | 00000029 |
| Import | 0000002A |
| Export | 0000002B |
| Log | 0000002C |
| Login | 0000002D |
| Logout | 0000002E |
| Delegated Login | 0000002F |
| Adjust Attribute | 00000030 |
| Set Attribute | 00000031 |
| Set Endpoint Role | 00000032 |
| PKCS#11 | 00000033 |
| Interop | 00000034 |
| Re-Provision | 00000035 |
| Set Defaults | 00000036 |
| Set Constraints | 00000037 |
| Get Constraints | 00000038 |
| Query Asynchronous Requests | 00000039 |
| Process | 0000003A |
| Ping | 0000003B |
| Create Group | 0000003C |
| Obliterate | 0000003D |
| Create User | 0000003E |
| Create Credential | 0000003F |
| Deactivate | 00000040 |
| Extensions | 8XXXXXXX |

Table 574: Operation Enumeration

## OTP Algorithm Enumeration

The following One-Time Password algorithms are currently defined:

|  |  |
| --- | --- |
| **Value** | **Description** |
| HOTP | HMAC-Based One-Time Password Algorithm **[RFC4226]** |
| TOTP | Time-Based One-Time Password Algorithm **[RFC6238]** |

Table 575: OTP Algorithm Description

|  |  |
| --- | --- |
| **OTP Algorithm** | |
| **Name** | **Value** |
| HOTP | 00000001 |
| TOTP | 00000002 |
| Extensions | 8xxxxxxx |

Table 576: OTP Algorithm Enumeration

## Padding Method Enumeration

|  |  |
| --- | --- |
| **Padding Method** | |
| **Name** | **Value** |
| None | 00000001 |
| OAEP | 00000002 |
| PKCS5 | 00000003 |
| SSL3 | 00000004 |
| Zeros | 00000005 |
| ANSI X9.23 | 00000006 |
| ISO 10126 | 00000007 |
| PKCS1 v1.5 | 00000008 |
| X9.31 | 00000009 |
| PSS | 0000000A |
| Extensions | 8XXXXXXX |

Table 577: Padding Method Enumeration

## PKCS#11 Function Enumeration

The PKCS#11 Function enumerations are the 1-based offset count of the function in the CK\_FUNCTION\_LIST\_3\_0 structure as specified in [PKCS#11]

## PKCS#11 Return Code Enumeration

The PKCS#11 Return Codes enumerations representing PKCS#11 return codes as specified in the CK\_RV values in [PKCS#11]

## Processing Stage Enumeration

|  |  |
| --- | --- |
| **Processing Stage** | |
| **Name** | **Value** |
| Submitted | 00000001 |
| In Process | 00000002 |
| Completed | 00000003 |
| Extensions | 8XXXXXXX |

Table 578: Processing Stage Enumeration

## Profile Name Enumeration

|  |  |
| --- | --- |
| **Profile Name** | |
| **Name** | **Value** |
| (Reserved) | 00000001-00000103 |
| Complete Server Basic | 00000104 |
| Complete Server TLS v1.2 | 00000105 |
| Tape Library Client | 00000106 |
| Tape Library Server | 00000107 |
| Symmetric Key Lifecycle Client | 00000108 |
| Symmetric Key Lifecycle Server | 00000109 |
| Asymmetric Key Lifecycle Client | 0000010A |
| Asymmetric Key Lifecycle Server | 0000010B |
| Basic Cryptographic Client | 0000010C |
| Basic Cryptographic Server | 0000010D |
| Advanced Cryptographic Client | 0000010E |
| Advanced Cryptographic Server | 0000010F |
| RNG Cryptographic Client | 00000110 |
| RNG Cryptographic Server | 00000111 |
| Basic Symmetric Key Foundry Client | 00000112 |
| Intermediate Symmetric Key Foundry Client | 00000113 |
| Advanced Symmetric Key Foundry Client | 00000114 |
| Symmetric Key Foundry Server | 00000115 |
| Opaque Managed Object Store Client | 00000116 |
| Opaque Managed Object Store Server | 00000117 |
| (Reserved) | 00000118 |
| (Reserved) | 00000119 |
| (Reserved) | 0000011A |
| (Reserved) | 0000011B |
| Storage Array with Self Encrypting Drive Client | 0000011C |
| Storage Array with Self Encrypting Drive Server | 0000011D |
| HTTPS Client | 0000011E |
| HTTPS Server | 0000011F |
| JSON Client | 00000120 |
| JSON Server | 00000121 |
| XML Client | 00000122 |
| XML Server | 00000123 |
| AES XTS Client | 00000124 |
| AES XTS Server | 00000125 |
| Quantum Safe Client | 00000126 |
| Quantum Safe Server | 00000127 |
| PKCS#11 Client | 00000128 |
| PKCS#11 Server | 00000129 |
| Baseline Client | 0000012A |
| Baseline Server | 0000012B |
| Complete Server | 0000012C |
| Extensions | 8XXXXXXX |

Table 579: Profile Name Enumeration

## Protection Level Enumeration

|  |  |
| --- | --- |
| **Protection Level** | |
| **Name** | **Value** |
| High | 00000001 |
| Low | 00000002 |
| Extensions | 8XXXXXXX |

Table 580: Protection Level Enumeration

## Put Function Enumeration

|  |  |
| --- | --- |
| **Put Function** | |
| **Name** | **Value** |
| New | 00000001 |
| Replace | 00000002 |
| Extensions | 8XXXXXXX |

Table 581: Put Function Enumeration

## Query Function Enumeration

|  |  |
| --- | --- |
| **Query Function** | |
| **Name** | **Value** |
| Query Operations | 00000001 |
| Query Objects | 00000002 |
| Query Server Information | 00000003 |
| Query Application Namespaces | 00000004 |
| Query Extension List | 00000005 |
| Query Extension Map | 00000006 |
| Query Attestation Types | 00000007 |
| Query RNGs | 00000008 |
| Query Validations | 00000009 |
| Query Profiles | 0000000A |
| Query Capabilities | 0000000B |
| Query Client Registration Methods | 0000000C |
| Query Defaults Information | 0000000D |
| Query Storage Protection Masks | 0000000E |
| Query Credential Information | 0000000F |
| Extensions | 8XXXXXXX |

Table 582: Query Function Enumeration

## Recommended Curve Enumeration

|  |  |
| --- | --- |
| **Recommended Curve Enumeration** | |
| **Name** | **Value** |
| P-192 | 00000001 |
| K-163 | 00000002 |
| B-163 | 00000003 |
| P-224 | 00000004 |
| K-233 | 00000005 |
| B-233 | 00000006 |
| P-256 | 00000007 |
| K-283 | 00000008 |
| B-283 | 00000009 |
| P-384 | 0000000A |
| K-409 | 0000000B |
| B-409 | 0000000C |
| P-521 | 0000000D |
| K-571 | 0000000E |
| B-571 | 0000000F |
| SECP112R1 | 00000010 |
| SECP112R2 | 00000011 |
| SECP128R1 | 00000012 |
| SECP128R2 | 00000013 |
| SECP160K1 | 00000014 |
| SECP160R1 | 00000015 |
| SECP160R2 | 00000016 |
| SECP192K1 | 00000017 |
| SECP224K1 | 00000018 |
| SECP256K1 | 00000019 |
| SECT113R1 | 0000001A |
| SECT113R2 | 0000001B |
| SECT131R1 | 0000001C |
| SECT131R2 | 0000001D |
| SECT163R1 | 0000001E |
| SECT193R1 | 0000001F |
| SECT193R2 | 00000020 |
| SECT239K1 | 00000021 |
| ANSIX9P192V2 | 00000022 |
| ANSIX9P192V3 | 00000023 |
| ANSIX9P239V1 | 00000024 |
| ANSIX9P239V2 | 00000025 |
| ANSIX9P239V3 | 00000026 |
| ANSIX9C2PNB163V1 | 00000027 |
| ANSIX9C2PNB163V2 | 00000028 |
| ANSIX9C2PNB163V3 | 00000029 |
| ANSIX9C2PNB176V1 | 0000002A |
| ANSIX9C2TNB191V1 | 0000002B |
| ANSIX9C2TNB191V2 | 0000002C |
| ANSIX9C2TNB191V3 | 0000002D |
| ANSIX9C2PNB208W1 | 0000002E |
| ANSIX9C2TNB239V1 | 0000002F |
| ANSIX9C2TNB239V2 | 00000030 |
| ANSIX9C2TNB239V3 | 00000031 |
| ANSIX9C2PNB272W1 | 00000032 |
| ANSIX9C2PNB304W1 | 00000033 |
| ANSIX9C2TNB359V1 | 00000034 |
| ANSIX9C2PNB368W1 | 00000035 |
| ANSIX9C2TNB431R1 | 00000036 |
| BRAINPOOLP160R1 | 00000037 |
| BRAINPOOLP160T1 | 00000038 |
| BRAINPOOLP192R1 | 00000039 |
| BRAINPOOLP192T1 | 0000003A |
| BRAINPOOLP224R1 | 0000003B |
| BRAINPOOLP224T1 | 0000003C |
| BRAINPOOLP256R1 | 0000003D |
| BRAINPOOLP256T1 | 0000003E |
| BRAINPOOLP320R1 | 0000003F |
| BRAINPOOLP320T1 | 00000040 |
| BRAINPOOLP384R1 | 00000041 |
| BRAINPOOLP384T1 | 00000042 |
| BRAINPOOLP512R1 | 00000043 |
| BRAINPOOLP512T1 | 00000044 |
| CURVE25519 | 00000045 |
| CURVE448 | 00000046 |
| Extensions | 8XXXXXXX |

Table 583: Recommended Curve Enumeration for ECDSA, ECDH, and ECMQV

## Result Reason Enumeration

Following are the Result Reason enumerations.

|  |  |
| --- | --- |
| **Value** | **Description** |
| Application Namespace Not Supported | The particular Application Namespace is not supported, and the server was not able to generate the Application Data field of an Application Specific Information attribute if the field was omitted from the client request |
| Attestation Failed | Operation requires attestation data and the attestation data provided by the client does not validate |
| Attestation Required | Operation requires attestation data which was not provided by the client, and the client has set the Attestation Capable indicator to True |
| Attribute Instance Not Found | A referenced attribute was found, but the specific instance was not found |
| Attribute Not Found | A referenced attribute was not found at all on an object |
| Attribute Read Only | Attempt to set a Read Only Attribute |
| Attribute Single Instance | Attempt to provide multiple values for a single instance attribute |
| Authentication not successful | The authentication information in the request could not be validated, or was not found |
| Bad Cryptographic Parameters | Bad Cryptographic Parameters |
| Bad Password | Key Format Type is PKCS#12, but missing or multiple PKCS#12 Password Links, or not Secret Data, or not Active |
| Circular Link Error | A ParentLink sets up a directed acyclic relationship. Detection of a cycle in the relationship graph results in this reason code. |
| Codec Error | The low level TTLV, XML, JSON etc. was badly formed and not understood by the server.TTLV connections should be closed as future requests might not be correctly separated |
| Constraint Violation | The request failed because one or more  constraints were violated |
| Cryptographic Failure | The operation failed due to a cryptographic error |
| Duplicate Process Request | The asynchronous request specified was already  processed |
| Encoding Option Error | The Encoding Option is not supported as specified by the Encoding Option Enumeration |
| Feature Not Supported | The operation is supported, but not a specific feature specified in the request is not supported |
| General failure | The request failed for a reason other than the defined reasons above |
| Illegal Object Type | Check cannot be performed on this object type |
| Incompatible Cryptographic Usage Mask | The cryptographic algorithm or other parameters is not valid for the requested operation |
| Internal Server Error | The server had an internal error and could not process the request at this time. |
| Invalid Asynchronous Correlation Value | No outstanding operation with the specified Asynchronous Correlation Value exists |
| Invalid Attribute | An attribute is invalid for this object for this operation |
| Invalid Attribute Value | The value supplied for an attribute is invalid |
| Invalid Correlation Value | For streaming cryptographic operations |
| Invalid CSR | Invalid Certificate Signing Request |
| Invalid Data Type | A data type was invalid for the requested operation |
| Invalid Field | The request is syntactically valid but some data in the request (other than an attribute value) has an invalid value |
| Invalid Message | The request message was not syntactically understood by the server. For example - the invalid use of a known tag |
| Invalid Object Type | Specified object is not valid for the requested operation |
| Invalid Password |  |
| Invalid Ticket | The ticket was invalid |
| Item Not Found | No object with the specified Unique Identifier exists |
| Key Compression Type Not Supported | The object exists, but the server is unable to provide it in the desired Key Compression Type |
| Key Format Type Not Supported | The object exists, but the server is unable to provide it in the desired Key Format Type |
| Key Value Not Present | A meta data only object. The key value is not present on the server |
| Key Wrap Type Not Supported | Key Wrap Type Type is not supported by the server |
| Missing data | The operation REQUIRED additional information in the request, which was not present |
| Missing Initialization Vector | Missing IV when required for crypto operation |
| Multi Valued Attribute | Attempt to Set or Adjust an attribute that has multiple values |
| Non Unique Name Attribute | Trying to perform an operation that requests the server to break the constraint on Name attribute being unique |
| Not Extractable | Object is not Extractable |
| Numeric Range | An operation produced a number that is too large or too small to be stored in the specified data type |
| Object Already Exists | for operations such as Import that require that no object with a specific unique identifier exists on a server |
| Object Archived | The object SHALL be recovered from the archive before performing the operation |
| Object Destroyed | Object exists, but has already been destroyed |
| Object Not Found | A requested managed object was not found or did not exist |
| Object Type | Invalid object type for the operation |
| Operation canceled by requester | The operation was asynchronous, and the operation was canceled by the Cancel operation before it completed successfully |
| Operation Not Supported | The operation requested by the request message is not supported by the server |
| Permission Denied | Client is not allowed to perform the specified operation |
| PKCS#11 Codec Error | There is a Codec error in the Input parameter |
| PKCS#11 Invalid Function | The PKCS function is not in the interface |
| PKCS#11 Invalid Interface | The interface is unknown or unavailable in the  server |
| Protection Storage Unavailable, Private Protection Storage Unavailable, Public Protection Storage Unavailable | The operation could not be completed with the protections requested (or defaulted). |
| Read Only Attribute | Attempt to set a Read Only Attribute |
| Response Too Large | Maximum Response Size has been exceeded |
| Sensitive | Sensitive keys may not be retrieved unwrapped |
| Server Limit Exceeded | Some limit on the server such as database size has been exceeded |
| Unknown Enumeration | An enumerated value is not known by the server |
| Unknown Message Extension | The server does not support the supplied Message Extension |
| Unknown Tag | A tag is not known by the server |
| Unsupported Attribute | Attribute is valid in the specification but unsupported by the Server |
| Unsupported Cryptographic Parameters | Cryptographic Parameters are valid in the specification but unsupported by the Server |
| Unsupported Protocol Version | The operation cannot be performed with the provided protocol version |
| Usage Limit Exceeded | The usage limits or request count has been exceeded |
| Wrapping Object Archived | Wrapping Object is archived |
| Wrapping Object Destroyed | The object exists, but is destroyed |
| Wrapping Object Not Found | Wrapping object does not exist |
| Wrong Key Lifecycle State | The key lifecycle state is invalid for the operation, for example not Active for an Encrypt operation |
| General failure | The request failed for a reason other than any other reason enumeration value. |

