



Symmetric Key Services Markup Language (SKSML) Version 1.0

Public Review Draft 01

22 July 2008

Specification URIs:

This Version:

<http://docs.oasis-open.org/ekmi/sksml/v1.0/pr01/SKSML-1.0-Specification.html>
<http://docs.oasis-open.org/ekmi/sksml/v1.0/pr01/SKSML-1.0-Specification.odt>
<http://docs.oasis-open.org/ekmi/sksml/v1.0/pr01/SKSML-1.0-Specification.pdf>

Previous Version:

None

Latest Version:

<http://docs.oasis-open.org/ekmi/sksml/v1.0/SKSML-1.0-Specification.html>
<http://docs.oasis-open.org/ekmi/sksml/v1.0/SKSML-1.0-Specification.odt>
<http://docs.oasis-open.org/ekmi/sksml/v1.0/SKSML-1.0-Specification.pdf>

Latest Approved Version:

None

Technical Committee:

OASIS Enterprise Key Management Infrastructure (EKMI) TC

Chair(s):

Arshad Noor, StrongAuth, Inc. (arshad.noor@strongauth.com)

Editor(s):

Allen Schaaf (netsecurity@sound-by-design.com)

Related Work:

This specification replaces or supercedes:

- None

This specification is related to:

- Advanced Encryption Standard (AES) - NIST FIPS 197 - <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
- Simple Object Access Protocol (SOAP) - W3C Recommendation 08 May 2000. <http://www.w3.org/TR/soap/>
- XML Encryption - W3C Recommendation 10 Dec 2002. <http://www.w3.org/TR/xmlenc-core/>

- 34 • XML Signature - W3C Recommendation 12 Feb 2002. <http://www.w3.org/TR/xmlsig-core/>
35 • Web Services Security - SOAP Message Security 1.0 - OASIS Standard 200401, March 2004 -
36 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>

37 **Declared XML Namespace(s):**

38 <http://docs.oasis-open.org/ekmi/2008/01>

39 **Abstract:**

40 This normative specification defines the first (1.0) version of the Symmetric Key Services Markup
41 Language (SKSML), an XML-based messaging protocol, by which applications executing on computing
42 devices may request and receive symmetric key-management services from centralized key-
43 management servers, securely, over networks. Applications using SKSML are expected to either
44 implement the SKSML protocol, or use a software library – called the Symmetric Key Client Library
45 (SKCL) – that implements this protocol. SKSML messages are transported within a SOAP layer,
46 protected by a Web Services Security (WSS) header and can be used over standard HTTP securely.

47 **Status:**

48 This document was last revised by the EKMI TC as of the above date. The level of approval is also
49 listed above. Check the "Latest Version" or "Latest Approved Version" location noted above for possible
50 later revisions of this document.

51 Technical Committee members should send comments on this specification to the Technical
52 Committee's email list. Others should send comments to the Technical Committee by using the "Send A
53 Comment" button on the Technical Committee's web page at [http://www.oasis-](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ekmi)
54 [open.org/committees/tc_home.php?wg_abbrev=ekmi](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ekmi)

55 For information on whether any patents have been disclosed that may be essential to implementing this
56 specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights
57 section of the Technical Committee web page (<http://www.oasis-open.org/committees/ekmi/ipr.php>).

58 The non-normative errata page for this specification is located at [http://www.oasis-open.org/committees/](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ekmi/)
59 [tc_home.php?wg_abbrev=ekmi/](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ekmi/).

Notices

60

61 Copyright © OASIS® 2008. All Rights Reserved.

62 All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property
63 Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

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

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

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

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

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

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

97 The names "OASIS" and "SKSML" are trademarks of OASIS, the owner and developer of this specification, and
98 should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and
99 implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses.
100 Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

101

102

103 Table of Contents

104	1 Introduction.....	6
105	1.1 Terminology.....	6
106	1.2 Glossary.....	6
107	1.3 Normative References.....	7
108	2 Background (non-normative).....	8
109	2.1 Requirements (non-normative).....	9
110	3 Examples of use of SKSML (non-normative).....	11
111	3.1 Request for a new symmetric key.....	11
112	3.2 Response with a new symmetric key.....	13
113	3.3 Request for an existing symmetric key.....	17
114	3.4 Response with an existing symmetric key.....	18
115	3.5 Request for a new symmetric key of a specific KeyClass.....	18
116	3.6 Response with a new symmetric key of a specific KeyClass.....	18
117	3.7 Request for multiple new symmetric keys.....	19
118	3.8 Response with multiple new symmetric keys.....	20
119	3.9 Response with an SKS error.....	24
120	3.10 Response with symmetric keys and errors.....	25
121	3.11 Request for a symmetric key-caching policy.....	27
122	3.12 Response with a symmetric key-caching policy (1).....	29
123	3.13 Response with a symmetric key-caching policy (2).....	30
124	3.14 Response with multiple symmetric key-caching policies (3).....	32
125	4 Specification.....	36
126	4.1 Element <SymkeyRequest>.....	36
127	4.2 Element <GlobalKeyID>.....	38
128	4.3 Element <KeyClasses> and <KeyClass>.....	39
129	4.4 Element <SymkeyResponse>.....	41
130	4.5 Element <Symkey>.....	42
131	4.6 Element <SymkeyError>.....	44
132	4.7 Element <KeyUsePolicy>.....	46
133	4.8 Type TwoPartIDType.....	48
134	4.9 Element <KeyAlgorithm>.....	49
135	4.10 Element <KeySize>.....	50
136	4.11 Element <Status>.....	51
137	4.12 Element <Permissions>.....	52
138	4.13 Element <PermittedApplications> and <PermittedApplication>.....	58
139	4.14 Element <PermittedDates> and <PermittedDate>.....	61
140	4.15 Element <PermittedDays> and <PermittedDay>.....	63
141	4.16 Element <PermittedDuration>.....	64
142	4.17 Element <PermittedLevels> and <PermittedLevel>.....	65
143	4.18 Element <PermittedLocations> and <PermittedLocation>.....	67
144	4.19 Element <PermittedNumberOfTransactions>.....	69

145	4.20 Element <PermittedTimes> and <PermittedTime>.....	70
146	4.21 Element <PermittedUses> and <PermittedUse>.....	72
147	4.22 Element <KeyCachePolicyRequest>.....	73
148	4.23 Element <KeyCachePolicyResponse>.....	74
149	4.24 Element <KeyCachePolicy>.....	74
150	4.25 Type KeyCacheDetailType.....	77
151	5 Conformance.....	80
152		

153 1 Introduction

154 This document presents the specification for the Symmetric Key Services Markup Language (SKSML), a protocol
155 by which applications may request and receive symmetric key-management services, securely, over networks or
156 other mechanisms as may be selected by implementers. All text is normative unless otherwise indicated.

157 1.1 Terminology

158 The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT",
159 "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF
160 RFC 2119.

161 1.2 Glossary

162 **3DES** – Triple Data Encryption Standard

163 **AES** – Advanced Encryption Standard

164 **Base64** – An encoding scheme for representing data

165 **Ciphertext** – Encrypted data

166 **Cryptographic module** – A software library or hardware module dedicated to performing cryptographic
167 operations

168 **DES** – Data Encryption Standard

169 **DID or Domain ID** – Domain Identifier; the unique **PEN** assigned to an implementation of an **SKMS** (Symmetric
170 Key Management System) within an enterprise

171 **GKID or Global Key ID** – Global Key Identifier; the unique identifier assigned to every symmetric encryption key
172 within an **SKMS**. It is the concatenation of the **DID-SID-KID**

173 **Initialization Vector or IV** – A block of bits required to encrypt/decrypt the first block of data when used with a
174 particular mode of cryptographic operations

175 **KeyCachePolicy** – The collection of rules that defines how a symmetric encryption key may be cached by a
176 client implementation

177 **KID or Key ID** – Key Identifier; the unique integer assigned to every symmetric encryption key generated within a
178 specific **SKS** (Symmetric Key Services) server within an **SKMS** (Symmetric Key Management System)

179 **KeyUsePolicy** – The collection of rules that defines how a symmetric encryption key may be used by an
180 application

181 **PEN** – Private Enterprise Number; the unique integer assigned by IANA to any organization that requests such a
182 number

183 **PII** – Personally Identifiable Information, such as credit card numbers, social security numbers, bank account
184 numbers, drivers license numbers, etc.

185 **Plaintext** – Unencrypted data

186 **SHA** – Secure Hashing Algorithm

187 **SHA-1** – Secure Hashing Algorithm with a resultant size of 160-bits

188 **SHA-256** – Secure Hashing Algorithm with a resultant size of 256-bits

189 **SHA-384** – Secure Hashing Algorithm with a resultant size of 384-bits

190 **SHA-512** – Secure Hashing Algorithm with a resultant size of 512-bits

191 **SID** or **Server ID** – Server Identifier; the unique integer assigned to every **SKS** server within an enterprise's **SKMS**

192 **SKCL** – Symmetric Key Client Library; a software library that supports the **SKSML** protocol

193 **SKMS** – Symmetric Key Management System; a collection of hardware and software providing symmetric
194 encryption key-management services

195 **SKS** – Symmetric Key Services; a server that provides symmetric key management services over a network or
196 other mechanism selected by implementers

197 **SKSML** – Symmetric Key Services Markup Language; an XML-based protocol to request and receive symmetric
198 encryption key-management services

199 **SOAP** – Simple Object Access Protocol

200 **SOAP Body** – The content part of a SOAP message

201 **SOAP Envelope** – The SOAP message consisting of a SOAP Header and a SOAP Body, conforming to the
202 SOAP protocol standard.

203 **SOAP Error** – A SOAP error message response to a SOAP request

204 **SOAP Header** – The header part of a SOAP message containing meta-information about the message, including
205 security-related objects

206 **Symkey** - A symmetric encryption key

207 **unbounded** – A parameter used with the “maxOccurs” attribute to indicate an unlimited number

208 **XMLEncryption** – Encrypted content represented in eXtensible Markup Language that conforms to the World
209 Wide Web Consortium's XML Encryption standard

210 **XMLSignature** – A digital signature represented in eXtensible Markup Language that conforms to the World
211 Wide Web Consortium's XML Signature standard

212 1.3 Normative References

213 **[AES]** Advanced Encryption Standard
214 NIST FIPS 197. <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>

215 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*.
216 IETF RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>

217 **[SOAP]** Simple Object Access Protocol 1.1
218 W3C Recommendation 08 May 2000. <http://www.w3.org/TR/soap/>

219 **[XMLEncryption]** XML Encryption Syntax and Processing
220 W3C Recommendation 10 Dec 2002. <http://www.w3.org/TR/xmlenc-core/>

221 **[XMLSignature]** XML Signature Syntax and Processing
222 W3C Recommendation 12 Feb 2002. <http://www.w3.org/TR/xmlsig-core/>

223 **[WSS]** Web Services Security – SOAP Message Security 1.0
224 OASIS Standard 200401, March 2004
225 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message->
226 [security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)

227 **[RFC 2578]** K. McCloghrie, et al. *Structure of Management Information Version 2 (SMIPv2)*.
228 IETF RFC 2578, April 1999. <http://www.ietf.org/html/rfc2578>

229

2 Background (non-normative)

230

231 A confluence of events is causing many companies to consider encrypting sensitive data across many
232 applications and platforms within their IT infrastructure. Some of these events include:

- 233 • “Breach Disclosure” laws in nearly 40 states of the USA, requiring companies that have suffered
234 breaches on computers containing Personally Identifiable Information (PII) of their employees or
235 customers, to disclose those breaches to the affected individuals
- 236 • Industry-specific regulations such as the credit card industry's Payment Card Industry Data Security
237 Standard, requiring the encryption of credit card numbers accompanied with strong key-management
238 controls
- 239 • National laws such as the US' Health Insurance Portability and Accountability Act (HIPAA) and the
240 European Union Directive, requiring the securing of health-related data and PII, respectively
- 241 • A significant increase in the number of business applications and e-commerce services on the internet
242 requiring credit card numbers for payment, which in turn becomes a target for attackers
- 243 • A significant increase in the number of users connected to the internet with inadequate protection,
244 leading to many attack vectors becoming propagated on these unprotected PC's

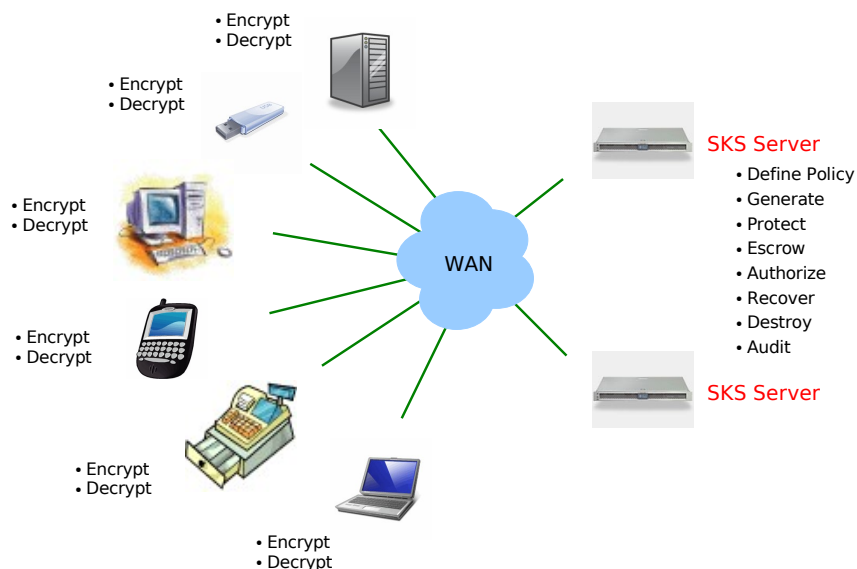
245 In a rush to provide solutions to the market, vendors have created many device-specific, platform-specific,
246 database-specific and application-specific encryption and key-management tools. While these tools may be
247 capable of performing their stated tasks adequately, a typical enterprise would have to deal with many
248 encryption and key-management solutions to adequately protect sensitive data. The following illustration shows
249 how the same key-management tasks need to be repeated across every single device, platform, database and/or
250 application where encryption is performed:



252 Not only does this raise the cost of ownership for implementing companies, but it raises the possibility that with
253 many dissimilar key-management systems, because of the typical complexity of key-management schemes,
254 there is a greater likelihood of human error leading to a vulnerability.

255 To ensure that encryption policies and designs are specified and used uniformly across applications, a common
256 key-management service capable of supporting enterprise platforms, applications and devices is needed. To
257 enable such applications to communicate with this service, a uniform protocol is needed. The Symmetric Key
258 Services Markup Language (SKSML) is that protocol.

259 Once an enterprise has implemented an SKMS, and applications have been modified to take advantage of
260 SKSML, they can expect to see their key-management infrastructure to resemble the following diagram:



262 Architected much like the Domain Name Service (DNS), an SKMS becomes the focal point for all symmetric
 263 encryption key-management services.

264 The Symmetric Key Client Library (SKCL) on client devices is responsible for communicating with the Symmetric
 265 Key Services (SKS) server using SKSML. The SKCL handles security, caching, cryptographic operations and
 266 ensuring that the use of the key is in conformance to policies specified for the key.

267 The SKS server is responsible for storing all policies, keys and information about authorized clients and servers
 268 within the SKMS, and responds to client requests.

269 2.1 Requirements (non-normative)

270 The requirements of the SKSML protocol are that:

- 271 • It must be platform independent;
- 272 • It must support the request of new and previously escrowed symmetric encryption keys;
- 273 • It must support the unique identification of every symmetric encryption key on the internet;
- 274 • It must provide message authenticity, confidentiality and integrity even when used over insecure
 275 networks;
- 276 • It must support the use of encryption/decryption services by a client even when disconnected from the
 277 network;
- 278 • It must provide flexibility in defining key-usage policies;

279 SKSML meets the above requirements in the following manner:

- 280 • SKSML uses SOAP and XML for encapsulating its requests and responses and can thus, be used on
 281 any platform that supports these two underlying protocols;
- 282 • Using a scheme that concatenates unique Domain identifiers (Private Enterprise Numbers issued by the
 283 IANA), unique SKS Server identifiers within a domain and unique Key identifiers within an SKS server,
 284 SKSML creates Global Key Identifiers (GKID) that can uniquely identify symmetric keys across the
 285 internet;
- 286 • SKSML relies on the Web Services Security (WSS) standard 1.0, which in turn supports the use of XML
 287 Signature and XML Encryption within the SOAP Header. Relying only the on the WSS profile that uses

288 RSA cryptographic key-pairs and digital certificates, SKSML uses the digital signatures for authenticity
289 and message-integrity, while using RSA-encryption for confidentiality;

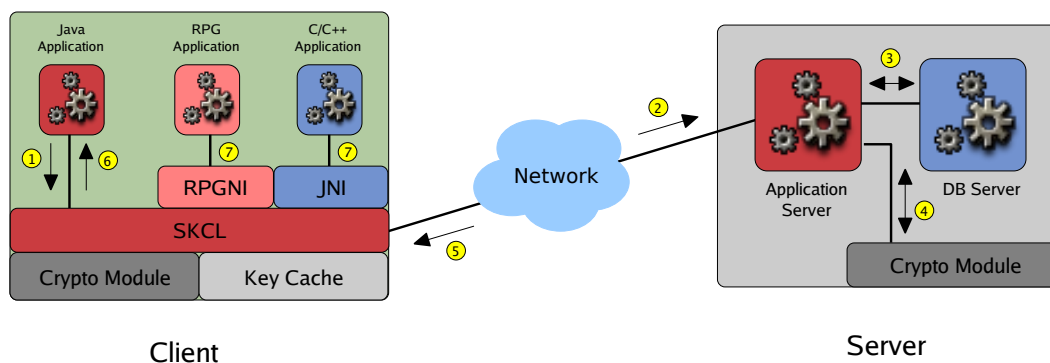
- 290 • Using secure key-caching enabled through centrally-defined policies, SKSML supports the request and
291 receipt of **KeyCachePolicy** elements by clients for the use of symmetric encryption keys even when the
292 client is disconnected from the network and an SKS server;
- 293 • SKSML provides significant flexibility for defining policies on how symmetric encryption keys may be
294 used by client applications. The **KeyUsePolicy** element allows Security Officers to define which
295 applications may use a specific key, days and times of use, location of use, purpose of use, key-sizes,
296 encryption algorithms, etc.

297 SKSML is the first key-management protocol that will do for encryption key-management services what DNS did
298 for name-service protocols: provide a single, standard means of requesting and receiving key-management
299 services from centrally defined servers.

3 Examples of use of SKSML (non-normative)

300

301 The following high-level diagram will be used to describe the use of SKSML.



1. Client Application makes a request for a symmetric key
2. SKCL makes a digitally signed request to the SKS
3. SKS verifies SKCL request, generates, encrypts, digitally signs & escrows key in DB
4. Crypto HSM provides security for RSA Signing & Encryption keys of SKS
5. SKS responds to SKCL with signed and encrypted symmetric key
6. SKCL verifies response, decrypts key and hands it to the Client Application
7. Native (non-Java) applications make requests through Java Native Interface

3.1 Request for a new symmetric key

304

305 When a client application (that has been linked to the SKCL) needs to encrypt sensitive data, it will call an API
306 method within the SKCL for a new symmetric key. After the SKCL has ensured that the application is authorized
307 to make such a request (by verifying that the configured/passed-in credentials can access the cryptographic
308 key-store module on the client containing the PrivateKey used for signing SKSML requests), the SKCL
309 assembles the following SKSML request:

```
310 [a01] <ekmi:SymkeyRequest  
311 [a02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
312 [a03]     <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
313 [a04] </ekmi:SymkeyRequest>
```

314 [a01] is the start of the *SymkeyRequest* element.

315 [a02] identifies the namespace to which this XML conforms, and the location of its XML Schema Definition
316 (XSD).

317 [a03] identifies the *GlobalKeyID* (GKID) being requested by the client application. The GKID is a concatenation
318 of three distinct identifiers in the following order: the unique Domain Identifier, the unique Server Identifier within
319 the domain and the unique Key Identifier generated on a server. Using a "zero" value for the Server ID and the
320 Key ID indicates a request for a new symmetric key.

321 [a04] is the closing tag of the *SymkeyRequest* element.

322 While the *SymkeyRequest* element is very simple, the Web Service Security (WSS) envelope – which provides
323 security for all SKSML messages – expands the size of the message. The same request shown above, is
324 displayed below in its entirety, with its WSS envelope. Please note that some content – such as Base64-
325 encoded binary content - has been reformatted for aesthetics and clarity of the XML elements. The actual
326 elements and data-types have been preserved from actual SKSML messages.