Table 584: Result Reason Encoding Descriptions

|  |  |
| --- | --- |
| **Result Reason** | |
| **Name** | **Value** |
| Item Not Found | 00000001 |
| Response Too Large | 00000002 |
| Authentication Not Successful | 00000003 |
| Invalid Message | 00000004 |
| Operation Not Supported | 00000005 |
| Missing Data | 00000006 |
| Invalid Field | 00000007 |
| Feature Not Supported | 00000008 |
| Operation Canceled By Requester | 00000009 |
| Cryptographic Failure | 0000000A |
| (Reserved) | 0000000B |
| Permission Denied | 0000000C |
| Object Archived | 0000000D |
| (Reserved) | 0000000E |
| Application Namespace Not Supported | 0000000F |
| Key Format Type Not Supported | 00000010 |
| Key Compression Type Not Supported | 00000011 |
| Encoding Option Error | 00000012 |
| Key Value Not Present | 00000013 |
| Attestation Required | 00000014 |
| Attestation Failed | 00000015 |
| Sensitive | 00000016 |
| Not Extractable | 00000017 |
| Object Already Exists | 00000018 |
| Invalid Ticket | 00000019 |
| Usage Limit Exceeded | 0000001A |
| Numeric Range | 0000001B |
| Invalid Data Type | 0000001C |
| Read Only Attribute | 0000001D |
| Multi Valued Attribute | 0000001E |
| Unsupported Attribute | 0000001F |
| Attribute Instance Not Found | 00000020 |
| Attribute Not Found | 00000021 |
| Attribute Read Only | 00000022 |
| Attribute Single Valued | 00000023 |
| Bad Cryptographic Parameters | 00000024 |
| Bad Password | 00000025 |
| Codec Error | 00000026 |
| (Reserved) | 00000027 |
| Illegal Object Type | 00000028 |
| Incompatible Cryptographic Usage Mask | 00000029 |
| Internal Server Error | 0000002A |
| Invalid Asynchronous Correlation Value | 0000002B |
| Invalid Attribute | 0000002C |
| Invalid Attribute Value | 0000002D |
| Invalid Correlation Value | 0000002E |
| Invalid CSR | 0000002F |
| Invalid Object Type | 00000030 |
| (Reserved) | 00000031 |
| Key Wrap Type Not Supported | 00000032 |
| (Reserved) | 00000033 |
| Missing Initialization Vector | 00000034 |
| Non Unique Name Attribute | 00000035 |
| Object Destroyed | 00000036 |
| Object Not Found | 00000037 |
| (Reserved) | 00000038 |
| Not Authorised | 00000039 |
| Server Limit Exceeded | 0000003A |
| Unknown Enumeration | 0000003B |
| Unknown Message Extension | 0000003C |
| Unknown Tag | 0000003D |
| Unsupported Cryptographic Parameters | 0000003E |
| Unsupported Protocol Version | 0000003F |
| Wrapping Object Archived | 00000040 |
| Wrapping Object Destroyed | 00000041 |
| Wrapping Object Not Found | 00000042 |
| Wrong Key Lifecycle State | 00000043 |
| Protection Storage Unavailable | 00000044 |
| PKCS#11 Codec Error | 00000045 |
| PKCS#11 Invalid Function | 00000046 |
| PKCS#11 Invalid Interface | 00000047 |
| Private Protection Storage Unavailable | 00000048 |
| Public Protection Storage Unavailable | 00000049 |
| (Reserved) | 0000004A |
| Constraint Violation | 0000004B |
| Duplicate Process Request | 0000004C |
| Circular Link Error | 0000004D |
| General Failure | 00000100 |
| Extensions | 8XXXXXXX |

Table 585: Result Reason Enumeration

## Result Status Enumeration

|  |  |
| --- | --- |
| **Result Status** | |
| **Name** | **Value** |
| Success | 00000000 |
| Operation Failed | 00000001 |
| Operation Pending | 00000002 |
| Operation Undone | 00000003 |
| Extensions | 8XXXXXXX |

Table 586: Result Status Enumeration

## Revocation Reason Code Enumeration

|  |  |
| --- | --- |
| **Revocation Reason Code** | |
| **Name** | **Value** |
| Unspecified | 00000001 |
| Key Compromise | 00000002 |
| CA Compromise | 00000003 |
| Affiliation Changed | 00000004 |
| Superseded | 00000005 |
| Cessation of Operation | 00000006 |
| Privilege Withdrawn | 00000007 |
| Extensions | 8XXXXXXX |

Table 587: Revocation Reason Code Enumeration

## RNG Algorithm Enumeration

|  |  |
| --- | --- |
| **RNG Algorithm** | |
| **Name** | **Value** |
| Unspecified | 00000001 |
| FIPS 186-2 | 00000002 |
| DRBG | 00000003 |
| NRBG | 00000004 |
| ANSI X9.31 | 00000005 |
| ANSI X9.62 | 00000006 |
| Extensions | 8XXXXXXX |

Table 588: RNG Algorithm Enumeration

Note: the user should be aware that a number of these algorithms are no longer recommended for general use and/or are deprecated. They are included for completeness.

## RNG Mode Enumeration

|  |  |
| --- | --- |
| **RNG Mode** | |
| **Name** | **Value** |
| Unspecified | 00000001 |
| Shared Instantiation | 00000002 |
| Non-Shared Instantiation | 00000003 |
| Extensions | 8XXXXXXX |

Table 589: RNG Mode Enumeration

## Secret Data Type Enumeration

|  |  |
| --- | --- |
| **Secret Data Type** | |
| **Name** | **Value** |
| Password | 00000001 |
| Seed | 00000002 |
| Extensions | 8XXXXXXX |

Table 590: Secret Data Type Enumeration

## Shredding Algorithm Enumeration

|  |  |
| --- | --- |
| **Shredding Algorithm** | |
| **Name** | **Value** |
| Unspecified | 00000001 |
| Cryptographic | 00000002 |
| Unsupported | 00000003 |
| Extensions | 8XXXXXXX |

Table 591: Shredding Algorithm Enumeration

## Split Key Method Enumeration

|  |  |
| --- | --- |
| **Split Key Method** | |
| **Name** | **Value** |
| XOR | 00000001 |
| Polynomial Sharing GF (216) | 00000002 |
| Polynomial Sharing Prime Field | 00000003 |
| Polynomial Sharing GF (28) | 00000004 |
| Extensions | 8XXXXXXX |

Table 592: Split Key Method Enumeration

## Split Key Polynomial Enumeration

|  |  |
| --- | --- |
| **State** | |
| **Name** | **Value** |
| Polynomial-283 | 00000001 |
| Polynomial-285 | 00000002 |
| Extensions | 8XXXXXXX |

Table 593: Split Key Polynomial Enumeration

## State Enumeration

|  |  |
| --- | --- |
| **State** | |
| **Name** | **Value** |
| Pre-Active | 00000001 |
| Active | 00000002 |
| Deactivated | 00000003 |
| Compromised | 00000004 |
| Destroyed | 00000005 |
| Destroyed Compromised | 00000006 |
| Extensions | 8XXXXXXX |

Table 594: State Enumeration

## Tag Enumeration

All tags SHALL contain either the value 42 in hex or the value 54 in hex as the first byte of a three (3) byte enumeration value. Tags defined by this specification contain hex 42 in the first byte. Extensions contain the value 54 hex in the first byte.

| **Tag** | |
| --- | --- |
| **Name** | **Value** |
| (Unused) | 000000 - 420000 |
| Activation Date | 420001 |
| Application Data | 420002 |
| Application Namespace | 420003 |
| Application Specific Information | 420004 |
| Archive Date | 420005 |
| Asynchronous Correlation Value | 420006 |
| Asynchronous Indicator | 420007 |
| Attribute | 420008 |
| (Reserved) | 420009 |
| Attribute Name | 42000A |
| Attribute Value | 42000B |
| Authentication | 42000C |
| (Reserved) | 42000D |
| Batch Error Continuation Option | 42000E |
| Batch Item | 42000F |
| (Reserved) | 420010 |
| Block Cipher Mode | 420011 |
| Cancellation Result | 420012 |
| Certificate | 420013 |
| (Reserved) | 420014 |
| (Reserved) | 420015 |
| (Reserved) | 420016 |
| (Reserved) | 420017 |
| Certificate Request | 420018 |
| Certificate Request Type | 420019 |
| (Reserved) | 42001A |
| (Reserved) | 42001B |
| (Reserved) | 42001C |
| Certificate Type | 42001D |
| Certificate Value | 42001E |
| (Reserved) | 42001F |
| Compromise Date | 420020 |
| Compromise Occurrence Date | 420021 |
| Contact Information | 420022 |
| Credential | 420023 |
| Credential Type | 420024 |
| Credential Value | 420025 |
| Criticality Indicator | 420026 |
| CRT Coefficient | 420027 |
| Cryptographic Algorithm | 420028 |
| Cryptographic Domain Parameters | 420029 |
| Cryptographic Length | 42002A |
| Cryptographic Parameters | 42002B |
| Cryptographic Usage Mask | 42002C |
| (Reserved) | 42002D |
| D | 42002E |
| Deactivation Date | 42002F |
| Derivation Data | 420030 |
| Derivation Method | 420031 |
| Derivation Parameters | 420032 |
| Destroy Date | 420033 |
| Digest | 420034 |
| Digest Value | 420035 |
| Encryption Key Information | 420036 |
| G | 420037 |
| Hashing Algorithm | 420038 |
| Initial Date | 420039 |
| Initialization Vector | 42003A |
| (Reserved) | 42003B |
| Iteration Count | 42003C |
| IV/Counter/Nonce | 42003D |
| J | 42003E |
| Key | 42003F |
| Key Block | 420040 |
| Key Compression Type | 420041 |
| Key Format Type | 420042 |
| Key Material | 420043 |
| Key Part Identifier | 420044 |
| Key Value | 420045 |
| Key Wrapping Data | 420046 |
| Key Wrapping Specification | 420047 |
| Last Change Date | 420048 |
| Lease Time | 420049 |
| (Reserved) | 42004A |
| (Reserved) | 42004B |
| (Reserved) | 42004C |
| MAC/Signature | 42004D |
| MAC/Signature Key Information | 42004E |
| Maximum Items | 42004F |
| Maximum Response Size | 420050 |
| Message Extension | 420051 |
| Modulus | 420052 |
| Name | 420053 |
| (Reserved) | 420054 |
| (Reserved) | 420055 |
| (Reserved) | 420056 |
| Object Type | 420057 |
| Offset | 420058 |
| Opaque Data Type | 420059 |
| Opaque Data Value | 42005A |
| Opaque Object | 42005B |
| Operation | 42005C |
| (Reserved) | 42005D |
| P | 42005E |
| Padding Method | 42005F |
| Prime Exponent P | 420060 |
| Prime Exponent Q | 420061 |
| Prime Field Size | 420062 |
| Private Exponent | 420063 |
| Private Key | 420064 |
| (Reserved) | 420065 |
| Private Key Unique Identifier | 420066 |
| Process Start Date | 420067 |
| Protect Stop Date | 420068 |
| Protocol Version | 420069 |
| Protocol Version Major | 42006A |
| Protocol Version Minor | 42006B |
| Public Exponent | 42006C |
| Public Key | 42006D |
| (Reserved) | 42006E |
| Public Key Unique Identifier | 42006F |
| Put Function | 420070 |
| Q | 420071 |
| Q String | 420072 |
| Qlength | 420073 |
| Query Function | 420074 |
| Recommended Curve | 420075 |
| Replaced Unique Identifier | 420076 |
| Request Header | 420077 |
| Request Message | 420078 |
| Request Payload | 420079 |
| Response Header | 42007A |
| Response Message | 42007B |
| Response Payload | 42007C |
| Result Message | 42007D |
| Result Reason | 42007E |
| Result Status | 42007F |
| Revocation Message | 420080 |
| Revocation Reason | 420081 |
| Revocation Reason Code | 420082 |
| Key Role Type | 420083 |
| Salt | 420084 |
| Secret Data | 420085 |
| Secret Data Type | 420086 |
| (Reserved) | 420087 |
| Server Information | 420088 |
| Split Key | 420089 |
| Split Key Method | 42008A |
| Split Key Parts | 42008B |
| Split Key Threshold | 42008C |
| State | 42008D |
| Storage Status Mask | 42008E |
| Symmetric Key | 42008F |
| (Reserved) | 420090 |
| (Reserved) | 420091 |
| Time Stamp | 420092 |
| (Reserved) | 420093 |
| Unique Identifier | 420094 |
| Usage Limits | 420095 |
| Usage Limits Count | 420096 |
| Usage Limits Total | 420097 |
| Usage Limits Unit | 420098 |
| Username | 420099 |
| Validity Date | 42009A |
| Validity Indicator | 42009B |
| Vendor Extension | 42009C |
| Vendor Identification | 42009D |
| Wrapping Method | 42009E |
| X | 42009F |
| Y | 4200A0 |
| Password | 4200A1 |
| Device Identifier | 4200A2 |
| Encoding Option | 4200A3 |
| Extension Information | 4200A4 |
| Extension Name | 4200A5 |
| Extension Tag | 4200A6 |
| Extension Type | 4200A7 |
| Fresh | 4200A8 |
| Machine Identifier | 4200A9 |
| Media Identifier | 4200AA |
| Network Identifier | 4200AB |
| (Reserved) | 4200AC |
| Certificate Length | 4200AD |
| Digital Signature Algorithm | 4200AE |
| Certificate Serial Number | 4200AF |
| Device Serial Number | 4200B0 |
| Issuer Alternative Name | 4200B1 |
| Issuer Distinguished Name | 4200B2 |
| Subject Alternative Name | 4200B3 |
| Subject Distinguished Name | 4200B4 |
| X.509 Certificate Identifier | 4200B5 |
| X.509 Certificate Issuer | 4200B6 |
| X.509 Certificate Subject | 4200B7 |
| Key Value Location | 4200B8 |
| Key Value Location Value | 4200B9 |
| Key Value Location Type | 4200BA |
| Key Value Present | 4200BB |
| Original Creation Date | 4200BC |
| PGP Key | 4200BD |
| PGP Key Version | 4200BE |
| Alternative Name | 4200BF |
| Alternative Name Value | 4200C0 |
| Alternative Name Type | 4200C1 |
| Data | 4200C2 |
| Signature Data | 4200C3 |
| Data Length | 4200C4 |
| Random IV | 4200C5 |
| MAC Data | 4200C6 |
| Attestation Type | 4200C7 |
| Nonce | 4200C8 |
| Nonce ID | 4200C9 |
| Nonce Value | 4200CA |
| Attestation Measurement | 4200CB |
| Attestation Assertion | 4200CC |
| IV Length | 4200CD |
| Tag Length | 4200CE |
| Fixed Field Length | 4200CF |
| Counter Length | 4200D0 |
| Initial Counter Value | 4200D1 |
| Invocation Field Length | 4200D2 |
| Attestation Capable Indicator | 4200D3 |
| Offset Items | 4200D4 |
| Located Items | 4200D5 |
| Correlation Value | 4200D6 |
| Init Indicator | 4200D7 |
| Final Indicator | 4200D8 |
| RNG Parameters | 4200D9 |
| RNG Algorithm | 4200DA |
| DRBG Algorithm | 4200DB |
| FIPS186 Variation | 4200DC |
| Prediction Resistance | 4200DD |
| Random Number Generator | 4200DE |
| Validation Information | 4200DF |
| Validation Authority Type | 4200E0 |
| Validation Authority Country | 4200E1 |
| Validation Authority URI | 4200E2 |
| Validation Version Major | 4200E3 |
| Validation Version Minor | 4200E4 |
| Validation Type | 4200E5 |
| Validation Level | 4200E6 |
| Validation Certificate Identifier | 4200E7 |
| Validation Certificate URI | 4200E8 |
| Validation Vendor URI | 4200E9 |
| Validation Profile | 4200EA |
| Profile Information | 4200EB |
| Profile Name | 4200EC |
| Server URI | 4200ED |
| Server Port | 4200EE |
| Streaming Capability | 4200EF |
| Asynchronous Capability | 4200F0 |
| Attestation Capability | 4200F1 |
| Unwrap Mode | 4200F2 |
| Destroy Action | 4200F3 |
| Shredding Algorithm | 4200F4 |
| RNG Mode | 4200F5 |
| Client Registration Method | 4200F6 |
| Capability Information | 4200F7 |
| Key Wrap Type | 4200F8 |
| Batch Undo Capability | 4200F9 |
| Batch Continue Capability | 4200FA |
| PKCS#12 Friendly Name | 4200FB |
| Description | 4200FC |
| Comment | 4200FD |
| Authenticated  Encryption Additional Data | 4200FE |
| Authenticated Encryption Tag | 4200FF |
| Salt Length | 420100 |
| Mask Generator | 420101 |
| Mask Generator Hashing Algorithm | 420102 |
| P Source | 420103 |
| Trailer Field | 420104 |
| Client Correlation Value | 420105 |
| Server Correlation Value | 420106 |
| Digested Data | 420107 |
| Certificate Subject CN | 420108 |
| Certificate Subject O | 420109 |
| Certificate Subject OU | 42010A |
| Certificate Subject Email | 42010B |
| Certificate Subject C | 42010C |
| Certificate Subject ST | 42010D |
| Certificate Subject L | 42010E |
| Certificate Subject UID | 42010F |
| Certificate Subject Serial Number | 420110 |
| Certificate Subject Title | 420111 |
| Certificate Subject DC | 420112 |
| Certificate Subject DN Qualifier | 420113 |
| Certificate Issuer CN | 420114 |
| Certificate Issuer O | 420115 |
| Certificate Issuer OU | 420116 |
| Certificate Issuer Email | 420117 |
| Certificate Issuer C | 420118 |
| Certificate Issuer ST | 420119 |
| Certificate Issuer L | 42011A |
| Certificate Issuer UID | 42011B |
| Certificate Issuer Serial Number | 42011C |
| Certificate Issuer Title | 42011D |
| Certificate Issuer DC | 42011E |
| Certificate Issuer DN Qualifier | 42011F |
| Sensitive | 420120 |
| Always Sensitive | 420121 |
| Extractable | 420122 |
| Never Extractable | 420123 |
| Replace Existing | 420124 |
| Attributes | 420125 |
| Common Attributes | 420126 |
| Private Key Attributes | 420127 |
| Public Key Attributes | 420128 |
| Extension Enumeration | 420129 |
| Extension Attribute | 42012A |
| Extension Parent Structure Tag | 42012B |
| Extension Description | 42012C |
| Server Name | 42012D |
| Server Serial Number | 42012E |
| Server Version | 42012F |
| Server Load | 420130 |
| Product Name | 420131 |
| Build Level | 420132 |
| Build Date | 420133 |
| Cluster Info | 420134 |
| Alternate Failover Endpoints | 420135 |
| Short Unique Identifier | 420136 |
| Reserved | 420137 |
| Tag | 420138 |
| Certificate Request Unique Identifier | 420139 |
| NIST Key Type | 42013A |
| Attribute Reference | 42013B |
| Current Attribute | 42013C |
| New Attribute | 42013D |
| (Reserved) | 42013E |
| (Reserved) | 42013F |
| Certificate Request Value | 420140 |
| Log Message | 420141 |
| Profile Version | 420142 |
| Profile Version Major | 420143 |
| Profile Version Minor | 420144 |
| Protection Level | 420145 |
| Protection Period | 420146 |
| Quantum Safe | 420147 |
| Quantum Safe Capability | 420148 |
| Ticket | 420149 |
| Ticket Type | 42014A |
| Ticket Value | 42014B |
| Request Count | 42014C |
| Rights | 42014D |
| Objects | 42014E |
| Operations | 42014F |
| Right | 420150 |
| Endpoint Role | 420151 |
| Defaults Information | 420152 |
| Object Defaults | 420153 |
| Ephemeral | 420154 |
| Server Hashed Password | 420155 |
| One Time Password | 420156 |
| Hashed Password | 420157 |
| Adjustment Type | 420158 |
| PKCS#11 Interface | 420159 |
| PKCS#11 Function | 42015A |
| PKCS#11 Input Parameters | 42015B |
| PKCS#11 Output Parameters | 42015C |
| PKCS#11 Return Code | 42015D |
| Protection Storage Mask | 42015E |
| Protection Storage Masks | 42015F |
| Interop Function | 420160 |
| Interop Identifier | 420161 |
| Adjustment Value | 420162 |
| Common Protection Storage Masks | 420163 |
| Private Protection Storage Masks | 420164 |
| Public Protection Storage Masks | 420165 |
| Object Groups | 420166 |
| Object Types | 420167 |
| Constraints | 420168 |
| Constraint | 420169 |
| Rotate Interval | 0x42016A |
| Rotate Automatic | 0x42016B |
| Rotate Offset | 0x42016C |
| Rotate Date | 0x42016D |
| Rotate Generation | 0x42016E |
| Rotate Name | 0x42016F |
| (Reserved) | 0x420170 |
| (Reserved) | 0x420171 |
| Rotate Latest | 0x420172 |
| Asynchronous Request | 0x420173 |
| Submission Date | 0x420174 |
| Processing Stage | 0x420175 |
| Asynchronous Correlation Values | 0x420176 |
| Certificate Link | 0x420190 |
| Child Link | 0x420191 |
| Derivation Object Link | 0x420192 |
| Derived Object Link | 0x420193 |
| Next Link | 0x420194 |
| Parent Link | 0x420195 |
| PKCS#12 Certificate Link | 0x420196 |
| PKCS#12 Password Link | 0x420197 |
| Previous Link | 0x420198 |
| Private Key Link | 0x420199 |
| Public Key Link | 0x42019A |
| Replaced Object Link | 0x42019B |
| Replacement Object Link | 0x42019C |
| Wrapping Key Link | 0x42019D |
| Object Class | 0x42019E |
| Object Class Mask | 0x42019F |
| Credential Link | 0x4201A0 |
| Password Credential | 0x4201A1 |
| Password Salt | 0x4201A2 |
| Password Salt Algorithm | 0x4201A3 |
| Salted Password | 0x4201A4 |
| Password Link | 0x4201A5 |
| Device Credential | 0x4201A6 |
| OTP Credential | 0x4201A7 |
| OTP Algorithm | 0x4201A8 |
| OTP Digest | 0x4201A9 |
| OTP Serial | 0x4201AA |
| OTP Seed | 0x4201AB |
| OTP Interval | 0x4201AC |
| OTP Digits | 0x4201AD |
| OTP Counter | 0x4201AE |
| Hashed Password Credential | 0x4201AF |
| Hashed Username Password | 0x4201B0 |
| Hashed Password Username | 0x4201B1 |
| Credential Information | 0x4201B2 |
| Group Link | 0x4201B3 |
| Split Key Base Link | 0x4201B4 |
| Joined Split Key Parts Link | 0x4201B5 |
| Split Key Polynomial | 0x4201B6 |
| Deactivation Message | 0x4201B7 |
| Deactivation Reason | 0x4201B8 |
| Deactivation Reason Code | 0x4201B9 |
| Certificate Subject DN | 0x4201BA |
| Certificate Issuer DN | 0x4201BB |
| Certificate Request Link | 0x4201BC |
| Certify Counter | 0x4201BD |
| Decrypt Counter | 0x4201BE |
| Encrypt Counter | 0x4201BF |
| Sign Counter | 0x4201C0 |
| Signature Verify Counter | 0x4201C1 |
| NIST Security Category | 0x4201C2 |
| (Reserved) | 420XXX – 42FFFF |
| (Unused) | 430000 – 53FFFF |
| Extensions | 540000 – 54FFFF |
| (Unused) | 550000 - FFFFFF |

Table 595: Tag Enumeration

## Ticket Type Enumeration

|  |  |
| --- | --- |
| **State** | |
| **Name** | **Value** |
| Login | 00000001 |
| Extensions | 8XXXXXXX |

Table 596: Ticket Type Enumeration

## Unique Identifier Enumeration

The following values may be specified in an operation request for a Unique Identifier: If multiple unique identifiers would be referenced then the operation is repeated for each of them as separate batch items. If an operation appears multiple times in a request, it is the most recent that is referred to.

|  |  |
| --- | --- |
| **Unique Identifier Enumerations** | |
| **Name** | **Value** |
| ID Placeholder | 00000001 |
| Certify | 00000002 |
| Create | 00000003 |
| Create Key Pair | 00000004 |
| Create Key Pair Private Key | 00000005 |
| Create Key Pair Public Key | 00000006 |
| Create Split Key | 00000007 |
| Derive Key | 00000008 |
| Import | 00000009 |
| Join Split Key | 0000000A |
| Locate | 0000000B |
| Register | 0000000C |
| Re-key | 0000000D |
| Re-certify | 0000000E |
| Re-key Key Pair | 0000000F |
| Re-key Key Pair Private Key | 00000010 |
| Re-key Key Pair Public Key | 00000011 |
| Re-Provision | 00000012 |
| Create User | 00000013 |
| Create Group | 00000014 |
| Create Credential | 00000015 |
| Extensions | 8XXXXXXX |

Table 597: Unique Identifier Enumeration

## Unwrap Mode Enumeration

|  |  |
| --- | --- |
| **Unwrap Mode** | |
| **Name** | **Value** |
| Unspecified | 00000001 |
| Processed | 00000002 |
| Not Processed | 00000003 |
| Extensions | 8XXXXXXX |

Table 598: Unwrap Mode Enumeration

## Usage Limits Unit Enumeration

|  |  |
| --- | --- |
| **Usage Limits Unit** | |
| **Name** | **Value** |
| Byte | 00000001 |
| Object | 00000002 |
| Extensions | 8XXXXXXX |