327 For an interpretation of the XML elements shown below, please refer to [WSS].

328 For the sake of brevity, this specification will dispense with showing the SOAP envelope and the WSS elements
329 in all other examples, when discussing SKSML.

```

330 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
331   <SOAP-ENV:Header>
332     <wsse:Security xmlns:wsse="http://docs.oasis-
333 open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" SOAP-
334 ENV:mustUnderstand="1">
335       <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-
336 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
337         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
338 message-security-1.0#Base64Binary"
339         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
340 profile-1.0#X509v3" wsu:Id="XWSSGID-1172790302111-1738806553">
341         MIIDfDCCAmSgAwIBAgIIAe/AvliGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdu3Ryb25nab
342         S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvcjBTdHJvbmdLZXkgREVNTyBVc2Ug1d
343         T25seTEXMBUGA1UEChM0U3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTcxMDMwWhcNMDcwNzIa64dd3k
344         A1UECxMbRm9yIFN0cm9uZ0tleSBERU1PIFVzZSBPbmx5M5RcwFQYDVQKew5TDHJvbmdBdXR0IEl2da
345         S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvcjBTdHJvbmdLZXkgREVNTyBVc2Ug1d
346         T25seTEXMBUGA1UEChM0U3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTY0NjEwWhcNMDcwNzI1s34wdd
347         NjEwWjBpMREwDwYKZCIzPyLGQBARMBOTEVMBMGA1UEAxMMU0tTIFNlcnZlci0xMSQwIgyYDVQsdw2
348         ExtG3Igu3Ryb25nS2V5IERFTU8gVXNlIE9ubHkxZzZAVBgNVBAoTDLN0cm9uZ0F1dGggS5w5jMIIBd2
349         NBGkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAztpqRoU5A8plxx1Rz1QEUnlAAM1D5g9+isIr3wxa
350         hbwtFSMYilnY4iV77xU/nsM0nMZ7RxsLYKdCzQ10DVYqQwqmAvaJ5Z6SVy34gZ51YG+rSWE3NjFsd
351         b0XW8RJYA/Tn6Lmht/qngrcaqmtP0cAAiMRZOWtCTmC2K/LEqDabXSyU6Hh8ySNE3njybvmmWpresf
352         zsYokTdvWQqT6tKo10wJsdJ1+hXM7DrnMLvMNq5reINfsKhDdX17wzhrBUx+hiYA/qo8tMXkL6wsd
353         4PN5dYugtZpSzIdU05tIg58Avhzw07hy5oofB1KFY22CeljQ36u0bMjuyGj6UYHs3rdfdfsds32rda
354         YzCBnzANBkgqhkiG9w0BAQEFAAOBjQAwYkCgYEAyAmxMZhYA8wHJ4UE4b61s51JvWe4Fyggj4Mcf3a
355         hvcNAQELBQADggEBACK05PtvZD4wPgl0e=
356       </wsse:BinarySecurityToken>
357       <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
358         <ds:SignedInfo>
359           <ds:CanonicalizationMethod
360             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
361             <InclusiveNamespaces
362               xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"
363               PrefixList="wsse SOAP-ENV"/>
364             </ds:CanonicalizationMethod>
365           <ds:SignatureMethod
366             Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
367           <ds:Reference URI="#XWSSGID-1172790300636-653454040">
368             <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
369             <ds:DigestValue>LU4m+rp4oebgl9g+t3nRaZYqULe=</ds:DigestValue>
370           </ds:Reference>
371           <ds:Reference URI="#XWSSGID-1172790300637708871805">
372             <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
373             <ds:DigestValue>WCp0mTCbffcEHXhGf5rLEWLRZg=</ds:DigestValue>
374           </ds:Reference>
375         </ds:SignedInfo>
376         <ds:SignatureValue>
377         svStAvBRRrF+g2biPL7uWHkJTQPIl8t4phMb0ZQsZlQcn36tcMSj/a4+4LPNf0B3Y8y02lr10a1
378         fGqCPAWZnuEH34VQEM196rRwV258mgp8uwpXEYJIGpJqg89w8+/Nda0DccLQ2Bizu7QM/HSM2ab
379         ogNjwqmbSyIaz0sn0cU=
380         </ds:SignatureValue>
381         <ds:KeyInfo>
382           <wsse:SecurityTokenReference xmlns:wssu="http://docs.oasis-
383 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
384           wsu:Id="XWSSGID-1172790300633-442423344">

```

```

385         <wsse:Reference URI="#XWSSGID-1172790302111-1738806553"
386             ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
387 wss-x509-token-profile-1.0#X509v3"/>
388         </wsse:SecurityTokenReference>
389     </ds:KeyInfo>
390 </ds:Signature>
391     <wsu:Timestamp xmlns:wsu="http://docs.oasis-
392 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
393 wsu:Id="XWSSGID-1172790300637708871805">
394         <wsu:Created>2007-03-01T23:05:00Z</wsu:Created>
395         <wsu:Expires>2007-03-01T23:05:05Z</wsu:Expires>
396     </wsu:Timestamp>
397 </wsse:Security>
398 </SOAP-ENV:Header>
399 <SOAP-ENV:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
400 wss-wssecurity-utility-1.0.xsd" wsu:Id="XWSSGID-1172790300636-653454040">
401     <ekmi:SymkeyRequest
402         xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
403         <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
404     </ekmi:SymkeyRequest>
405 </SOAP-ENV:Body>
406 </SOAP-ENV:Envelope>

```

407 3.2 Response with a new symmetric key

408 After an SKS server has performed its operations of authenticating the request, identifying the requester,
409 determining policies that apply to the requester, generating the symmetric encryption key in conformance to the
410 defined policy and finally escrowing a symmetric key securely, it assembles the following response and returns it
411 to the client. (The SOAP message, as indicated earlier, is secured using WSS, but only the actual SKSML
412 content is displayed and discussed here).

```

413 [b01] <ekmi:SymkeyResponse
414 [b02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
415 [b03]     xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
416 [b04] <ekmi:Symkey>
417 [b05]     <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
418 [b06]     <ekmi:KeyUsePolicy>
419 [b07]         <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>
420 [b08]         <ekmi:PolicyName>DES-EDE KeyUsePolicy</ekmi:PolicyName>
421 [b09]         <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
422 [b10]         <ekmi:KeyAlgorithm>
423 [b11]             http://www.w3.org/2001/04/xmlenc#tripledes-cbc
424 [b12]         </ekmi:KeyAlgorithm>
425 [b13]         <ekmi:KeySize>192</ekmi:KeySize>
426 [b14]         <ekmi:Status>Active</ekmi:Status>
427 [b15]     <ekmi:Permissions>
428 [b16]         <ekmi:PermittedApplications ekmi:any="false">
429 [b17]             <ekmi:PermittedApplication>
430 [b18]                 <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
431 [b19]                 <ekmi:ApplicationName>
432 [b20]                     Payroll Application
433 [b21]                 </ekmi:ApplicationName>
434 [b22]                 <ekmi:ApplicationVersion>1.0</ekmi:ApplicationVersion>
435 [b23]                 <ekmi:ApplicationDigestAlgorithm>
436 [b24]                     http://www.w3.org/2000/09/xmldsig#sha1
437 [b25]                 </ekmi:ApplicationDigestAlgorithm>
438 [b26]                 <ekmi:ApplicationDigestValue>
439 [b27]                     NIG4bKkt4cziEqFFu0oBTM81efU=
440 [b28]                 </ekmi:ApplicationDigestValue>

```

```

441 [b29]         </ekmi:PermittedApplication>
442 [b30]     </ekmi:PermittedApplications>
443 [b31]     <ekmi:PermittedDates ekmi:any="false">
444 [b32]         <ekmi:PermittedDate>
445 [b33]             <ekmi:StartDate>2008-01-01</ekmi:StartDate>
446 [b34]             <ekmi:EndDate>2008-12-31</ekmi:EndDate>
447 [b35]         </ekmi:PermittedDate>
448 [b36]     </ekmi:PermittedDates>
449 [b37]     <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
450 [b38]     <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
451 [b39]     <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
452 [b40]     <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
453 [b41]     <ekmi:PermittedNumberOfTransactions ekmi:any="true"
454 xsi:nil="true"/>
455 [b42]         <ekmi:PermittedTimes ekmi:any="false">
456 [b43]             <ekmi:PermittedTime>
457 [b44]                 <ekmi:StartTime>07:00:00</ekmi:StartTime>
458 [b45]                 <ekmi:EndTime>19:00:00</ekmi:EndTime>
459 [b46]             </ekmi:PermittedTime>
460 [b47]         </ekmi:PermittedTimes>
461 [b48]         <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
462 [b49]     </ekmi:Permissions>
463 [b50] </ekmi:KeyUsePolicy>
464 [b51]     <ekmi:EncryptionMethod
465 [b52]         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
466 [b53]     <xenc:CipherData>
467 [b54]         <xenc:CipherValue>
468 [b55]             E9zWB/y93hVSzeTLiDcQoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJM
469 [b56]             hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/2fw
470 [b57]             UJow/qimxi8+huUYJMtaGHtXuLLWtx27STRcRpIsY=
471 [b58]         </xenc:CipherValue>
472 [b59]     </xenc:CipherData>
473 [b60] </ekmi:Symkey>
474 [b61] </ekmi:SymkeyResponse>

```

475 [b01] is the start of the *SymkeyResponse* element.

476 [b02] and [b03] identify the namespaces to which this XML conforms, and the location of their XML Schema Definitions (XSD).

478 [b04] is the start tag of the *Symkey* element which contains the symmetric encryption key and related elements.

479 [b05] identifies the *GlobalKeyID* (GKID) assigned by the SKS server for the new symmetric key being returned. In this example, the concatenated values of the Domain ID, Server ID and Key ID indicate that the key belongs to the organization represented by the PEN, 10514; it was generated on an SKS server with a Server ID of 1 and is the 235th unique key generated on that SKS server.

483 [b06] is the start of the *KeyUsePolicy* element. This element contains details of the policy to which SKCL implementations must conform when using the symmetric key.

485 [b07] identifies the unique *KeyUsePolicyID* (KUPID) which identifies this policy within the SKMS.

486 [b08] provides a descriptive name for this key-use policy, which is helpful to human readers when identifying this policy.

488 [b09] identifies the *KeyClass* to which this symmetric key belongs. Key-classes are useful to applications that wish to encrypt plaintext with a key that has specific characteristics. The requesting application is expected to know what *KeyClass* it needs before it asks for a key corresponding to that class.

491 [b10] is the start tag of the *KeyAlgorithm* element.

492 **[b11]** identifies the cryptographic algorithm that this symmetric key must be used with to for cryptographic
493 operations.

494 **[b12]** is the closing tag of the *KeyAlgorithm* element.