Table 599: Usage Limits Unit Enumeration

## Validity Indicator Enumeration

|  |  |
| --- | --- |
| **Validity Indicator** | |
| **Name** | **Value** |
| Valid | 00000001 |
| Invalid | 00000002 |
| Unknown | 00000003 |
| Extensions | 8XXXXXXX |

Table 600: Validity Indicator Enumeration

## Wrapping Method Enumeration

The following wrapping methods are currently defined:

|  |  |
| --- | --- |
| **Value** | **Description** |
| Encrypt only | encryption using a symmetric key or public key, or authenticated encryption algorithms that use a single key |
| MAC/sign only | either MACing the Key Value with a symmetric key, or signing the Key Value with a private key |
| Encrypt then MAC/sign |  |
| MAC/sign then encrypt. |  |
| TR-31 |  |
| Extensions |  |

Table 601: Key Wrapping Methods Description

|  |  |
| --- | --- |
| **Wrapping Method** | |
| **Name** | **Value** |
| Encrypt | 00000001 |
| MAC/sign | 00000002 |
| Encrypt then MAC/sign | 00000003 |
| MAC/sign then encrypt | 00000004 |
| TR-31 | 00000005 |
| Extensions | 8XXXXXXX |

Table 602: Wrapping Method Enumeration

## Validation Authority Type Enumeration

|  |  |
| --- | --- |
| **Validation Authority Type** | |
| **Name** | **Value** |
| Unspecified | 00000001 |
| NIST CMVP | 00000002 |
| Common Criteria | 00000003 |
| Extensions | 8XXXXXXX |

Table 603: Validation Authority Type Enumeration

## Validation Type Enumeration

|  |  |
| --- | --- |
| **Validation Type** | |
| **Name** | **Value** |
| Unspecified | 00000001 |
| Hardware | 00000002 |
| Software | 00000003 |
| Firmware | 00000004 |
| Hybrid | 00000005 |
| Extensions | 8XXXXXXX |

Table 604: Validation Type Enumeration

# Bit Masks

All mask values SHALL be encoded as Integers in TTLV encoding and SHALL be encoded as Integers but in symbolic form in XML and JSON encodings.

## Cryptographic Usage Mask

The following Cryptographic Usage Masks are currently defined:

|  |  |  |
| --- | --- | --- |
| **Value** | **Description** | **Valid KMIP Server Operation** |
| Sign | Allow for signing. Applies to Sign operation. Valid for PGP Key, Private Key | Yes |
| Verify | Allow for signature verification. Applies to Signature Verify and Validate operations. Valid for PGP Key, Certificate and Public Key. | Yes |
| Encrypt | Allow for encryption. Applies to Encrypt operation. Valid for PGP Key, Private Key, Public Key and Symmetric Key. Encryption for the purpose of wrapping is separate Wrap Key value. | Yes |
| Decrypt | Allow for decryption. Applies to Decrypt operation. Valid for PGP Key, Private Key, Public Key and Symmetric Key. Decryption for the purpose of unwrapping is separate Unwrap Key value. | Yes |
| Wrap Key | Allow for key wrapping. Applies to Get operation when wrapping is required by Wrapping Specification is provided on the object used to Wrap. Valid for PGP Key, Private Key and Symmetric Key. Note: even if the underlying wrapping mechanism is encryption, this value is logically separate. | Yes |
| Unwrap Key | Allow for key unwrapping. Applies to Get operation when unwrapping is required on the object used to Unwrap. Valid for PGP Key, Private Key, Public Key and Symmetric Key. Not interchangeable with Decrypt. Note: even if the underlying unwrapping mechanism is decryption, this value is logically separate. | Yes |
| (Reserved) |  |  |
| MAC Generate | Allow for MAC generation. Applies to MAC operation. Valid for Symmetric Keys | Yes |
| MAC Verify | Allow for MAC verification. Applies to MAC Verify operation. Valid for Symmetric Keys | Yes |
| Derive Key | Allow for key derivation. Applied to Derive Key operation. Valid for PGP Keys, Private Keys, Public Keys, Secret Data and Symmetric Keys. | Yes |
| Key Agreement | Allow for Key Agreement. Valid for PGP Keys, Private Keys, Public Keys, Secret Data and Symmetric Keys | No |
| Certificate Sign | Allow for Certificate Signing. Applies to Certify operation on a private key. Valid for Private Keys. | Yes |
| CRL Sign | Allow for CRL Sign. Valid for Private Keys | Yes |
| Authenticate | Allow for Authentication. Valid for Secret Data. | Yes |
| Unrestricted | Cryptographic Usage Mask contains no Usage Restrictions. | Yes |
| FPE Encrypt | Allow for Format Preserving Encrypt. Valid for Symmetric Keys, Public Keys and Private Keys | Yes |
| FPE Decrypt | Allow for Format Preserving Decrypt. Valid for Symmetric Keys, Public Keys and Private Keys | Yes |
| Extensions | Extensions |  |

Table 605: Cryptographic Usage Masks Description

|  |  |
| --- | --- |
| **Cryptographic Usage Mask** | |
| **Name** | **Value** |
| Sign | 00000001 |
| Verify | 00000002 |
| Encrypt | 00000004 |
| Decrypt | 00000008 |
| Wrap Key | 00000010 |
| Unwrap Key | 00000020 |
| (Reserved) | 00000040 |
| MAC Generate | 00000080 |
| MAC Verify | 00000100 |
| Derive Key | 00000200 |
| (Reserved) | 00000400 |
| Key Agreement | 00000800 |
| Certificate Sign | 00001000 |
| CRL Sign | 00002000 |
| (Reserved) | 00004000 |
| (Reserved) | 00008000 |
| (Reserved) | 00010000 |
| (Reserved) | 00020000 |
| (Reserved) | 00040000 |
| (Reserved) | 00080000 |
| Authenticate | 00100000 |
| Unrestricted | 00200000 |
| FPE Encrypt | 00400000 |
| FPE Decrypt | 00800000 |
| Extensions | XXX00000 |

Table 606: Cryptographic Usage Mask enumerations

This list takes into consideration values which MAY appear in the Key Usage extension in an X.509 certificate.

## Protection Storage Mask

|  |  |
| --- | --- |
| **Protection Storage Mask** | |
| **Name** | **Value** |
| Software | 00000001 |
| Hardware | 00000002 |
| On Processor | 00000004 |
| On System | 00000008 |
| Off System | 00000010 |
| Hypervisor | 00000020 |
| Operating System | 00000040 |
| Container | 00000080 |
| On Premises | 00000100 |
| Off Premises | 00000200 |
| Self Managed | 00000400 |
| Outsourced | 00000800 |
| Validated | 00001000 |
| Same Jurisdiction | 00002000 |
| Extensions | XXXX0000 |

Table 607: Protection Storage Mask enumerations

## Storage Status Mask

|  |  |
| --- | --- |
| **Storage Status Mask** | |
| **Name** | **Value** |
| On-line storage | 00000001 |
| Archival storage | 00000002 |
| Destroyed storage | 00000004 |
| Extensions | XXXX0000 |

Table 608: Storage Status Mask enumerations

## Object Class Mask

|  |  |
| --- | --- |
| **Storage Status Mask** | |
| **Name** | **Value** |
| User | 00000001 |
| System | 00000002 |
| Extensions | XXXX0000 |

Table 609: Object Class Mask enumerations

# Algorithm Implementation

## Split Key Algorithms

There are three *Split Key Methods* for secret sharing: the first one is based on XOR, and the other two are based on polynomial secret sharing, according to [w1979].

Let *L* be the minimum number of bits needed to represent all values of the secret.

* When the Split Key Method is XOR, then the Key Material in the Key Value of the Key Block is of length *L* bits. The number of split keys is Split Key Parts (identical to Split Key Threshold), and the secret is reconstructed by XORing all of the parts.
* When the Split Key Method is Polynomial Sharing Prime Field, then secret sharing is performed in the field GF(*Prime Field Size*), represented as integers, where Prime Field Size is a prime bigger than 2*L*.
* When the Split Key Method is Polynomial Sharing GF(216), then secret sharing is performed in the field GF(216). The Key Material in the Key Value of the Key Block is a bit string of length *L*, and when *L* is bigger than 216, then secret sharing is applied piecewise in pieces of 16 bits each. The Key Material in the Key Value of the Key Block is the concatenation of the corresponding shares of all pieces of the secret.

Secret sharing is performed in the field GF(216), which is represented as an algebraic extension of GF(28):

GF(216) ≈ GF(28) [*y*]/(*y2+y+m*), where *m* is defined later.

An element of this field then consists of a linear combination *uy + v*, where *u* and *v* are elements of the smaller field GF(28).

The representation of field elements and the notation in this section rely on [FIPS197], Sections 3 and 4. The field GF(28) is as described in a format consistent with [FIPS197],

GF(28) ≈ 285 - *x8+x4+x3+x2+*1.

An element of GF(28) is represented as a byte. Addition and subtraction in GF(28) is performed as a bit-wise XOR of the bytes. Multiplication and inversion are more complex (see [FIPS197] Section 4.1 and 4.2 for details).

An element of GF(216) is represented as a pair of bytes (*u, v*). The element *m* is given by

*m* = *x5+x4+x3+x,*

which is represented by the byte 0x3A (or {3A} in notation according to [FIPS197]).

Addition and subtraction in GF(216) both correspond to simply XORing the bytes. The product of two elements *ry + s* and *uy + v*  is given by

(*ry* + *s*) (*uy* + *v*) = ((*r* + *s*)(*u* + *v*) + *sv*)*y* + (*ru* + *svm*).

The inverse of an element *uy* + *v* is given by

(*uy* + *v*)-1 = *ud*-1*y* + (*u* + *v*)*d*-1, where *d* = (*u* + *v*)*v* + *mu2.*

# KMIP Client and Server Implementation Conformance

## KMIP Client Implementation Conformance

An implementation is a conforming KMIP Client if the implementation meets the conditions specified in one or more client profiles specified in [KMIP-Prof].

A KMIP client implementation SHALL be a conforming KMIP Client.

If a KMIP client implementation claims support for a particular client profile, then the implementation SHALL conform to all normative statements within the clauses specified for that profile and for any subclauses to each of those clauses.

## KMIP Server Implementation Conformance

An implementation is a conforming KMIP Server if the implementation meets the conditions specified in one or more server profiles specified in [KMIP-Prof].

A KMIP server implementation SHALL be a conforming KMIP Server.

If a KMIP server implementation claims support for a particular server profile, then the implementation SHALL conform to all normative statements within the clauses specified for that profile and for any subclauses to each of those clauses.

1. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Rinkesh Bansal - IBM

Jeff Bartell - Individual

Gabriel Becker - KRYPTUS

Andre Bereza - KRYPTUS

Anthony Berglas - Cryptsoft Pty Ltd.

Mathias Bjorkqvist - IBM

Joseph Brand - Semper Fortis Solutions

Alan Brown - Thales e-Security

Andrew Byrne - Dell

Tim Chevalier - NetApp

Kenli Chong - QuintessenceLabs Pty Ltd.

Justin Corlett - Cryptsoft Pty Ltd.

Tony Cox - Cryptsoft Pty Ltd.

James Crossland - Northrop Grumman

Stephen Edwards - Semper Fortis Solutions

Stan Feather - Hewlett Packard Enterprise (HPE)

Indra Fitzgerald - Utimaco IS GmbH

Judith Furlong - Dell

Gary Gardner - Fornetix

Susan Gleeson - Oracle

Steve He - Thales e-Security

Christopher Hillier - Hewlett Packard Enterprise (HPE)

Tim Hudson - Cryptsoft Pty Ltd.

Nitin Jain - SafeNet, Inc.

Gershon Janssen - Individual

Mark Joseph - P6R, Inc

Paul Lechner - KeyNexus Inc

John Leiseboer - QuintessenceLabs Pty Ltd.

Jarrett Lu - Oracle

Jeff MacMillan - KeyNexus Inc

John Major - QuintessenceLabs Pty Ltd.

Cecilia Majorel - Quintessence Labs

Gabriel Mandaji - KRYPTUS

Jon Mentzell - Fornetix

Prashant Mestri - IBM

Kevin Mooney - Fornetix

Ladan Nekuii - Thales e-Security

Jason Novecosky - KeyNexus Inc

Matt O'reilly - Fornetix

Sanjay Panchal - IBM

Mahesh Paradkar - IBM

Steve Pate - Thales e-Security

Greg Pepus - Semper Fortis Solutions

Bruce Rich - Cryptsoft Pty Ltd.

Thad Roemer - Dyadic Security Ltd.

Thad Roemer - Unbound Tech

Greg Scott - Cryptsoft Pty Ltd.

Martin Shannon - QuintessenceLabs Pty Ltd.

Gerald Stueve - Fornetix

Jim Susoy - P6R, Inc

Jason Thatcher - Cryptsoft Pty Ltd.