495 **[b13]** specifies the size of the symmetric encryption key in bits. While it is possible for application developers to
496 determine this programmatically from the key-object, this element provides this information as a convenience.

497 **[b14]** indicates the **Status** of this *KeyUsePolicy* and whether it is an active policy or not. This is useful in
498 situations where an application may wish to re-use a symmetric key to encrypt related data to the data originally
499 encrypted with the symmetric key. While it is possible for the symmetric key object to be active in the database,
500 it is conceivable that the *KeyUsePolicy* used by the key has changed and the application technically needs to
501 use a new symmetric key to encrypt new data.

502 **[b15]** is the start of the *Permissions* element. This element provides a sophisticated mechanism for controlling
503 how, where, when and by which applications symmetric keys be used. While there are many sub-elements
504 within a *Permissions* element, not all *KeyUsePolicy* objects might use all *Permissions* sub-elements. The
505 example shown in this section only uses three of the possible *Permissions* sub-elements.

506 **[b16]** is the start of the *PermittedApplications* element. This element allows SKMS policies to be defined that
507 allow only specific applications to use symmetric encryption keys associated with this policy. The
508 **ekmi:any="true"** attribute of the *PermittedApplications* element indicates that not just any application on the
509 client machine is permitted to use this symmetric key; only the specified applications within the sub-elements of
510 this element are permitted to use the symmetric key in question..

511 **[b17]** is the start of the first *PermittedApplication* element. This element identifies a specific application within a
512 list of *PermittedApplications* that is allowed to use the symmetric key. There can be any number of
513 *PermittedApplication* elements with *PermittedApplications*.

514 **[b18]** identifies the unique *ApplicationID* (as identified within the SKMS) of the *PermittedApplication*.

515 **[b19]** is the start of the *ApplicationName* element.

516 **[b20]** identifies the *ApplicationName* of the *PermittedApplication* (as identified within the SKMS).

517 **[b21]** is the closing tag of the *ApplicationName* element.

518 **[b22]** identifies the specific *ApplicationVersion* of the *PermittedApplication*. This is helpful when there are
519 multiple versions of a specific application, and the policy-makers need to distinguish between the symmetric keys
520 in use by a specific version of the application.

521 **[b23]** is the start of the *ApplicationDigestAlgorithm* element. This element permits enterprises to ensure that
522 only a cryptographically-verified application is authorized to use the symmetric encryption key. This assumes
523 that the implementation has an infrastructure where the SKCL is capable of determining a cryptographic value to
524 uniquely identify an application within the run-time environment.

525 **[b24]** identifies the W3C-specified URL of the cryptographic algorithm used to calculate the message digest of
526 the application's image.

527 **[b25]** is the closing tag of the *ApplicationDigestAlgorithm* element.

528 **[b26]** is the start of the *ApplicationDigestValue* element. This element permits enterprises to ensure that only a
529 cryptographically-verified application is authorized to use the symmetric encryption key. This assumes that the
530 implementation has an infrastructure where the SKCL is capable of determining a cryptographic value to
531 uniquely identify an application within the run-time environment.

532 **[b27]** identifies the Base64-encoded message digest of the *PermittedApplication's* image, based on the
533 algorithm specified in the *ApplicationDigestAlgorithm* element.

534 **[b28]** is the closing tag of the *ApplicationDigestValue* element.

535 **[b29]** is the closing tag of the *PermittedApplication* element.

536 **[b30]** is the closing tag of the *PermittedApplications* element.

537 [b31] is the start of the *PermittedDates* element. This element permits enterprises to ensure that the symmetric
538 encryption key can be used only during a specified date period. This assumes that the implementation has an
539 infrastructure where the SKCL is capable of accurately determining the current date within the run-time
540 environment.

541 [b32] is the start of the first *PermittedDate* element. There can be any number of *PermittedDate* elements
542 within a *PermittedDates* element.

543 [b33] identifies the *StartDate* of the duration period during which the symmetric encryption key can be used by
544 authorized applications.

545 [b34] identifies the *EndDate* of the duration period during which the symmetric encryption key can be used by
546 authorized applications.

547 [b35] is the closing tag of the *PermittedDate* element.

548 [b36] is the closing tag of the *PermittedDates* element.

549 [b37] is an empty (null) *PermittedDays* element. This element permits enterprises to ensure that the symmetric
550 encryption key can be used only on specific days of the week. This assumes that the implementation has an
551 infrastructure where the SKCL is capable of accurately determining the current day-of-week within the run-time
552 environment. In this specific instance, the null element indicates that this permission does not apply to this
553 symmetric key, and therefore, there are no constraints on its use. However, the key is still subject to all non-null
554 permission clauses.

555 [b38] is an empty (null) *PermittedDuration* element. This element permits enterprises to ensure that the
556 symmetric encryption key can be used only for a specific duration of time once the symmetric key has been used
557 for the first time on the client. This assumes that the implementation has an infrastructure where the SKCL is
558 capable of accurately determining the current time within the run-time environment. In this specific instance, the
559 null element indicates that this permission does not apply to this symmetric key, and therefore, there are no
560 constraints on its use. However, the key is still subject to all non-null permission clauses.

561 [b39] is an empty (null) *PermittedLevels* element. This element permits enterprises to ensure that the
562 symmetric encryption key can be used only by applications that are classified at a given level within the Multi-
563 Level Security (MLS) system as defined in the Bell-LaPadula model. In this specific instance, the null element
564 indicates that this permission does not apply to this symmetric key, and therefore, there are no constraints on its
565 use. However, the key is still subject to all non-null permission clauses.

566 [b40] is an empty (null) *PermittedLocations* element. This element permits enterprises to ensure that the
567 symmetric encryption key can be used only by applications at specified geographic locations on the planet. This
568 assumes that the implementation has an infrastructure where the SKCL is capable of accurately determining the
569 current GPS location of the client within the run-time environment. In this specific instance, the null element
570 indicates that this permission does not apply to this symmetric key, and therefore, there are no constraints on its
571 use. However, the key is still subject to all non-null permission clauses.

572 [b41] is an empty (null) *PermittedNumberOfTransactions* element. This element permits enterprises to ensure
573 that the symmetric encryption key can be used by applications only for a specified number of encryption
574 transactions. In this specific instance, the null element indicates that this permission does not apply to this
575 symmetric key, and therefore, there are no constraints on its use. However, the key is still subject to all non-null
576 permission clauses.

577 [b42] is the start of the *PermittedTimes* element. This element permits enterprises to ensure that the symmetric
578 encryption key can be used only during a specified time period within any date. This assumes that the
579 implementation has an infrastructure where the SKCL is capable of accurately determining the current time
580 within the run-time environment.

581 [b43] is the start of the first *PermittedTime* element. There can be any number of *PermittedTime* elements
582 within a *PermittedTimes* element.

583 [b44] identifies the *StartTime* of the duration period during which the symmetric encryption key can be used by
584 authorized applications.

585 [b45] identifies the *EndTime* of the duration period during which the symmetric encryption key can be used by
586 authorized applications.

587 [b46] is the closing tag of the *PermittedTime* element.

588 [b47] is the closing tag of the *PermittedTimes* element.

589 [b48] is an empty (null) *PermittedUses* element. This element permits enterprises to ensure that the symmetric
590 encryption key can be used by applications for specific uses. In this specific instance, the null element indicates
591 that this permission does not apply to this symmetric key, and therefore, there are no constraints on its use.
592 However, the key is still subject to all non-null permission clauses.

593 [b49] is the closing tag of the *Permissions* element.

594 [b50] is the closing tag of the *KeyUsePolicy* element.

595 [b51]-[b52] identifies the encryption algorithm used in the *EncryptionMethod* element to encrypt the symmetric
596 encryption key itself, to transport to the requesting client. The symmetric key is encrypted using the *PublicKey*
597 or the requesting client. The *Algorithm* attribute uses the W3C-specified URLs for identifying the encryption and
598 padding algorithms.

599 [b53] is the start of the *CipherData* element. This element is from the W3C XML Encryption namespace (as
600 identified by the "xenc" qualifier in the element name).

601 [b54] is the start of the *CipherValue* element. This element contains the Base64-encoded ciphertext of the
602 symmetric encryption key.

603 [b55] – [b57] is the Base64-encoded ciphertext of the symmetric encryption key.

604 [b58] is the closing tag of the *CipherValue* element.

605 [b59] is the closing tag of the *CipherData* element.

606 [b60] is the closing tag of the *Symkey* element.

607 [b61] is the closing tag of the *SymkeyResponse* element.

608 3.3 Request for an existing symmetric key

609 Typically, when a client application encrypts data, it must make accommodations to store the *GlobalKeyID* of
610 the symmetric encryption key it uses to encrypt the plaintext, along with the ciphertext. Without the
611 *GlobalKeyID*, neither the client application nor the SKS server can determine which key was used to encrypt
612 specific ciphertext. When the client application needs to decrypt such ciphertext, it must request the symmetric
613 key with the appropriate *GlobalKeyID* from the SKS server if it does not already have the key cached within its
614 key-cache.

615 The client application (linked to the call SKCL) will an API method within the SKCL for the appropriate symmetric
616 key. After the SKCL has ensured that the application is authorized to make such a request, and assuming that
617 the client application needs the key with the *GlobalKeyID* value of 10514-1-235, the SKCL assembles the
618 following SKSML request. (The SOAP message is secured using WSS, but only the actual SKSML content is
619 displayed and discussed here).

```
620 [c01] <ekmi:SymkeyRequest  
621 [c02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
622 [c03]     <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>  
623 [c04] </ekmi:SymkeyRequest>
```

624 [c01] is the start of the *SymkeyRequest* element.

625 [c02] identifies the namespace to which this XML conforms, and the location of its XML Schema Definition
626 (XSD).

627 [c03] identifies the **GlobalKeyID** (GKID) of the specific symmetric encryption key being requested by the client
628 application.

629 [c04] is the closing tag of the **SymkeyRequest** element.

630 Note that the request for an existing symmetric key is no different from a request for a new symmetric key other
631 than that the **GlobalKeyID** being requested has non-zero values for the Server ID and Key ID parts of the
632 **GlobalKeyID**.

633 3.4 Response with an existing symmetric key

634 After an SKS server has performed its operations of authenticating the request, identifying the requester and
635 determining whether the requester is authorized to receive the symmetric key, the SKS server sends back the
636 symmetric key using a **SymkeyResponse** message secured within a WSS envelope. This **SymkeyResponse** is
637 identical to the message described in Section 3.2 (since the **SymkeyRequest** was for the same symmetric key
638 identified there) and is therefore, not repeated here for brevity.