Peter Tsai - Thales e-Security

Charles White - Fornetix

Steven Wierenga - Utimaco IS GmbH

Kyle Wuolle - KeyNexus Inc

1. Acronyms

The following abbreviations and acronyms are used in this document:

| **Item** | **Description** |
| --- | --- |
| 3DES | Triple Data Encryption Standard specified in ANSI X9.52 |
| AES | Advanced Encryption Standard specified in [FIPS197]FIPS 197 |
| ASN.1 | Abstract Syntax Notation One specified in ITU-T X.680 |
| BDK | Base Derivation Key specified in ANSI X9 TR-31 |
| CA | Certification Authority |
| CBC | Cipher Block Chaining |
| CCM | Counter with CBC-MAC specified in [SP800-38C] |
| CFB | Cipher Feedback specified in [SP800-38A] |
| CMAC | Cipher-based MAC specified in [SP800-38B] |
| CMC | Certificate Management Messages over CMS specified in [RFC5272] |
| CMP | Certificate Management Protocol specified in [RFC4210] |
| CPU | Central Processing Unit |
| CRL | Certificate Revocation List specified in [RFC5280] |
| CRMF | Certificate Request Message Format specified in [RFC4211] |
| CRT | Chinese Remainder Theorem |
| CTR | Counter specified in [SP800-38A] |
| CVK | Card Verification Key specified in ANSI X9 TR-31 |
| DEK | Data Encryption Key |
| DER | Distinguished Encoding Rules specified in ITU-T X.690 |
| DES | Data Encryption Standard specified in FIPS 46-3 |
| DH | Diffie-Hellman specified in ANSI X9.42 |
| DNS | Domain Name Server |
| DSA | Digital Signature Algorithm specified in FIPS 186-3 |
| DSKPP | Dynamic Symmetric Key Provisioning Protocol |
| ECB | Electronic Code Book |
| ECDH | Elliptic Curve Diffie-Hellman specified in [X9.63][SP800-56A] |
| ECDSA | Elliptic Curve Digital Signature Algorithm specified in [X9.62] |
| ECMQV | Elliptic Curve Menezes Qu Vanstone specified in [X9.63][SP800-56A] |
| FFC | Finite Field Cryptography |
| FIPS | Federal Information Processing Standard |
| GCM | Galois/Counter Mode specified in [SP800-38D] |
| GF | Galois field (or finite field) |
| HKDF | HMAC-based Extract-and-Expand Key Derivation Function (HKDF) **[RFC5869]** |
| HMAC | Keyed-Hash Message Authentication Code specified in [FIPS198-1][RFC2104] |
| HTTP | Hyper Text Transfer Protocol |
| HTTP(S) | Hyper Text Transfer Protocol (Secure socket) |
| IEEE | Institute of Electrical and Electronics Engineers |
| IETF | Internet Engineering Task Force |
| IP | Internet Protocol |
| IPsec | Internet Protocol Security |
| IV | Initialization Vector |
| KEK | Key Encryption Key |
| KMIP | Key Management Interoperability Protocol |
| MAC | Message Authentication Code |
| MKAC | EMV/chip card Master Key: Application Cryptograms specified in ANSI X9 TR-31 |
| MKCP | EMV/chip card Master Key: Card Personalization specified in ANSI X9 TR-31 |
| MKDAC | EMV/chip card Master Key: Data Authentication Code specified in ANSI X9 TR-31 |
| MKDN | EMV/chip card Master Key: Dynamic Numbers specified in ANSI X9 TR-31 |
| MKOTH | EMV/chip card Master Key: Other specified in ANSI X9 TR-31 |
| MKSMC | EMV/chip card Master Key: Secure Messaging for Confidentiality specified in X9 TR-31 |
| MKSMI | EMV/chip card Master Key: Secure Messaging for Integrity specified in ANSI X9 TR-31 |
| MD2 | Message Digest 2 Algorithm specified in [RFC1319] |
| MD4 | Message Digest 4 Algorithm specified in [RFC1320] |
| MD5 | Message Digest 5 Algorithm specified in [RFC1321] |
| NIST | National Institute of Standards and Technology |
| OAEP | Optimal Asymmetric Encryption Padding specified in [PKCS#1] |
| OFB | Output Feedback specified in [SP800-38A] |
| PBKDF2 | Password-Based Key Derivation Function 2 specified in [RFC2898] |
| PCBC | Propagating Cipher Block Chaining |
| PEM | Privacy Enhanced Mail specified in [RFC1421] |
| PGP | OpenPGP specified in **[RFC4880]** |
| PKCS | Public-Key Cryptography Standards |
| PKCS#1 | RSA Cryptography Specification Version 2.1 specified in [RFC3447] |
| PKCS#5 | Password-Based Cryptography Specification Version 2 specified in [RFC2898] |
| PKCS#8 | Private-Key Information Syntax Specification Version 1.2 specified in **[PKCS#8]** |
| PKCS#10 | Certification Request Syntax Specification Version 1.7 specified in **[RFC2986]** |
| PKCS#11 | Cryptographic Token Interface Standard |
| PKCS#12 | Personal Information Exchange Syntax |
| POSIX | Portable Operating System Interface |
| RFC | Request for Comments documents of IETF |
| RSA | Rivest, Shamir, Adelman (an algorithm) |
| RNG | Random Number Generator |
| SCEP | Simple Certificate Enrollment Protocol |
| SCVP | Server-based Certificate Validation Protocol |
| SHA | Secure Hash Algorithm specified in FIPS 180-2 |
| SP | Special Publication |
| SSL/TLS | Secure Sockets Layer/Transport Layer Security |
| S/MIME | Secure/Multipurpose Internet Mail Extensions |
| TDEA | see 3DES |
| TCP | Transport Control Protocol |
| TTLV | Tag, Type, Length, Value |
| URI | Uniform Resource Identifier |
| UTC | Coordinated Universal Time |
| UTF-8 | Universal Transformation Format 8-bit specified in [RFC3629] |
| XKMS | XML Key Management Specification |
| XML | Extensible Markup Language |
| XTS | XEX Tweakable Block Cipher with Ciphertext Stealing specified in [SP800-38E] |
| X.509 | Public Key Certificate specified in [RFC5280] |
| ZPK | PIN Block Encryption Key specified in ANSI X9 TR-31 |

1. List of Figures and Tables

[Figure 1: Cryptographic Object States and Transitions 86](#_Toc106287962)

[Table 1: Terminology 16](#_Toc106287963)

[Table 2: Minimum required Object attributes 20](#_Toc106287964)

[Table 3: Required User Aattributes 20](#_Toc106287965)

[Table 4: Required Group Attributes 21](#_Toc106287966)

[Table 5: Password Credential Structure 21](#_Toc106287967)

[Table 6: Password Credential Attributes 22](#_Toc106287968)

[Table 7: Device Credential Structure 22](#_Toc106287969)

[Table 8: Device Credential Attributes 22](#_Toc106287970)

[Table 9: One Time Password Credential Structure 22](#_Toc106287971)

[Table 10: One Time Password Credential Attributes 23](#_Toc106287972)

[Table 11: Hashed Password Credential Structure 23](#_Toc106287973)

[Table 12: Hashed Password Credential Attributes 23](#_Toc106287974)

[Table 13: Minimum required Certificate Object attributes 24](#_Toc106287975)

[Table 14: Certificate Object Structure 24](#_Toc106287976)

[Table 15: Minimum required Certificate Request Object attributes 24](#_Toc106287977)

[Table 16: Certificate Request Structure 24](#_Toc106287978)

[Table 17: Minimum required Opaque Object Object attributes 25](#_Toc106287979)

[Table 18: Opaque Object Structure 25](#_Toc106287980)

[Table 19: Minimum required PGP Key Object attributes 25](#_Toc106287981)

[Table 20: PGP Key Object Structure 25](#_Toc106287982)

[Table 21: Minimum required Private Key Object attributes 26](#_Toc106287983)

[Table 22: Private Key Object Structure 26](#_Toc106287984)

[Table 23: Minimum required Public Key Object attributes 26](#_Toc106287985)

[Table 24: Public Key Object Structure 26](#_Toc106287986)

[Table 25: Minimum required Secret Data Object attributes 27](#_Toc106287987)

[Table 26: Secret Data Object Structure 27](#_Toc106287988)

[Table 27: Minimum required Split Key Object attributes 27](#_Toc106287989)

[Table 28: Split Key Object Structure 28](#_Toc106287990)

[Table 29: Minimum required Symmetric Key Object attributes 28](#_Toc106287991)

[Table 30: Symmetric Key Object Structure 28](#_Toc106287992)

[Table 31: Key Block Cryptographic Algorithm & Length Description 29](#_Toc106287993)

[Table 32: Key Block Object Structure 29](#_Toc106287994)

[Table 33: Key Value Object Structure 30](#_Toc106287995)

[Table 34: Key Wrapping Data Structure Description 30](#_Toc106287996)

[Table 35: Key Wrapping Data Object Structure 31](#_Toc106287997)

[Table 36: Encryption Key Information Object Structure 31](#_Toc106287998)

[Table 37: MAC/Signature Key Information Object Structure 31](#_Toc106287999)

[Table 38: Key Material Object Structure for Transparent Symmetric Keys 32](#_Toc106288000)

[Table 39: Key Material Object Structure for Transparent DSA Private Keys 32](#_Toc106288001)

[Table 40: Key Material Object Structure for Transparent DSA Public Keys 32](#_Toc106288002)

[Table 41: Key Material Object Structure for Transparent RSA Private Keys 33](#_Toc106288003)

[Table 42: Key Material Object Structure for Transparent RSA Public Keys 33](#_Toc106288004)

[Table 43: Key Material Object Structure for Transparent DH Private Keys 33](#_Toc106288005)

[Table 44: Key Material Object Structure for Transparent DH Public Keys 34](#_Toc106288006)

[Table 45: Key Material Object Structure for Transparent EC Private Keys 34](#_Toc106288007)

[Table 46: Key Material Object Structure for Transparent EC Public Keys 34](#_Toc106288008)

[Table 47: Attribute Rules 36](#_Toc106288009)

[Table 48: Default Cryptographic Parameters 36](#_Toc106288010)

[Table 49: Activation Date Attribute 36](#_Toc106288011)

[Table 50: Activation Date Attribute Rules 37](#_Toc106288012)

[Table 51: Alternative Name Attribute Structure 37](#_Toc106288013)

[Table 52: Alternative Name Attribute Rules 37](#_Toc106288014)

[Table 53: Always Sensitive Attribute 37](#_Toc106288015)

[Table 54: Always Sensitive Attribute Rules 38](#_Toc106288016)

[Table 55: Application Specific Information Attribute 38](#_Toc106288017)

[Table 56: Application Specific Information Attribute Rules 38](#_Toc106288018)

[Table 57: Archive Date Attribute 39](#_Toc106288019)

[Table 58: Archive Date Attribute Rules 39](#_Toc106288020)

[Table 59: Certificate Attributes (Subject) 39](#_Toc106288021)

[Table 60: Certificate Attributes (Issuer) 40](#_Toc106288022)

[Table 61: Certificate Attribute Rules 40](#_Toc106288023)

[Table 62: Certificate Type Attribute 40](#_Toc106288024)

[Table 63: Certificate Type Attribute Rules 41](#_Toc106288025)

[Table 64: Certificate Length Attribute 41](#_Toc106288026)

[Table 65: Certificate Length Attribute Rules 41](#_Toc106288027)

[Table 66: Comment Attribute 41](#_Toc106288028)

[Table 67: Comment Rules 42](#_Toc106288029)

[Table 68: Compromise Date Attribute 42](#_Toc106288030)

[Table 69: Compromise Date Attribute Rules 42](#_Toc106288031)

[Table 70: Compromise Occurrence Date Attribute 42](#_Toc106288032)

[Table 71: Compromise Occurrence Date Attribute Rules 43](#_Toc106288033)

[Table 72: Contact Information Attribute 43](#_Toc106288034)

[Table 73: Contact Information Attribute Rules 43](#_Toc106288035)

[Table 74: Certify Counter Attribute 44](#_Toc106288036)

[Table 75: Certify Counter Attribute Rules 44](#_Toc106288037)

[Table 76: Decrypt Counter Attribute 44](#_Toc106288038)

[Table 77: Decrypt Counter Attribute Rules 44](#_Toc106288039)

[Table 78: Encrypt Counter Attribute 45](#_Toc106288040)

[Table 79: Encrypt Counter Attribute Rules 45](#_Toc106288041)

[Table 80: Sign Counter Attribute 45](#_Toc106288042)

[Table 81: Sign Counter Attribute Rules 45](#_Toc106288043)

[Table 82: Signature Verify Counter Attribute 45](#_Toc106288044)

[Table 83: Signature Verify Counter Attribute Rules 46](#_Toc106288045)

[Table 84: Credential Type Attribute 46](#_Toc106288046)

[Table 85: Credential Type Attribute Rules 46](#_Toc106288047)

[Table 86: Cryptographic Algorithm Attribute 46](#_Toc106288048)

[Table 87: Cryptographic Algorithm Attribute Rules 47](#_Toc106288049)

[Table 88: Cryptographic Domain Parameters Attribute Structure 47](#_Toc106288050)

[Table 89: Cryptographic Domain Parameters Attribute Rules 47](#_Toc106288051)

[Table 90: Cryptographic Length Attribute 48](#_Toc106288052)

[Table 91: Cryptographic Length Attribute Rules 48](#_Toc106288053)

[Table 92: Cryptographic Parameters Attribute Structure 49](#_Toc106288054)

[Table 93: Cryptographic Parameters Attribute Rules 50](#_Toc106288055)

[Table 94: Cryptographic Usage Mask Attribute 50](#_Toc106288056)

[Table 95: Cryptographic Usage Mask Attribute Rules 50](#_Toc106288057)

[Table 96: Deactivation Date Attribute 51](#_Toc106288058)

[Table 97: Deactivation Date Attribute Rules 51](#_Toc106288059)

[Table 98: Deactivation Reason Attribute 51](#_Toc106288060)

[Table 99: Deactivation Reason Attribute Rules 51](#_Toc106288061)

[Table 100: Description Attribute 52](#_Toc106288062)

[Table 101: Description Attribute Rules 52](#_Toc106288063)

[Table 102: Destroy Date Attribute 52](#_Toc106288064)

[Table 103: Destroy Date Attribute Rules 52](#_Toc106288065)

[Table 104: Digest Attribute Structure 53](#_Toc106288066)

[Table 105: Digest Attribute Rules 53](#_Toc106288067)

[Table 106: Digital Signature Algorithm Attribute 53](#_Toc106288068)

[Table 107: Digital Signature Algorithm Attribute Rules 54](#_Toc106288069)

[Table 108: Extractable Attribute 54](#_Toc106288070)

[Table 109: Extractable Attribute Rules 54](#_Toc106288071)

[Table 110: Fresh Attribute 54](#_Toc106288072)

[Table 111: Fresh Attribute Rules 55](#_Toc106288073)

[Table 112: Initial Date Attribute 55](#_Toc106288074)

[Table 113: Initial Date Attribute Rules 55](#_Toc106288075)

[Table 114: Key Format Type Attribute 56](#_Toc106288076)

[Table 115: Key Format Type Attribute Rules 56](#_Toc106288077)

[Table 116: Default Key Format Type, by Object 56](#_Toc106288078)

[Table 117: Key Part Identifier Attribute 56](#_Toc106288079)

[Table 118: Key Part Identifier Rules 57](#_Toc106288080)

[Table 119: Key Value Location Attribute 57](#_Toc106288081)

[Table 120: Key Value Location Attribute Rules 57](#_Toc106288082)

[Table 121: Key Value Present Attribute 57](#_Toc106288083)

[Table 122: Key Value Present Attribute Rules 58](#_Toc106288084)

[Table 123: Last Change Date Attribute 58](#_Toc106288085)

[Table 124: Last Change Date Attribute Rules 58](#_Toc106288086)

[Table 125: Lease Time Attribute 58](#_Toc106288087)

[Table 126: Lease Time Attribute Rules 59](#_Toc106288088)

[Table 127: Linked Object Identifier encoding descriptions 59](#_Toc106288089)

[Table 128: Certificate Link Attribute 60](#_Toc106288090)

[Table 129: Certificate Link Attribute Rules 60](#_Toc106288091)

[Table 130: Certificate Request Link Attribute 60](#_Toc106288092)

[Table 131: Certificate Request Link Attribute Rules 61](#_Toc106288093)

[Table 132: Child Link Attribute 61](#_Toc106288094)

[Table 133: Child Link Attribute Rules 61](#_Toc106288095)

[Table 134: Credential Link Attribute 62](#_Toc106288096)

[Table 135: Credential Link Attribute Rules 62](#_Toc106288097)

[Table 136: Derivation Base Object Link Attribute 62](#_Toc106288098)

[Table 137: Derivation Base Object Link Attribute Rules 62](#_Toc106288099)

[Table 138: Derived Object Link Attribute 63](#_Toc106288100)

[Table 139: Derived Object Link Attribute Rules 63](#_Toc106288101)

[Table 140: Group Link Attribute 63](#_Toc106288102)

[Table 141: Group Link Attribute Rules 63](#_Toc106288103)

[Table 142: Joined Split Key Parts Link Attribute 64](#_Toc106288104)

[Table 143: Joined Split Key Parts Link Attribute Rules 64](#_Toc106288105)

[Table 144: Next Link Attribute 64](#_Toc106288106)

[Table 145: Next Link Attribute Rules 64](#_Toc106288107)

[Table 146: Parent Link Attribute 65](#_Toc106288108)

[Table 147: Parent Link Attribute Rules 65](#_Toc106288109)

[Table 148: Password Link Attribute 65](#_Toc106288110)

[Table 149: Password Link Attribute Rules 65](#_Toc106288111)

[Table 150: PKCS#12 Certificate Link Attribute 66](#_Toc106288112)

[Table 151: PKCS#12 Certificate Link Attribute Rules 66](#_Toc106288113)

[Table 152: PKCS#12 Password Link Attribute 66](#_Toc106288114)

[Table 153: PKCS#12 Password Link Attribute Rules 66](#_Toc106288115)

[Table 154: Previous Link Attribute 67](#_Toc106288116)

[Table 155: Previous Link Attribute Rules 67](#_Toc106288117)

[Table 156: Private Key Link Attribute 67](#_Toc106288118)

[Table 157: Private Key Link Attribute Rules 67](#_Toc106288119)

[Table 158: Public Link Attribute 68](#_Toc106288120)

[Table 159: Public Key Link Attribute Rules 68](#_Toc106288121)

[Table 160: Replaced Object Link Attribute 68](#_Toc106288122)

[Table 161: Replaced Object Link Attribute Rules 69](#_Toc106288123)

[Table 162: Replacement Object Link Attribute 69](#_Toc106288124)

[Table 163: Replacement Object Link Attribute Rules 69](#_Toc106288125)

[Table 164: Split Key Base Link Attribute 69](#_Toc106288126)

[Table 165: Split Key Base Link Attribute Rules 70](#_Toc106288127)

[Table 166: Wrapping Key Link Attribute 70](#_Toc106288128)

[Table 167: Wrapping Key Link Attribute Rules 70](#_Toc106288129)

[Table 168: Name Attribute 70](#_Toc106288130)

[Table 169: Name Attribute Rules 71](#_Toc106288131)

[Table 170: Never Extractable Attribute 71](#_Toc106288132)

[Table 171: Never Extractable Attribute Rules 71](#_Toc106288133)

[Table 172 SP800-57 Key Type Attribute 71](#_Toc106288134)

[Table 173 SP800-57 Key Type Attribute Rules 72](#_Toc106288135)

[Table 174: Object Class Attribute 72](#_Toc106288136)

[Table 175: Object Class Attribute Rules 72](#_Toc106288137)

[Table 176: Object Type Attribute 72](#_Toc106288138)

[Table 177: Object Type Attribute Rules 73](#_Toc106288139)

[Table 178: Opaque Data Type Attribute 73](#_Toc106288140)

[Table 179: Opaque Data Type Attribute Rules 73](#_Toc106288141)

[Table 180: Original Creation Date Attribute 74](#_Toc106288142)

[Table 181: Original Creation Date Attribute Rules 74](#_Toc106288143)

[Table 182: OTP Counter Attribute 74](#_Toc106288144)

[Table 183: OTP Counter Attribute Rules 74](#_Toc106288145)

[Table 184: PKCS#12 Friendly Name Attribute 75](#_Toc106288146)

[Table 185: Friendly Name Attribute Rules 75](#_Toc106288147)

[Table 186: Process Start Date Attribute 75](#_Toc106288148)

[Table 187: Process Start Date Attribute Rules 75](#_Toc106288149)

[Table 188: Protect Stop Date Attribute 76](#_Toc106288150)

[Table 189: Protect Stop Date Attribute Rules 76](#_Toc106288151)

[Table 190: Protection Level Attribute 76](#_Toc106288152)

[Table 191: Protection Level Attribute Rules 76](#_Toc106288153)

[Table 192: Protection Period Attribute 77](#_Toc106288154)

[Table 193: Protection Period Attribute Rules 77](#_Toc106288155)

[Table 194: Protection Storage Mask 77](#_Toc106288156)

[Table 195: Protection Storage Mask Rules 77](#_Toc106288157)

[Table 196: Quantum Safe Attribute 77](#_Toc106288158)

[Table 197: Quantum Safe Attribute Rules 78](#_Toc106288159)

[Table 198: Random Number Generator Attribute 78](#_Toc106288160)

[Table 199: Random Number Generator Attribute Rules 79](#_Toc106288161)

[Table 200: Revocation Reason Attribute Structure 79](#_Toc106288162)

[Table 201: Revocation Reason Attribute Rules 79](#_Toc106288163)

[Table 202: Rotate Automatic Attribute 80](#_Toc106288164)

[Table 203: Rotate Automatic Attribute Rules 80](#_Toc106288165)

[Table 204: Rotate Date Attribute 80](#_Toc106288166)

[Table 205: Rotate Date Attribute Rules 80](#_Toc106288167)

[Table 206: Rotate Generation Attribute 80](#_Toc106288168)

[Table 207: Rotate Generation Attribute Rules 81](#_Toc106288169)

[Table 208: Rotate Interval Attribute 81](#_Toc106288170)

[Table 209: Rotate Interval Attribute Rules 81](#_Toc106288171)

[Table 210: Rotate Latest Attribute 81](#_Toc106288172)

[Table 211: Rotate Latest Attribute Rules 82](#_Toc106288173)

[Table 212: Rotate Name Attribute 82](#_Toc106288174)

[Table 213: Rotate Name Attribute Rules 82](#_Toc106288175)

[Table 214: Rotate Offset Attribute 82](#_Toc106288176)

[Table 215: Rotate Offset Attribute Rules 83](#_Toc106288177)

[Table 216: Sensitive Attribute 83](#_Toc106288178)

[Table 217: Sensitive Attribute Rules 83](#_Toc106288179)

[Table 218: Unique Identifier Attribute 84](#_Toc106288180)

[Table 219: Short Unique Identifier Attribute Rules 84](#_Toc106288181)

[Table 220: Split Key Polynomial Attribute 84](#_Toc106288182)

[Table 221: Split Key Polynomial Rules 84](#_Toc106288183)

[Table 222: Split Key Method Attribute 85](#_Toc106288184)

[Table 223: Split Key Method Rules 85](#_Toc106288185)

[Table 224: Split Key Parts Attribute 85](#_Toc106288186)

[Table 225: Split Key Parts Rules 85](#_Toc106288187)

[Table 226: Split Key Threshold Attribute 85](#_Toc106288188)

[Table 227: Split Key Threshold Rules 86](#_Toc106288189)

[Table 228: State Attribute 88](#_Toc106288190)

[Table 229: State Attribute Rules 88](#_Toc106288191)

[Table 230: Unique Identifier encoding descriptions 89](#_Toc106288192)

[Table 231: Unique Identifier Attribute 89](#_Toc106288193)

[Table 232: Unique Identifier Attribute Rules 89](#_Toc106288194)

[Table 233: Usage Limits Descriptions 90](#_Toc106288195)

[Table 234: Usage Limits Attribute Rules 90](#_Toc106288196)

[Table 235: Attribute Object Structure 91](#_Toc106288197)

[Table 236: X.509 Certificate Identifier Attribute Structure 91](#_Toc106288198)

[Table 237: X.509 Certificate Identifier Attribute Rules 91](#_Toc106288199)

[Table 238: X.509 Certificate Issuer Attribute Structure 92](#_Toc106288200)

[Table 239: X.509 Certificate Issuer Attribute Rules 92](#_Toc106288201)

[Table 240: X.509 Certificate Subject Attribute Structure 93](#_Toc106288202)

[Table 241: X.509 Certificate Subject Attribute Rules 93](#_Toc106288203)

[Table 242: Attributes Definition 94](#_Toc106288204)

[Table 243: Common Attributes Definition 94](#_Toc106288205)

[*Table 244: Private Key Attributes Definition* 94](#_Toc106288206)

[*Table 245: Public Key Attributes Definition* 95](#_Toc106288207)

[Table 246: Attribute Reference Definition 95](#_Toc106288208)

[Table 247: Current Attribute Definition 95](#_Toc106288209)

[Table 248: New Attribute Definition 95](#_Toc106288210)

[Table 249: Activate Request Payload 97](#_Toc106288211)

[Table 250: Activate Response Payload 97](#_Toc106288212)

[Table 251: Activate Errors 97](#_Toc106288213)

[Table 252: Add Attribute Request Payload 98](#_Toc106288214)

[Table 253: Add Attribute Response Payload 98](#_Toc106288215)

[Table 254: Add Attribute Errors 98](#_Toc106288216)

[Table 255: Adjust Attribute Request Payload 99](#_Toc106288217)

[Table 256: Adjust Attribute Response Payload 99](#_Toc106288218)

[Table 257: Adjust Attribute Errors 99](#_Toc106288219)

[Table 258: Archive Request Payload 99](#_Toc106288220)

[Table 259: Archive Response Payload 100](#_Toc106288221)

[Table 260: Archive Errors 100](#_Toc106288222)

[Table 261: Cancel Request Payload 100](#_Toc106288223)

[Table 262: Cancel Response Payload 100](#_Toc106288224)

[Table 263: Cancel Errors 100](#_Toc106288225)

[Table 264: Certify Request Payload 101](#_Toc106288226)

[Table 265: Certify Response Payload 101](#_Toc106288227)

[Table 266: Certify Errors 102](#_Toc106288228)

[Table 267: Check value description 102](#_Toc106288229)

[Table 268: Check Request Payload 103](#_Toc106288230)

[Table 269: Check Response Payload 103](#_Toc106288231)

[Table 270: Check Errors 103](#_Toc106288232)

[Table 271: Create Request Payload 104](#_Toc106288233)

[Table 272: Create Response Payload 104](#_Toc106288234)

[Table 273: Create Errors 104](#_Toc106288235)

[Table 274: Create Credential Request Payload 105](#_Toc106288236)

[Table 275: Create Credential Response Payload 105](#_Toc106288237)

[Table 276: Create Credential Errors 105](#_Toc106288238)

[Table 277: Create Group Request Payload 105](#_Toc106288239)

[Table 278: Create Group Response Payload 106](#_Toc106288240)

[Table 279: Create Group Errors 106](#_Toc106288241)

[Table 280: Create Key Pair Request Payload 107](#_Toc106288242)

[Table 281: Create Key Pair Response Payload 107](#_Toc106288243)

[Table 282: Create Key Pair Attribute Requirements 108](#_Toc106288244)

[Table 283: Create Key Pair Errors 108](#_Toc106288245)

[Table 284: Create Split Key Request Payload 109](#_Toc106288246)

[Table 285: Create Split Key Response Payload 109](#_Toc106288247)

[Table 286: Create Split Key Errors 109](#_Toc106288248)

[Table 287: Create User Request Payload 110](#_Toc106288249)

[Table 288: Create User Response Payload 110](#_Toc106288250)

[Table 289: Create User Errors 110](#_Toc106288251)

[Table 290: Deactivate Request Payload 110](#_Toc106288252)

[Table 291: Deactivate Response Payload 111](#_Toc106288253)

[Table 292: Deactivate Errors 111](#_Toc106288254)