639 3.5 Request for a new symmetric key of a specific **KeyClass**

640 There are business situations where an application might need a symmetric key that conforms to a
641 **KeyUsePolicy** that has a specific **KeyClass** attribute within the policy. This is useful in situations where there
642 may be many encryption policies within a company that apply to different type of data, geographical zones,
643 applications, level of access to sensitive data, etc., and the requesting application needs a symmetric key which
644 satisfies its business rules.

645 In those situations, a client application (that has been linked to the SKCL) will call an API method within the
646 SKCL for a new symmetric key of a specific **KeyClass**. After the SKCL has ensured that the application is
647 authorized to make such a request the SKCL assembles the following SKSML request:

```
648 [e01] <ekmi:SymkeyRequest  
649 [e02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
650 [e03]     <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
651 [e04]     <ekmi:KeyClasses>  
652 [e05]         <ekmi:KeyClass>HR-Class</ekmi:KeyClass>  
653 [e06]     </ekmi:KeyClasses>  
654 [e07] </ekmi:SymkeyRequest>
```

655 [e01] is the start of the **SymkeyRequest** element.

656 [e02] identifies the namespace to which this XML conforms, and the location of its XML Schema Definition
657 (XSD).

658 [e03] identifies the **GlobalKeyID** (GKID) being requested by the client application. The “zero” value for the
659 Server ID and the Key ID indicates a request for a new symmetric key.

660 [e04] is the start of the **KeyClasses** element.

661 [e05] identifies the specific **KeyClass** being requested by the client application.

662 [e06] is the closing tag of the **KeyClasses** element.

663 [e07] is the closing tag of the **SymkeyRequest** element.

664 3.6 Response with a new symmetric key of a specific **KeyClass**

665 After an SKS server has performed its operations of authenticating the request, identifying the requester and
666 determining whether the requester is authorized to receive a symmetric key of the requested **KeyClass**, the SKS
667 server generates, escrows, encrypts and sends back the symmetric key using a **SymkeyResponse** message
668 secured within a WSS envelope. This **SymkeyResponse** is identical to the message described in Section 3.2

669 (since the symmetric key identified there is of the *KeyClass* requested here) and is therefore, not repeated here
670 for brevity.

671 3.7 Request for multiple new symmetric keys

672 There are business situations where an application needs many symmetric keys to encrypt different parts of a
673 single record. This is useful in applications where many entities might have access to the same “master” record,
674 but only some entities have access to sensitive data within “detail” records. In these situations, the use of
675 multiple symmetric keys addresses this business requirement, while allowing the entire record to freely move to
676 any system without fear of loss of confidentiality.

677 For example, within an Electronic Health Record (EHR) application, the application might store a Patient's
678 medical data as a single logical EHR within a database (even though they may be physically represented by
679 many hundreds of detail records). This has the benefit of presenting a single view of a Patient's EHR to all
680 actors within the use-case. However, the information necessary to a Physician treating the patient is quite
681 different from the information necessary to an insurance company processing a claim, a government agency
682 tracks diseases, a pharmacy filling out a prescription or a nurse administering treatment to the patient.

683 While there is a clear business advantage to maintaining all patient data as a single logical EHR, security and
684 privacy requirements dictate that appropriate sensitive data must be made visible only to authorized entities.

685 In these situations, a client application (that has been linked to the SKCL) will call an API method within the
686 SKCL for multiple new symmetric keys. In order to request multiple symmetric keys, the SKSML request must
687 contain a *KeyClass* element within the *KeyClasses* element for every key the client application needs. Thus, if
688 the EHR application were to request nine symmetric keys to encrypt different parts of the EHR, the client
689 application would make the following SKSML *SymkeyRequest*:

```
690 [g01] <ekmi:SymkeyRequest  
691 [g02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
692 [g03]     <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
693 [g04]     <ekmi:KeyClasses>  
694 [g05]         <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>  
695 [g06]         <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>  
696 [g07]         <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>  
697 [g08]         <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>  
698 [g09]         <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>  
699 [g10]        <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>  
700 [g11]        <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>  
701 [g12]        <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>  
702 [g13]        <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>  
703 [g14]     </ekmi:KeyClasses>  
704 [g15] </ekmi:SymkeyRequest>
```

705 [g01] is the start of the *SymkeyRequest* element.

706 [g02] identifies the namespace to which this XML conforms, and the location of its XML Schema Definition
707 (XSD).

708 [g03] identifies the *GlobalKeyID* (GKID) being requested by the client application. The “zero” value for the
709 Server ID and the Key ID indicates a request for a new symmetric key.

710 [g04] is the start of the *KeyClasses* element.

711 [g05] identifies a *KeyClass* for a symmetric encryption key being requested by the client application, ostensibly
712 for encrypting data that can later be decrypted by an authorized application at the Center for Disease Control
713 (CDC) and any other application that has been granted access to keys of the EHR-CDC *KeyClass*.

714 [g06] identifies a *KeyClass* for a symmetric encryption key being requested by the client application, for
715 encrypting data that can later be decrypted only by an authorized application at Clinical Research Organizations
716 (CRO) and any other application that has been granted access to keys of the EHR-CRO *KeyClass*.

717 [g07] identifies a default **KeyClass** (EHR-DEF) for a symmetric encryption key for encrypting data that can later
718 be decrypted by any authorized application within the EHR system.

719 [g08] identifies a **KeyClass** for a symmetric encryption key for encrypting data that was collected by an
720 Emergency Medical Technician (EMT), and which can later be decrypted only by authorized applications at the
721 hospital that have access to keys of the EHR-EMT **KeyClass**.

722 [g09] identifies a **KeyClass** for a symmetric encryption key for encrypting data collected by a hospital where the
723 patient might have used for treatment. Data encrypted by keys of this EHR-HOS **KeyClass** can later be
724 decrypted only by authorized application that has access to keys of this **KeyClass**.

725 [g10] identifies a **KeyClass** (EHR-INS) for a symmetric encryption key that might be used for encrypting data
726 which will be submitted to an insurance company for claims processing.

727 [g11] identifies a **KeyClass** for a symmetric encryption key for encrypting data that can later be decrypted by
728 any authorized application used by nurses and/or other authorized entities in providing treatment to a patient at
729 the hospital (EHR-NUR).

730 [g12] identifies a **KeyClass** for a symmetric encryption key for encrypting patient related data (EHR-PAT) that
731 may not be medical in nature, but nevertheless sensitive.

732 [g13] identifies a **KeyClass** for a symmetric encryption key for encrypting data that can later be decrypted by
733 authorized physicians and other entities that have access to keys of the EHR-PHY **KeyClass**.

734 [g14] is the closing tag of the **KeyClasses** element.

735 [g15] is the closing tag of the **SymkeyRequest** element.

736 3.8 Response with multiple new symmetric keys

737 After an SKS server has performed its operations of authenticating the request, identifying the requester,
738 determining policies that apply to the requester, generating the symmetric encryption keys in conformance to the
739 defined policies and **KeyClasses** and finally escrowing the symmetric keys securely, it assembles the following
740 response and returns it to the client.

741 To reduce the verbosity of the response that includes nine symmetric encryption keys, the SKSML shows only
742 the details of two symmetric keys and the encapsulating element tags for the remaining seven keys. Regardless
743 of what the KeyUsePolicy and/or Permissions elements state in those seven keys, the schema for each response
744 conforms to the specification described in this document. Additionally, the SOAP message, as indicated earlier,
745 is secured using WSS, but only the actual SKSML content is displayed and discussed here.

```

746 [h01] <ekmi:SymkeyResponse
747 [h02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
748 [h03]     xmlns:xenc='http://www.w3.org/2001/04/xmenc#'>
749 [h04]     <ekmi:Symkey>
750 [h05]         <ekmi:GlobalKeyID>10514-4-3792</ekmi:GlobalKeyID>
751 [h06]         <ekmi:KeyUsePolicy>
752 [h07]             <ekmi:KeyUsePolicyID>10514-9</ekmi:KeyUsePolicyID>
753 [h08]             <ekmi:PolicyName>DES-EDE KeyUsePolicy for EHR-CDC</ekmi:PolicyName>
754 [h09]             <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
755 [h10]             <ekmi:KeyAlgorithm>
756 [h11]                 http://www.w3.org/2001/04/xmenc#tripleDES-cbc
757 [h12]             </ekmi:KeyAlgorithm>
758 [h13]             <ekmi:KeySize>192</ekmi:KeySize>
759 [h14]             <ekmi:Status>Active</ekmi:Status>
760 [h15]             <ekmi:Permissions>
761 [h16]                 <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
762 [h17]                 <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
763 [h18]                 <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
764 [h19]                 <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
765 [h20]                 <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>

```

```

766 [h21]         <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
767 [h22]         <ekmi:PermittedNumberOfTransactions
768 [h23]             ekmi:any="true" xsi:nil="true"/>
769 [h24]         <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
770 [h25]         <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
771 [h26]     </ekmi:Permissions>
772 [h27] </ekmi:KeyUsePolicy>
773 [h28] <ekmi:EncryptionMethod
774 [h29]     Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
775 [h30] <xenc:CipherData>
776 [h31]     <xenc:CipherValue>
777 [h32]         E9zWB/y93hVSzeTLiDcQoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJMa1w
778 [h33]         hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfg1pU8tLWtx27STRcR/2fw2ava
779 [h34]         ULWtx27STRcRJMtaGHtXuLlWtx27STRcRpIsY=
780 [h35]     </xenc:CipherValue>
781 [h36] </xenc:CipherData>
782 [h37] </ekmi:Symkey>
783 [h38] <ekmi:Symkey>
784 [h39]     <ekmi:GlobalKeyID>10514-4-3793</ekmi:GlobalKeyID>
785 [h40]     <ekmi:KeyUsePolicy>
786 [h41]         <ekmi:KeyUsePolicyID>10514-12</ekmi:KeyUsePolicyID>
787 [h42]         <ekmi:PolicyName>DES-EDE KeyUsePolicy for EHR-CR0</ekmi:PolicyName>
788 [h43]         <ekmi:KeyClass>EHR-CR0</ekmi:KeyClass>
789 [h44]         <ekmi:KeyAlgorithm>
790 [h45]             http://www.w3.org/2001/04/xmlenc#tripledes-cbc
791 [h46]         </ekmi:KeyAlgorithm>
792 [h47]         <ekmi:KeySize>192</ekmi:KeySize>
793 [h48]         <ekmi:Status>Active</ekmi:Status>
794 [h49]         <ekmi:Permissions>
795 [h50]             <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
796 [h51]             <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
797 [h52]                 <ekmi:PermittedDate>
798 [h53]                     <ekmi:StartDate>2008-01-01</ekmi:StartDate>
799 [h54]                     <ekmi:EndDate>2009-12-31</ekmi:EndDate>
800 [h55]                 </ekmi:PermittedDate>
801 [h56]             </ekmi:PermittedDates>
802 [h57]             <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
803 [h58]             <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
804 [h59]             <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
805 [h60]             <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
806 [h61]             <ekmi:PermittedNumberOfTransactions
807 [h62]                 ekmi:any="true" xsi:nil="true"/>
808 [h63]             <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
809 [h64]             <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
810 [h65]         </ekmi:Permissions>
811 [h66]     </ekmi:KeyUsePolicy>
812 [h67]     <ekmi:EncryptionMethod
813 [h68]         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
814 [h69]     <xenc:CipherData>
815 [h70]         <xenc:CipherValue>
816 [h71]             qUiQXG0ca8EU871zBoXBjDoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJM1
817 [h72]             hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/2fw1
818 [h73]             UJow/qimxi8+ huUYJMtaGHtXuLlWtx27STRcRpIsY=
819 [h74]         </xenc:CipherValue>
820 [h75]     </xenc:CipherData>
821 [h76] </ekmi:Symkey>
822 [h77] <ekmi:Symkey>
823 [h78]     <ekmi:GlobalKeyID>10514-4-3795</ekmi:GlobalKeyID>
824     ...
825 [h79]     <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>