[Table 293: Decrypt Request Payload 112](#_Toc106288255)

[Table 294: Decrypt Response Payload 112](#_Toc106288256)

[Table 295: Decrypt Errors 113](#_Toc106288257)

[Table 296: Delegated Login Request Payload 113](#_Toc106288258)

[Table 297: Delegated Login Response Payload 113](#_Toc106288259)

[Table 298: Delegated Login Errors 114](#_Toc106288260)

[Table 299: Delete Attribute Request Payload 114](#_Toc106288261)

[Table 300: Delete Attribute Response Payload 114](#_Toc106288262)

[Table 301: Delete Attribute Errors 114](#_Toc106288263)

[Table 302: Derive Key Request Payload 115](#_Toc106288264)

[Table 303: Derive Key Response Payload 115](#_Toc106288265)

[Table 304: Derive Key Errors 116](#_Toc106288266)

[Table 305: Destroy Request Payload 116](#_Toc106288267)

[Table 306: Destroy Response Payload 116](#_Toc106288268)

[Table 307: Destroy Errors 116](#_Toc106288269)

[Table 308: Discover Versions Request Payload 117](#_Toc106288270)

[Table 309: Discover Versions Response Payload 117](#_Toc106288271)

[Table 310: Discover Versions Errors 117](#_Toc106288272)

[Table 311: Encrypt Request Payload 118](#_Toc106288273)

[Table 312: Encrypt Response Payload 119](#_Toc106288274)

[Table 313: Encrypt Errors 119](#_Toc106288275)

[Table 314: Export Request Payload 120](#_Toc106288276)

[Table 315: Export Response Payload 120](#_Toc106288277)

[Table 316: Export Errors 120](#_Toc106288278)

[Table 317: Get Request Payload 121](#_Toc106288279)

[Table 318: Get Response Payload 121](#_Toc106288280)

[Table 319: Get Errors 122](#_Toc106288281)

[Table 320: Get Attributes Request Payload 122](#_Toc106288282)

[Table 321: Get Attributes Response Payload 122](#_Toc106288283)

[Table 322: Get Attributes Errors 123](#_Toc106288284)

[Table 323: Get Attribute List Request Payload 123](#_Toc106288285)

[Table 324: Get Attribute List Response Payload 123](#_Toc106288286)

[Table 325: Get Attribute List Errors 123](#_Toc106288287)

[Table 326: Get Constraints Request Payload 124](#_Toc106288288)

[Table 327: Get Constraints Response Payload 124](#_Toc106288289)

[Table 328: Get Constraints Errors 124](#_Toc106288290)

[Table 329: Get Usage Allocation Request Payload 125](#_Toc106288291)

[Table 330: Get Usage Allocation Response Payload 125](#_Toc106288292)

[Table 331: Get Usage Allocation Errors 125](#_Toc106288293)

[Table 332: Hash Request Payload 126](#_Toc106288294)

[Table 333: Hash Response Payload 126](#_Toc106288295)

[Table 334: HASH Errors 126](#_Toc106288296)

[Table 335: Import Request Payload 127](#_Toc106288297)

[Table 336: Import Response Payload 127](#_Toc106288298)

[Table 337: Import Errors 127](#_Toc106288299)

[Table 338: Interop Request Payload 128](#_Toc106288300)

[Table 339: Interop Response Payload 128](#_Toc106288301)

[Table 340: Interop Errors 128](#_Toc106288302)

[Table 341: Join Split Key Request Payload 129](#_Toc106288303)

[Table 342: Join Split Key Response Payload 129](#_Toc106288304)

[Table 343: Join Split Key Errors 129](#_Toc106288305)

[Table 344: Locate Request Payload 131](#_Toc106288306)

[Table 345: Locate Response Payload 131](#_Toc106288307)

[Table 346: Locate Errors 132](#_Toc106288308)

[Table 347: Log Request Payload 132](#_Toc106288309)

[Table 348: Log Response Payload 132](#_Toc106288310)

[Table 349: Log Errors 132](#_Toc106288311)

[Table 350: Login Request Payload 133](#_Toc106288312)

[Table 351: Login Response Payload 133](#_Toc106288313)

[Table 352: Login Errors 133](#_Toc106288314)

[Table 353: Logout Request Payload 133](#_Toc106288315)

[Table 354: Logout Response Payload 133](#_Toc106288316)

[Table 355: Logout Errors 134](#_Toc106288317)

[Table 356: MAC Request Payload 134](#_Toc106288318)

[Table 357: MAC Response Payload 135](#_Toc106288319)

[Table 358: MAC Errors 135](#_Toc106288320)

[Table 359: MAC Verify Request Payload 136](#_Toc106288321)

[Table 360: MAC Verify Response Payload 136](#_Toc106288322)

[Table 361: MAC Verify Errors 137](#_Toc106288323)

[Table 362: Modify Attribute Request Payload 137](#_Toc106288324)

[Table 363: Modify Attribute Response Payload 137](#_Toc106288325)

[Table 364: Modify Attribute Errors 138](#_Toc106288326)

[Table 365: Obliterate Request Payload 138](#_Toc106288327)

[Table 366: Obliterate Response Payload 138](#_Toc106288328)

[Table 367: Obliterate Errors 138](#_Toc106288329)

[Table 368: Obtain Lease Request Payload 139](#_Toc106288330)

[Table 369: Obtain Lease Response Payload 139](#_Toc106288331)

[Table 370: Obtain Lease Errors 139](#_Toc106288332)

[Table 371: Ping Request Payload 139](#_Toc106288333)

[Table 372: Ping Response Payload 140](#_Toc106288334)

[Table 373: PKCS#11 Request Payload 140](#_Toc106288335)

[Table 374: PKCS#11 Response Payload 140](#_Toc106288336)

[Table 375: PKCS#11 Errors 141](#_Toc106288337)

[Table 376: Poll Request Payload 141](#_Toc106288338)

[Table 377: Poll Errors 141](#_Toc106288339)

[Table 378: Process Request Payload 142](#_Toc106288340)

[Table 379: Process Response Payload 142](#_Toc106288341)

[Table 380: Process Request Errors 142](#_Toc106288342)

[Table 381: Query Functions 143](#_Toc106288343)

[Table 382: Query Request Payload 144](#_Toc106288344)

[Table 383: Query Response Payload 144](#_Toc106288345)

[Table 384: Query Errors 145](#_Toc106288346)

[Table 385: Query Asynchronous Requests Request Payload 145](#_Toc106288347)

[Table 386: PKCS#11 Response Payload 145](#_Toc106288348)

[Table 387: Query Asynchronous Requests Errors 145](#_Toc106288349)

[Table 388: Recover Request Payload 146](#_Toc106288350)

[Table 389: Recover Response Payload 146](#_Toc106288351)

[Table 390: Recover Errors 146](#_Toc106288352)

[Table 391: Register Request Payload 147](#_Toc106288353)

[Table 392: Register Response Payload 147](#_Toc106288354)

[Table 393: Register Attribute Requirements 147](#_Toc106288355)

[Table 394: Register Errors 148](#_Toc106288356)

[Table 395: Revoke Request Payload 148](#_Toc106288357)

[Table 396: Revoke Response Payload 148](#_Toc106288358)

[Table 397: Revoke Errors 149](#_Toc106288359)

[Table 398: Computing New Dates from Offset during Re-certify 149](#_Toc106288360)

[Table 399: Re-certify Attribute Requirements 150](#_Toc106288361)

[Table 400: Re-certify Request Payload 150](#_Toc106288362)

[Table 401: Re-certify Response Payload 150](#_Toc106288363)

[Table 402: Re-certify Errors 151](#_Toc106288364)

[Table 403: Computing New Dates from Offset during Re-key 151](#_Toc106288365)

[Table 404: Re-key Attribute Requirements 152](#_Toc106288366)

[Table 405: Re-key Request Payload 152](#_Toc106288367)

[Table 406: Re-key Response Payload 152](#_Toc106288368)

[Table 407: Re-key Errors 153](#_Toc106288369)

[Table 408: Computing New Dates from Offset during Re-key Key Pair 153](#_Toc106288370)

[Table 409: Re-key Key Pair Attribute Requirements 154](#_Toc106288371)

[Table 410: Re-key Key Pair Request Payload 154](#_Toc106288372)

[Table 411: Re-key Key Pair Response Payload 155](#_Toc106288373)

[Table 412: Re-key Key Pair Errors 155](#_Toc106288374)

[Table 413: Re-Provision Request Payload 156](#_Toc106288375)

[Table 414: Re-Provision Response Payload 156](#_Toc106288376)

[Table 415: RNG Retrieve Errors 156](#_Toc106288377)

[Table 416: RNG Retrieve Request Payload 156](#_Toc106288378)

[Table 417: RNG Retrieve Response Payload 156](#_Toc106288379)

[Table 418: RNG Retrieve Errors 157](#_Toc106288380)

[Table 419: RNG Seed Request Payload 157](#_Toc106288381)

[Table 420: RNG Seed Response Payload 157](#_Toc106288382)

[Table 421: RNG Seed Errors 157](#_Toc106288383)

[Table 422: Set Attribute Request Payload 158](#_Toc106288384)

[Table 423: Set Attribute Response Payload 158](#_Toc106288385)

[Table 424: Set Attribute Errors 158](#_Toc106288386)

[Table 425: Set Constraints Request Payload 158](#_Toc106288387)

[Table 426: Set Constraints Response Payload 159](#_Toc106288388)

[Table 427: Set Constraints Errors 159](#_Toc106288389)

[Table 428: Set Defaults Request Payload 159](#_Toc106288390)

[Table 429: Set Defaults Response Payload 159](#_Toc106288391)

[Table 430: Set Defaults Errors 159](#_Toc106288392)

[Table 431: Set Endpoint Role Request Payload 160](#_Toc106288393)

[Table 432: Set Endpoint Role Response Payload 160](#_Toc106288394)

[Table 433: Set Endpoint Role Errors 160](#_Toc106288395)

[Table 434: Sign Request Payload 161](#_Toc106288396)

[Table 435: Sign Response Payload 162](#_Toc106288397)

[Table 436: Sign Errors 162](#_Toc106288398)

[Table 437: Signature Verify Request Payload 163](#_Toc106288399)

[Table 438: Signature Verify Response Payload 164](#_Toc106288400)

[Table 439: Signature Verify Errors 164](#_Toc106288401)

[Table 440: Validate Request Payload 165](#_Toc106288402)

[Table 441: Validate Response Payload 165](#_Toc106288403)

[Table 442: Validate Errors 165](#_Toc106288404)

[Table 443: Discover Versions Request Payload 166](#_Toc106288405)

[Table 444: Discover Versions Errors 166](#_Toc106288406)

[Table 445: Notify Message Errors 167](#_Toc106288407)

[Table 446: Put Errors 168](#_Toc106288408)

[Table 447: Query Request Payload 169](#_Toc106288409)

[Table 448: Query Errors 170](#_Toc106288410)

[Table 449: Set Endpoint Role Request Payload 170](#_Toc106288411)

[Table 450: Set Endpoint Role Response Payload 170](#_Toc106288412)

[Table 451: Set Endpoint Role Errors 171](#_Toc106288413)

[Table 452 Asynchronous Correlation Values Structure 172](#_Toc106288414)

[Table 453 Asynchronous Request Structure 172](#_Toc106288415)

[Table 454 Authenticated Encryption Additional Data 172](#_Toc106288416)

[Table 455 Authenticated Encryption Tag 172](#_Toc106288417)

[Table 456: Capability Information Structure 173](#_Toc106288418)

[Table 457: Constraint Structure 173](#_Toc106288419)

[Table 458: Constraints Structure 173](#_Toc106288420)

[Table 459: Correlation Value Structure 174](#_Toc106288421)

[Table 460: Credential Information Structure 174](#_Toc106288422)

[Table 461: Data encoding descriptions 174](#_Toc106288423)

[Table 462: Data 174](#_Toc106288424)

[Table 463: Data Length Structure 174](#_Toc106288425)

[Table 464: Defaults Information Structure 175](#_Toc106288426)

[Table 465: Derivation Parameters Structure 175](#_Toc106288427)

[Table 466: Extension Information Structure 176](#_Toc106288428)

[Table 467: Final Indicator Structure 176](#_Toc106288429)

[Table 468: Interop Function Structure 176](#_Toc106288430)

[Table 469: Interop Function Structure 177](#_Toc106288431)

[Table 470: Init Indicator Structure 177](#_Toc106288432)

[Table 471: Key Wrapping Specification Object Structure 178](#_Toc106288433)

[Table 472: Log Message Structure 178](#_Toc106288434)

[Table 473: MAC Data Structure 178](#_Toc106288435)

[Table 474: Objects Structure 178](#_Toc106288436)

[Table 475: Object Defaults Structure 179](#_Toc106288437)

[Table 476: Object Groups Structure 179](#_Toc106288438)

[Table 477: ObjectTypes Structure 179](#_Toc106288439)

[Table 478: Operations Structure 179](#_Toc106288440)

[Table 479: PKCS#11 Function Structure 179](#_Toc106288441)

[Table 480: PKCS#11 Input Parameters Structure 180](#_Toc106288442)

[Table 481: PKCS#11 Interface Structure 180](#_Toc106288443)

[Table 482: PKCS#11 Output Parameters Structure 180](#_Toc106288444)

[Table 483: PKCS#11 Return Code Structure 180](#_Toc106288445)

[Table 484: Profile Information Structure 181](#_Toc106288446)

[Table 485: Profile Version Structure 181](#_Toc106288447)

[Table 486: Protection Storage Mask Structure 181](#_Toc106288448)

[Table 487: Right Structure 182](#_Toc106288449)

[Table 488: Rights Structure 182](#_Toc106288450)