```

```

826      ...
827 [h80]    </ekmi:Symkey>
828 [h81]    <ekmi:Symkey>
829 [h82]      <ekmi:GlobalKeyID>10514-4-3797</ekmi:GlobalKeyID>
830
831 [h83]      ... <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
832      ...
833 [h84]    </ekmi:Symkey>
834 [h85]    <ekmi:Symkey>
835 [h86]      <ekmi:GlobalKeyID>10514-4-3798</ekmi:GlobalKeyID>
836
837 [h87]      ... <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
838      ...
839 [h88]    </ekmi:Symkey>
840 [h89]    <ekmi:Symkey>
841 [h90]      <ekmi:GlobalKeyID>10514-4-3799</ekmi:GlobalKeyID>
842
843 [h91]      ... <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
844      ...
845 [h92]    </ekmi:Symkey>
846 [h93]    <ekmi:Symkey>
847 [h94]      <ekmi:GlobalKeyID>10514-4-3801</ekmi:GlobalKeyID>
848
849 [h95]      ... <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
850      ...
851 [h96]    </ekmi:Symkey>
852 [h97]    <ekmi:Symkey>
853 [h98]      <ekmi:GlobalKeyID>10514-4-3803</ekmi:GlobalKeyID>
854
855 [h99]      ... <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
856      ...
857 [h100]   </ekmi:Symkey>
858 [h101]   <ekmi:Symkey>
859 [h102]     <ekmi:GlobalKeyID>10514-4-3805</ekmi:GlobalKeyID>
860
861 [h103]     ... <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>
862     ...
863 [h104]   </ekmi:Symkey>
864 [h105] </ekmi:SymkeyResponse>

```

865 [h01] is the start of the *SymkeyResponse* element.

866 [h02] and [h03] identify the namespaces to which this XML conforms, and the location of their XML Schema Definitions (XSD).

868 [h04] is the start tag of the first *Symkey* element which contains the symmetric encryption key and related elements.

870 [h05] identifies the *GlobalKeyID* (GKID) assigned by the SKS server of this first symmetric key. In this example, the concatenated values of the Domain ID, Server ID and Key ID indicate that the key belongs to the organization represented by the PEN, 10514; it was generated on an SKS server with a Server ID of 4 and is the 3792nd unique key generated on that SKS server.

874 [h06] is the start of the *KeyUsePolicy* element that applies just to this symmetric key. This element contains details of the policy to which SKCL implementations must conform when using the symmetric key.

876 [h07] identifies the unique *KeyUsePolicyID* (*KUPID*) which identifies this policy within the SKMS.

877 [h08] provides a descriptive name for this key-use policy, which is helpful to human readers when identifying this policy.

879 [h09] identifies the **KeyClass** to which this symmetric key belongs. In the case of this example, the first
880 symmetric key in the response conforms to the **EHR-CDC** class which, presumably, might be a key that covers
881 data encrypted for/by the Center for Disease Control (CDC) within an Electronic Health Record (EHR) system.
882 Key-classes are useful to applications that wish to encrypt plaintext with a key that has specific characteristics.
883 The requesting application is expected to know what **KeyClass** it needs before it asks for a key corresponding to
884 that class.

885 [h10] is the start tag of the **KeyAlgorithm** element.

886 [h11] identifies the cryptographic algorithm that this symmetric key must be used with. For this symmetric key
887 example, the algorithm is the Triple Data Encryption Standard (3DES) with Cipher Block Chaining (CBC)
888 padding. The URL is a standard notation for this algorithm and padding as defined within [XMLEncryption]..

889 [h12] is the closing tag of the **KeyAlgorithm** element.

890 [h13] specifies the size of the symmetric encryption key in bits. For this Triple-DES key, it is 192-bits.

891 [h14] indicates the **Status** of this **KeyUsePolicy** and whether it is an active policy or not. This is useful in
892 situations where an application may wish to re-use a symmetric key to encrypt related data to the data originally
893 encrypted with the symmetric key. While it is possible for the symmetric key object to be active in the database,
894 it is conceivable that the **KeyUsePolicy** used by the key has changed and the application technically needs to
895 use a new symmetric key to encrypt new data.

896 [h15] is the start of the **Permissions** element. This element provides a sophisticated mechanism for controlling
897 how, where, when and by which applications symmetric keys be used. While there are many sub-elements
898 within a **Permissions** element, not all **KeyUsePolicy** objects might use all **Permissions** sub-
899 elements<ekmi:RequestedKeyClass>Payroll-Tax-Class</ekmi:RequestedKeyClass>. The example shown for this
900 symmetric key indicates that there are no other specific restrictions on the use of this symmetric key by
901 authorized client applications; i.e. any authorized client application may use it at any time, on any date, in any
902 location for any purpose.

903 [h16] is the start and end of the null **PermittedApplications** element. This implies that there are no restrictions
904 on which application can use this symmetric key.

905 [h17] is the start and end of the null **PermittedDates** element. This implies that there are no date restrictions on
906 when this symmetric key can be used.

907 [h18] is the start and end of the null **PermittedDays** element. This implies that there are no day-of-week
908 restrictions on when this symmetric key can be used.

909 [h19] is the start and end of the null **PermittedDuration** element. This implies that there are no restrictions to
910 how long this symmetric key may be used.

911 [h20] is the start and end of the null **PermittedLevels** element. This implies that there are no restrictions on the
912 MLS security level in which this symmetric key can be used.

913 [h21] is the start and end of the null **PermittedLocations** element. This implies that there are no geophysical
914 restrictions where this symmetric key can be used.

915 [h22] - [h23] is the start and end of the null **PermittedNumberOfTransactions** element. This implies that there
916 are no restrictions on how many encryption transactions that can be performed by this symmetric key.

917 [h24] is the start and end of the null **PermittedTimes** element. This implies that there are no time-of-day
918 restrictions when this symmetric key can be used.

919 [h25] is the start and end of the null **PermittedUses** element. This implies that there are no restrictions how this
920 symmetric key can be used by applications.

921 [h26] is the closing tag of the **Permissions** element.

922 [h27] is the closing tag of the **KeyUsePolicy** element.

923 [h28] and [h29] identify the encryption algorithm used in the **EncryptionMethod** element to encrypt the
924 symmetric encryption key itself, to transport to the requesting client. The symmetric key is encrypted using the

925 PublicKey or the requesting client. The **Algorithm** attribute uses the W3C-specified URLs for identifying the
926 encryption and padding algorithms.

927 [h30] is the start of the **CipherData** element. This element is from the W3C XML Encryption namespace (as
928 identified by the "xenc" qualifier in the element name).

929 [h31] is the start of the **CipherValue** element. This element contains the Base64-encoded ciphertext of the
930 symmetric encryption key.

931 [h32] – [h34] is the Base64-encoded ciphertext of the symmetric encryption key.

932 [h35] is the closing tag of the **CipherValue** element.

933 [h36] is the closing tag of the **CipherData** element.

934 [h37] is the closing tag of the first **Symkey** element within this **SymkeyResponse**.

935 [h38] - [h76] represents the **second Symkey** element in this **SymkeyResponse**. The differences in this
936 symmetric key element from the first, can be summarized as follows:

- 937 • [h39] identifies a different **GlobalKeyID** (10514-4-3793) for this symmetric key;
- 938 • [h41] identifies a different **KeyUsePolicy** (10514-12) for this symmetric key;
- 939 • [h43] identifies a different **KeyClass** (EHR-CRO) for this symmetric key;
- 940 • [h49] - [h65] defines a different **Permissions** element for this symmetric key;
- 941 • [h71] - [h73] contains a different **CipherValue** for this symmetric key;

942 [h78] – [h80] is the container for the **third Symkey** element in this response. For the sake of brevity, all the
943 usual **Symkey** elements have been dispensed with, but the unique **GlobalKeyID** and **KeyClass** are shown to
944 indicate the SKS server's response to the request. In this example, they are 10514-4-3795 and EHR-DEF
945 respectively.

946 [h81] – [h104] contain the remaining **Symkey** elements of this **SymkeyResponse**. Once again, for brevity, all
947 the details of the **Symkey** elements are dispensed with, except for the unique GKIDs and KeyClasses.

948 [h105] is the closing tag of the **SymkeyResponse** element.

949 Note that it is possible for a request to contain the same **KeyClass** multiple times; there is no requirement that
950 they need to be unique within a request if an application has a legitimate business need for multiple symmetric
951 keys of the same **KeyClass**. The SKS server will respond with unique symmetric keys, all belonging to the
952 **KeyClass** requested by the client application.

953 3.9 Response with an SKS error

954 While one hopes that all authorized requesters will get favorable responses from the SKS server, there are
955 situations in which the client application can receive an error to a request for a symmetric key. The following
956 XML shows one example of such an error response. Depending on the type of error, the actual message
957 content might be different.

```
958 [i01] <ekmi:SymkeyResponse  
959 [i02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'  
960 [i03]     xmlns:xenc='http://www.w3.org/2001/04/xmllenc#'>  
961 [i04]     <ekmi:SymkeyError>  
962 [i05]         <ekmi:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>  
963 [i06]         <ekmi:RequestedKeyClass>Payroll</ekmi:RequestedKeyClass>  
964 [i07]         <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>  
965 [i08]         <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>  
966 [i09]     </ekmi:SymkeyError>  
967 [i10] </ekmi:SymkeyResponse>
```


968 [i01] is the start of the *SymkeyResponse* element.

969 [i02] and [i03] identify the namespaces to which this XML conforms, and the location of their XML Schema
970 Definitions (XSD).

971 [i04] is the start of the *SymkeyError* element, which tells the Symmetric Key Client Library (SKCL) that the
972 request for a symmetric key resulted in an error.

973 [i05] indicates the *RequestedGlobalKeyID* the client requested. Returning the GKID in the error response
974 allows the SKCL to associate the error message with the requesting application on the client machine.

975 [i06] indicates the *RequestedKeyClass* the client requested. Returning the key-class in the error response allows
976 the SKCL to associate the error message with the requesting application on the client machine.

977 [i07] is an *ErrorCode* returned by the SKS server. The code may be one of the standard error codes defined in
978 this specification, or may be a vendor-specific error code.

979 [i08] is the text of the *ErrorMessage*, localized to the region and language of the requesting client application.

980 [i09] is the closing tag of the *SymkeyError* tag.

981 [i10] is the closing tag of the *SymkeyResponse* tag.

982 3.10 Response with symmetric keys and errors

983 When a client application requests multiple symmetric keys, the SKS server may respond in one of three ways.
984 The SKS server may:

- 985 i. Return all symmetric keys as requested;
- 986 ii. Return no symmetric keys – i.e. it returns all errors;
- 987 iii. Return some symmetric keys and some errors.

988 The following SKSML shows the third case, where the server returns some symmetric keys and errors in
989 response to a request for multiple keys (such as the one shown in *Section 3.7 Request for multiple new*
990 *symmetric keys*).

991 In a response that contains a mix of symmetric keys and errors, all symmetric keys precede all errors – i.e. the
992 *SymkeyResponse* element will not consist of *Symkeys* interspersed with *SymkeyErrors* in between; all
993 *Symkeys* (if any) will start from the top of the response till the first *SymkeyError* element.

```

994 [j01] <ekmi:SymkeyResponse
995 [j02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
996 [j03]     xmlns:xenc='http://www.w3.org/2001/04/xmenc#'>
997 [j04]   <ekmi:Symkey>
998 [j05]     <ekmi:GlobalKeyID>10514-4-3792</ekmi:GlobalKeyID>
999 [j06]     <ekmi:KeyUsePolicy>
1000 [j07]       <ekmi:KeyUsePolicyID>10514-9</ekmi:KeyUsePolicyID>
1001 [j08]       <ekmi:PolicyName>DES-EDE Policy for EHR-CDC</ekmi:PolicyName>
1002 [j09]       <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
1003 [j10]       <ekmi:KeyAlgorithm>
1004 [j11]         http://www.w3.org/2001/04/xmenc#tripleDES-cbc
1005 [j12]       </ekmi:KeyAlgorithm>
1006 [j13]     <ekmi:KeySize>192</ekmi:KeySize>
1007 [j14]     <ekmi:Status>Active</ekmi:Status>
1008 [j15]     <ekmi:Permissions>
1009 [j16]       <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
1010 [j17]       <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
1011 [j18]       <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
1012 [j19]       <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
1013 [j20]       <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>

```

```

1014 [j21]         <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
1015 [j22]         <ekmi:PermittedNumberOfTransactions
1016 [j23]             ekmi:any="true" xsi:nil="true"/>
1017 [j24]         <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
1018 [j25]         <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
1019 [j26]         </ekmi:Permissions>
1020 [j27]     </ekmi:KeyUsePolicy>
1021 [j28]     <ekmi:EncryptionMethod
1022 [j29]         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
1023 [j30]     <xenc:CipherData>
1024 [j31]         <xenc:CipherValue>
1025 [j32]             E9zWB/y93hVSzeTLiDcQoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJ
1026 [j33]             hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfglpU8tLWtx27STRcR/2fw
1027 [j34]             ULWtx27STRcRJMtaGHtXuLlWtx27STRcRpIsY=
1028 [j35]         </xenc:CipherValue>
1029 [j36]     </xenc:CipherData>
1030 [j37] </ekmi:Symkey>
1031 [j38] <ekmi:Symkey>
1032 [j39]     <ekmi:GlobalKeyID>10514-4-3793</ekmi:GlobalKeyID>
1033 [j40]     <ekmi:KeyUsePolicy>
1034 [j41]         <ekmi:KeyUsePolicyID>10514-12</ekmi:KeyUsePolicyID>
1035 [j42]         <ekmi:PolicyName>DES-EDE Policy for EHR-CRO</ekmi:PolicyName>
1036 [j43]         <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
1037 [j44]         <ekmi:KeyAlgorithm>
1038 [j45]             http://www.w3.org/2001/04/xmlenc#tripledes-cbc
1039 [j46]         </ekmi:KeyAlgorithm>
1040 [j47]         <ekmi:KeySize>192</ekmi:KeySize>
1041 [j48]         <ekmi:Status>Active</ekmi:Status>
1042 [j49]         <ekmi:Permissions>
1043 [j50]             <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
1044 [j51]             <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
1045 [j52]                 <ekmi:PermittedDate>
1046 [j53]                     <ekmi:StartDate>2008-01-01</ekmi:StartDate>
1047 [j54]                     <ekmi:EndDate>2009-12-31</ekmi:EndDate>
1048 [j55]                 </ekmi:PermittedDate>
1049 [j56]             </ekmi:PermittedDates>
1050 [j57]             <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
1051 [j58]             <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
1052 [j59]             <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
1053 [j60]             <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
1054 [j61]             <ekmi:PermittedNumberOfTransactions
1055 [j62]                 ekmi:any="true" xsi:nil="true"/>
1056 [j63]             <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
1057 [j64]             <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
1058 [j65]         </ekmi:Permissions>
1059 [j66]     </ekmi:KeyUsePolicy>
1060 [j67]     <ekmi:EncryptionMethod
1061 [j68]         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
1062 [j69]     <xenc:CipherData>
1063 [j70]         <xenc:CipherValue>
1064 [j71]             qUiQXG0ca8EU871zBoXBjDoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCF
1065 [j72]             hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfglpU8tGFbpWZcd/ATpJD/
1066 [j73]             UJow/qimxi8+ huUYJMtaGHtXuLlWtx27STRcRpIsY=
1067 [j74]         </xenc:CipherValue>
1068 [j75]     </xenc:CipherData>
1069 [j76] </ekmi:Symkey>
1070 [j77] <ekmi:Symkey>
1071 [j78]     <ekmi:GlobalKeyID>10514-4-3795</ekmi:GlobalKeyID>
1072 [j79]     ...
1073 [j79]     <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>

```

```

1074     ...
1075 [j80]     </ekmi:Symkey>
1076 [j81]     <ekmi:Symkey>
1077 [j82]     <ekmi:GlobalKeyID>10514-4-3797</ekmi:GlobalKeyID>
1078     ...
1079 [j83]     <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
1080     ...
1081 [j84]     </ekmi:Symkey>
1082 [j85]     <ekmi:Symkey>
1083 [j86]     <ekmi:GlobalKeyID>10514-4-3798</ekmi:GlobalKeyID>
1084     ...
1085 [j87]     <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
1086     ...
1087 [j88]     </ekmi:Symkey>
1088 [j89]     <ekmi:Symkey>
1089 [j90]     <ekmi:GlobalKeyID>10514-4-3799</ekmi:GlobalKeyID>
1090     ...
1091 [j91]     <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
1092     ...
1093 [j92]     </ekmi:Symkey>
1094 [j93]     <ekmi:Symkey>
1095 [j94]     <ekmi:GlobalKeyID>10514-4-3801</ekmi:GlobalKeyID>
1096     ...
1097 [j95]     <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
1098     ...
1099 [j96]     </ekmi:Symkey>
1100 [j97]     <ekmi:SymkeyError>
1101 [j98]         <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
1102 [j99]         <ekmi:RequestedKeyClass>EHR-PAT</ekmi:RequestedKeyClass>
1103 [j100]        <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
1104 [j101]        <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
1105 [j102]     </ekmi:SymkeyError>
1106 [j103]     <ekmi:SymkeyError>
1107 [j104]         <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
1108 [j105]         <ekmi:RequestedKeyClass>EHR-PHY</ekmi:RequestedKeyClass>
1109 [j106]         <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
1110 [j107]         <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
1111 [j108]     </ekmi:SymkeyError>
1112 [j109] </ekmi:SymkeyResponse>

```

1113 [j01] – [j96] is no different from the response shown in *Section 3.8 Response with multiple new symmetric*
1114 *keys*.

1115 [j97] - [j102] identifies the first *SymkeyError* returned by the SKS server. It is not unlike the error described in
1116 *Section 3.9 Response with an SKS error*. However, there is one difference in this element on line [j79]. This
1117 element includes the *RequestedKeyClass* of the request.

1118 [j103] - [j108] identifies the second *SymkeyError* returned by the SKS server. It is similar to the error described
1119 in lines [j77] through [j82] including the *RequestedKeyClass* of the request.

1120 [j109] is the closing tag of the *SymkeyResponse* tag.

1121 3.11 Request for a symmetric key-caching policy

1122 When a client application (that has been linked to the SKCL) needs to encrypt sensitive data, it will call an API
1123 method within the SKCL for a new symmetric key. After the SKCL has ensured that the application is authorized
1124 to make such a request (by verifying that the configured/passed-in credentials can access the cryptographic
1125 key-store module on the client containing the PrivateKey used for signing SKSML requests), the SKCL
1126 assembles the following SKSML request:

1127 [k01] <ekmi:KeyCachePolicyRequest
1128 [k02] xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01"/>

1129 [k01] is the start of the *KeyCachePolicyRequest* element.

1130 [k02] identifies the namespace to which this XML conforms, and the location of its XML Schema Definition (XSD).

1132 The *KeyCachePolicyRequest* is an "empty" request. It does not need to specify any requesting parameter or element, since the SKS server only needs to know who requested the policy. The server derives this information from the SOAP Header and consequently has everything it needs to know – the digital signature to establish the identity of the requester, as well as to ensure message integrity in the request.

1136 While the *KeyCachePolicyRequest* element is very simple, the Web Service Security (WSS) envelope – which provides security for all SKSML messages – expands the size of the message. The same request shown above, is displayed below in its entirety, with its WSS envelope. Please note that some content – such as Base64-encoded binary content - has been reformatted for aesthetics and clarity of the XML elements. The actual elements and data-types have been preserved from actual SKSML messages.