[Table 489: RNG Parameters Structure 182](#_Toc106288451)

[Table 490: Server Information Structure 183](#_Toc106288452)

[Table 491: Signature Data Structure 183](#_Toc106288453)

[Table 492: Ticket Structure 183](#_Toc106288454)

[Table 493: Usage limits Structure 184](#_Toc106288455)

[Table 494: Validation Information Structure 184](#_Toc106288456)

[Table 495: Request Message Structure 185](#_Toc106288457)

[Table 496: Request Header Structure 186](#_Toc106288458)

[Table 497: Request Batch Item Structure 186](#_Toc106288459)

[Table 498: Response Message Structure 186](#_Toc106288460)

[Table 499: Response Header Structure 187](#_Toc106288461)

[Table 500: Response Batch Item Structure 187](#_Toc106288462)

[Table 501: Asynchronous Correlation Value in Response Batch Item 188](#_Toc106288463)

[Table 502: Asynchronous Indicator in Message Request Header 188](#_Toc106288464)

[Table 503: Attestation Capable Indicator in Message Request Header 188](#_Toc106288465)

[Table 504: Authentication Structure in Message Header 189](#_Toc106288466)

[Table 505: Batch Error Continuation Option in Message Request Header 189](#_Toc106288467)

[Table 506: Batch Item in Message 189](#_Toc106288468)

[Table 507: Client Correlation Value in Message Request Header 189](#_Toc106288469)

[Table 508: Server Correlation Value in Message Request Header 189](#_Toc106288470)

[Table 509: Credential Object Structure 190](#_Toc106288471)

[Table 510: Credential Value Structure for the Username and Password Credential 190](#_Toc106288472)

[Table 511: Credential Value Structure for the Device Credential 190](#_Toc106288473)

[Table 512: Credential Value Structure for the Attestation Credential 191](#_Toc106288474)

[Table 513: Credential Value Structure for the One Time Password Credential 191](#_Toc106288475)

[Table 514: Credential Value Structure for the Hashed Password Credential 191](#_Toc106288476)

[Table 515: Credential Value Structure for the Ticket 191](#_Toc106288477)

[Table 516: Maximum Response Size in Message Request Header 192](#_Toc106288478)

[Table 517: Message Extension Structure in Batch Item 192](#_Toc106288479)

[Table 518: Nonce Structure 192](#_Toc106288480)

[Table 519: Operation in Batch Item 193](#_Toc106288481)

[Table 520: Protocol Version Structure in Message Header 193](#_Toc106288482)

[Table 521: Result Message in Response Batch Item 193](#_Toc106288483)

[Table 522: Result Reason in Response Batch Item 193](#_Toc106288484)

[Table 523: Result Status in Response Batch Item 194](#_Toc106288485)

[Table 524: Time Stamp in Message Header 194](#_Toc106288486)

[Table 525: Item Types 196](#_Toc106288487)

[Table 526: Allowed Item Length Values 196](#_Toc106288488)

[Table 527: Adjustment Type Descriptions 198](#_Toc106288489)

[Table 528: Adjustment Type Enumeration 198](#_Toc106288490)

[Table 529: Alternative Name Type Enumeration 198](#_Toc106288491)

[Table 530: Asynchronous Indicator Descriptions 199](#_Toc106288492)

[Table 531: Asynchronous Indicator Enumeration 199](#_Toc106288493)

[Table 532: Attestation Type Enumeration 199](#_Toc106288494)

[Table 533: Batch Error Continuation Option Descriptions 200](#_Toc106288495)

[Table 534: Batch Error Continuation Option Enumeration 200](#_Toc106288496)

[Table 535: Block Cipher Mode Enumeration 201](#_Toc106288497)

[Table 536: Cancellation Result Enumeration Descriptions 201](#_Toc106288498)

[Table 537: Cancellation Result Enumeration 201](#_Toc106288499)

[Table 538: Certificate Request Type Enumeration 202](#_Toc106288500)

[Table 539: Certificate Type Enumeration 202](#_Toc106288501)

[Table 540: Client Registration Method Enumeration Descriptions 202](#_Toc106288502)

[Table 541: Client Registration Method Enumerations 203](#_Toc106288503)

[Table 542: Credential Type Enumeration 203](#_Toc106288504)

[Table 543: Cryptographic Algorithm Enumeration 205](#_Toc106288505)

[Table 544: Data Enumeration 205](#_Toc106288506)

[Table 545: Deactivation Reason Code Enumeration 205](#_Toc106288507)

[Table 546: Derivation Method Enumeration Descriptions 206](#_Toc106288508)

[Table 547: Derivation Method Enumeration 206](#_Toc106288509)

[Table 548: Destroy Action Enumeration 207](#_Toc106288510)

[Table 549: Digital Signature Algorithm Enumeration 207](#_Toc106288511)

[Table 550: DRGB Algorithm Enumeration 208](#_Toc106288512)

[Table 551: Encoding Option Description 208](#_Toc106288513)

[Table 552: Encoding Option Enumeration 208](#_Toc106288514)

[Table 553: Endpoint Role Description 208](#_Toc106288515)

[Table 554: Endpoint Role Enumeration 209](#_Toc106288516)

[Table 555: FIPS186 Variation Enumeration 209](#_Toc106288517)

[Table 556: Hashing Algorithm Enumeration 210](#_Toc106288518)

[*Table 557: Interop Function Descriptions* 210](#_Toc106288519)

[Table 558: Interop Function Enumeration 211](#_Toc106288520)

[*Table 559: Item Type Descriptions* 211](#_Toc106288521)

[Table 560: Item Type Enumeration 212](#_Toc106288522)

[Table 561: Key Compression Type Enumeration values 212](#_Toc106288523)

[Table 562: Key Format Types Description 212](#_Toc106288524)

[Table 563: Key Format Type Enumeration 213](#_Toc106288525)

[Table 564: Key Role Type Enumeration 214](#_Toc106288526)

[Table 565: Key Value Location Type Enumeration 214](#_Toc106288527)

[Table 566: Key Wrap Enumeration 214](#_Toc106288528)

[Table 567: Name Type Enumeration 215](#_Toc106288529)

[Table 568: NIST Key Type Enumeration 216](#_Toc106288530)

[Table 569: Object Class Enumeration 216](#_Toc106288531)

[Table 570: Object Type Enumeration 216](#_Toc106288532)

[Table 571: Opaque Data Type Enumeration 217](#_Toc106288533)

[Table 572: Operation Enumeration 218](#_Toc106288534)

[Table 573: OTP Algorithm Description 219](#_Toc106288535)

[Table 574: OTP Algorithm Enumeration 219](#_Toc106288536)

[Table 575: Padding Method Enumeration 219](#_Toc106288537)

[Table 576: Processing Stage Enumeration 220](#_Toc106288538)

[Table 577: Profile Name Enumeration 221](#_Toc106288539)

[Table 578: Protection Level Enumeration 221](#_Toc106288540)

[Table 579: Put Function Enumeration 221](#_Toc106288541)

[Table 580: Query Function Enumeration 222](#_Toc106288542)

[Table 581: Recommended Curve Enumeration for ECDSA, ECDH, and ECMQV 224](#_Toc106288543)

[Table 582: Result Reason Encoding Descriptions 228](#_Toc106288544)

[Table 583: Result Reason Enumeration 230](#_Toc106288545)

[Table 584: Result Status Enumeration 230](#_Toc106288546)

[Table 585: Revocation Reason Code Enumeration 231](#_Toc106288547)

[Table 586: RNG Algorithm Enumeration 231](#_Toc106288548)

[Table 587: RNG Mode Enumeration 231](#_Toc106288549)

[Table 588: Secret Data Type Enumeration 231](#_Toc106288550)

[Table 589: Shredding Algorithm Enumeration 232](#_Toc106288551)

[Table 590: Split Key Method Enumeration 232](#_Toc106288552)

[Table 591: Split Key Polynomial Enumeration 232](#_Toc106288553)

[Table 592: State Enumeration 232](#_Toc106288554)

[Table 593: Tag Enumeration 245](#_Toc106288555)

[Table 594: Ticket Type Enumeration 245](#_Toc106288556)

[Table 595: Unique Identifier Enumeration 246](#_Toc106288557)

[Table 596: Unwrap Mode Enumeration 246](#_Toc106288558)

[Table 597: Usage Limits Unit Enumeration 247](#_Toc106288559)

[Table 598: Validity Indicator Enumeration 247](#_Toc106288560)

[Table 599: Key Wrapping Methods Description 247](#_Toc106288561)

[Table 600: Wrapping Method Enumeration 248](#_Toc106288562)

[Table 601: Validation Authority Type Enumeration 248](#_Toc106288563)

[Table 602: Validation Type Enumeration 248](#_Toc106288564)

[Table 603: Cryptographic Usage Masks Description 250](#_Toc106288565)

[Table 604: Cryptographic Usage Mask enumerations 250](#_Toc106288566)

[Table 605: Protection Storage Mask enumerations 251](#_Toc106288567)

[Table 606: Storage Status Mask enumerations 251](#_Toc106288568)

[Table 607: Object Class Mask enumerations 251](#_Toc106288569)

1. Revision History

| **Revision** | **Date** | **Editor** | **Changes Made** |
| --- | --- | --- | --- |
| WD01 | 04 May 2020 | Tony Cox | Initial Working Draft |
| WD02 | 18 June 2020 | Tony Cox | - Added approved spec-delta from Name Representation proposal  - Added approved spec-delta from Link Representation - Added new section for Links and references |
| WD03 | 14 August 2020 | Tony Cox | - Added approved spec-delta from Miscellaneous (batch) proposal  - Added approved spec-delta from Automation Architecture proposal  - Added approved spec-delta from Split Key (Polynomial) Proposal  - Added approved spec-delta from Name Lifecycle Proposal  - Added approved spec-delta from Grouping Objects Proposal  - Moved Links and References section |
| WD04 | 28 August 2020 | Tony Cox | - Added approved spec-delta from Obliterate proposal  - Removed redundant content from “Link” (result of adding “Links” attribute) |
| WD05 | 10 August 2020 | Tony Cox | - General editorial cleanup  - Added missing table references  - Added approved spec-delta from Hierarchy of Groups Proposal |
| WD06 | 6 October 2020 | Tony Cox | - Added approved spec-delta from User Credential Proposal |
| WD07 | 23 October 2020 | Tony Cox | - Corrected formatting error introduced in WD06  - Added approved spec-delta from Client Mutual Authentication Proposal  - Added approved spec-delta updated from User Credential Proposal |
| WD08 | 18 December 2020 | Tony Cox | - Amended Derive Key operation (previous text was incorrect for some key derivation processes)  - Added missing Unique Identifier Enumerations  - Corrected broken reference in §9.6  - Added approved spec-delta from Revoke & Deactivate Proposal |
| WD09 | 4 January 2021 | Tony Cox | - Added Group Link enumeration  - Added Tag for Split Key Polynomial |
| WD10 | 10 March 2021 | Tony Cox | - Clarified Ephemeral payload return  - Corrected error in Object Class  - Corrected error in Object Class Attribute  - Other minor corrections |
| WD11 | 24 March 2021 | Tony Cox | - Added Object Class Mask  - Increased reserved space for Protection Storage Masks  - Increased reserved space for Storage Masks |
| WD12 | 15 July 2021 | Tony Cox | - Corrected Certificate Attributes  - Added missing Link Attributes inc. tags  - Certificate Request Link  - Joined Split Key Link  - Split Key Base Link  - Corrected Object Class references |
| WD13 | 13 August 2021 | Tony Cox | - Corrected unique identifier referencing  - Corrected Split Key Base Link syntax  - Corrected errors in Extractable and Sensitive attributes  - Added Counters  - Added client-provided UUID |
| WD14 | 16 June 2022 | Greg Scott | * Update Chair and Editor details * Removed Counter Type Enumeration * Removed Link Type Enumeration * Updated all old Link references to use the new Link attributes * Align prose in Re-certify with Certify for updates related to the Certificate Request object | |
| WD15 | 7 September 2023 | Greg Scott | * Deactivation Reason and Revocation Reason can be omitted * Clarification on Hashed Password initialization * Ephemeral Enumeration added * IDPlaceHolder changes added * Credential Type is now required on Create Credential for consistency * Added clarifying statement to Cryptographic Usage Mask noting that State must be checked | |
| WD16 | 23 August 2024 | Greg Scott | * Added NIST FIPS203, 204, 205 algorithms * Additional updates from Sept 2023 testing   + Unique Identifier UUID recommendation   + Lease Time is a mandatory attribute   + Unique Identifier expansion on Identifier/Reference/NameReference   + Attribute Name do-not-trim note   + Note Interop operation usage of an Interop Identifier of “\*” for general cleanup during interoperability testing | |

1. Notices

Copyright © OASIS Open 2024. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full [Policy](https://www.oasis-open.org/policies-guidelines/ipr/) may be found at the OASIS website: [<https://www.oasis-open.org/policies-guidelines/ipr/>].

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. OASIS AND ITS MEMBERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THIS DOCUMENT OR ANY PART THEREOF.

As stated in the OASIS IPR Policy, the following three paragraphs in brackets apply to OASIS Standards Final Deliverable documents (Committee Specifications, OASIS Standards, or Approved Errata).

[OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Standards Final Deliverable, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this deliverable.]

[OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this OASIS Standards Final Deliverable by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this OASIS Standards Final Deliverable. OASIS may include such claims on its website, but disclaims any obligation to do so.]

[OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this OASIS Standards Final Deliverable or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Standards Final Deliverable, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.]

The name "OASIS" is a trademark of [OASIS](https://www.oasis-open.org/), the owner and developer of this document, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, documents, while reserving the right to enforce its marks against misleading uses. Please see <https://www.oasis-open.org/policies-guidelines/trademark/> for above guidance.