1141 For an interpretation of the XML elements shown below, please refer to [WSS].

1142 For the sake of brevity, this specification will dispense with showing the SOAP envelope and the WSS elements in all other examples, when discussing SKSML.

```
1144 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
1145   <SOAP-ENV:Header>
1146     <wsse:Security xmlns:wsse="http://docs.oasis-
1147 open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" SOAP-
1148 ENV:mustUnderstand="1">
1149       <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-
1150 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
1151       EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
1152 message-security-1.0#Base64Binary"
1153       ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
1154 profile-1.0#X509v3" wsu:Id="XWSSGID-1172790302111-1738806553">
1155         MIIDfDCCAmSgAwIBAgIIAe/AvliGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdU3Ryb25nab
1156 S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvcjBTdHJvbmdLZXkgREVNTyBVc2Ug1d
1157 T25seTEXMBUGA1UEChMOU3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTcxMDMwHhcNMDcwNzIa64dd3k
1158 A1UECxMmRm9yIFN0cm9uZ0tleSBERU1PIFVzZSBPbmx5MRcwFQYDVQKEw5TdHJvbmdBdXRoIEI2da
1159 S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvcjBTdHJvbmdLZXkgREVNTyBVc2Ugia
1160 T25seTEXMBUGA1UEChMOU3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTY0NjEwWWhcNMDcwNzI1s34wdd
1161 NjEwWjBpMREwDwYKZCIzImiZPYLQGBARMB0TEVMBMGA1UEAxMMU0tTIFNlcnZlci0xMSQwIgwYDVQsdw2
1162 ExtG63IgwYDVQsdw25nS2V5IERFTU8gVXNlIE9ubHxkFzAVBgNVBAoTDlN0cm9uZ0F1dGggSW5jMIIBD2
1163 NBGkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAztpqRoU5A8plxx1Rz1QEUnlAAM1D5g9+isIr3wxa
1164 hbwjtFSMYilnY4iv77xU/nsM0nMz7RxsLYKdCzQ10DVYqQwqmAvaJ5Z6SVy34gZ51YG+rSWE3NjFsd
1165 b0XW8RJYA/Tn6Lmht/qngrcaqmtP0cAAiMRZOWtCTmC2K/LEqDabXSyU6Hh8ySNE3njybvMwPresf
1166 zsYokTdvNwQqT6tKolowJsdJl+hxM7DrnMLvMnq5reINfsKhDdX17wzhrBUx+hiYA/qo8tMXkL6wsd
1167 4PN5dYugtZpSzIdU05tIg58Avhzw7hy5ooFbLKFY22CeljQ36u0BmJuyGj6UYHs3rdfdfsds32rda
1168 YzCBnzANBqkqhkiG9w0BAQEFAAOBjQAwYkCgYEAyAmxMzHYA8wHJ4UE4b61s51JVWe4Fygj4Mcf3a
1169 hvcNAQELBQADggEBACK05PtvZD4WPgl0e=
1170   </wsse:BinarySecurityToken>
1171   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
1172     <ds:SignedInfo>
1173       <ds:CanonicalizationMethod
1174         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1175       <InclusiveNamespaces
1176         xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"
1177         PrefixList="wsse SOAP-ENV"/>
1178       </ds:CanonicalizationMethod>
1179     <ds:SignatureMethod
1180       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
1181     <ds:Reference URI="#XWSSGID-1172790300636-653454040">
```

```

1182         <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1183         <ds:DigestValue>LU4m+rp4oebgl9g+t3nRaZYqULe=</ds:DigestValue>
1184     </ds:Reference>
1185     <ds:Reference URI="#XWSSGID-1172790300637708871805">
1186     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1187     <ds:DigestValue>WCp0mTCbffcEHXhGf5rlEYWLrZg=</ds:DigestValue>
1188     </ds:Reference>
1189 </ds:SignedInfo>
1190 <ds:SignatureValue>
1191 svStAvBRRrF+g2biPL7uWHkJTQPIl8t4phMb0ZQsZlQcn36tcMSj/a4+4LPNf0B3Y8y02lr10a1
1192 fGqCPAWZnuEH34VQEM196rRwV258mmp8uwpXEYJIgPJqg89w8+/Nda0DccLQ2Bizu7QM/HSM2ab
1193 ogNJwqmbSyIaz0sn0cU=
1194 </ds:SignatureValue>
1195 <ds:KeyInfo>
1196     <wsse:SecurityTokenReference xmlns:wsu="http://docs.oasis-
1197 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
1198     wsu:Id="XWSSGID-1172790300633-442423344">
1199     <wsse:Reference URI="#XWSSGID-1172790302111-1738806553"
1200     ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1201 wss-x509-token-profile-1.0#X509v3"/>
1202     </wsse:SecurityTokenReference>
1203     </ds:KeyInfo>
1204 </ds:Signature>
1205     <wsu:Timestamp xmlns:wsu="http://docs.oasis-
1206 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
1207     wsu:Id="XWSSGID-1172790300637708871805">
1208     <wsu:Created>2007-03-01T23:05:00Z</wsu:Created>
1209     <wsu:Expires>2007-03-01T23:05:05Z</wsu:Expires>
1210     </wsu:Timestamp>
1211 </wsse:Security>
1212 </SOAP-ENV:Header>
1213 <SOAP-ENV:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1214 wss-wssecurity-utility-1.0.xsd" wsu:Id="XWSSGID-1172790300636-653454040">
1215     <ekmi:KeyCachePolicyRequest xmlns:ekmi="http://docs.oasis-
1216 open.org/ekmi/2008/01"/>
1217 </SOAP-ENV:Body>
1218 </SOAP-ENV:Envelope>

```

1219 3.12 Response with a symmetric key-caching policy (1)

1220 After an SKS server has performed its operations of authenticating a **KeyCachePolicyRequest**, identifying the
1221 requester, determining policies that apply to the requester, it assembles the following response and returns it to
1222 the client. (The SOAP message, as indicated earlier, is secured using WSS, but only the actual SKSML content
1223 is displayed and discussed here).

```

1224 [m01] <ekmi:KeyCachePolicyResponse
1225 [m02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1226 [m03]     xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
1227 [m04]     <ekmi:KeyCachePolicy>
1228 [m05]         <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
1229 [m06]         <ekmi:PolicyName>No Caching Policy</ekmi:PolicyName>
1230 [m07]         <ekmi:Description>
1231 [m08]             This policy is for high-risk, always-connected machines on the
1232 [m09]             network, which will never cache symmetric keys locally. This
1233 [m10]             policy never expires (but checks monthly for any updates).
1234 [m11]         </ekmi:Description>
1235 [m12]         <ekmi:KeyClass>NoCachingClass</ekmi:KeyClass>
1236 [m13]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1237 [m14]         <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>

```


1238 [m15] <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1239 [m16] <ekmi:Status>Active</ekmi:Status>
1240 [m17] </ekmi:KeyCachePolicy>
1241 [m18] </ekmi:KeyCachePolicyResponse>

1242 [m01] is the start of the *KeyCachePolicyResponse* element.

1243 [m02] and [m03] identify the namespaces to which this XML conforms, and the location of their XML Schema
1244 Definitions (XSD).

1245 [m04] is the start of the first – and only - *KeyCachePolicy* element.

1246 [m05] identifies the *KeyCachePolicyID* (KCPID) assigned to this policy by the SKS server. In this example, the
1247 concatenated values of the Domain ID and Policy ID indicate that the key belongs to the organization
1248 represented by the PEN, 10514; and is the first key-caching policy within the SKMS.

1249 [m06] provides a descriptive name for this key-cache policy through the *PolicyName* element, which is helpful to
1250 human readers when identifying this policy.

1251 [m07] is the start tag of the *Description* element.

1252 [m08] - [m10] provides a human-readable description about this key-cache policy.

1253 [m11] is the closing tag of the *Description* element.

1254 [m12] specifies the *KeyClass* to which this policy applies. Only keys that belong to this key-class are subject to
1255 this caching policy.

1256 [m13] specifies the date and time that this *KeyCachePolicy* is effective. This is accomplished through a
1257 *StartDate* element. In this example, the policy is effective as of January 01, 2008.

1258 [m14] specifies the date and time that this *KeyCachePolicy* becomes invalid. This is accomplished through a
1259 *EndDate* element. In this example, the use of the UNIX "epoch" date (January 01, 1969) indicates that this
1260 policy never expires.

1261 [m15] specifies the frequency at which this client must check with the SKS server for updates to the key-caching
1262 policy. This is specified in seconds in the *PolicyCheckInterval* element; in this example it is a monthly interval.

1263 [m16] indicates the *Status* of this *KeyCachePolicy* and whether it is an active policy or not.

1264 [m17] is the closing tag of the *KeyCachePolicy* element.

1265 [m18] is the closing tag of the *KeyCachePolicyResponse* element.

1266 3.13 Response with a symmetric key-caching policy (2)

1267 This is a second example of a key-caching policy response that has additional elements in the policy permitting
1268 caching and specify the number of unused and used (for encryption) symmetric keys that may be cached by the
1269 client machine.

```

1270 [m01] <ekmi:KeyCachePolicyResponse
1271 [m02]       xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1272 [m03]       xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
1273 [m04]       <ekmi:KeyCachePolicy>
1274 [m05]           <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
1275 [m06]           <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
1276 [m07]           <ekmi:Description>
1277 [m08]               This policy defines how company-issued laptops will manage symmetric
1278 [m09]               keys used for file/disk encryption in their local cache. This
1279 [m10]               policy must be used by all laptops that use the company EKMI.
1280 [m11]           </ekmi:Description>
1281 [m12]           <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
1282 [m13]           <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>

```

```

1283 [m14] <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
1284 [m15] <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1285 [m16] <ekmi:Status>Active</ekmi:Status>
1286 [m17] <ekmi:NewKeysCacheDetail>
1287 [m18] <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1288 [m19] <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1289 [m20] </ekmi:NewKeysCacheDetail>
1290 [m21] <ekmi:UsedKeysCacheDetail>
1291 [m22] <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1292 [m23] <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1293 [m24] </ekmi:UsedKeysCacheDetail>
1294 [m25] </ekmi:KeyCachePolicy>
1295 [m26] </ekmi:KeyCachePolicyResponse>

```

1296 [m01] is the start of the **KeyCachePolicyResponse** element.

1297 [m02] and [m03] identify the namespaces to which this XML conforms, and the location of their XML Schema
1298 Definitions (XSD).

1299 [m04] is the start of the first – and only - **KeyCachePolicy** element.

1300 [m05] identifies the **KeyCachePolicyID** (KCPID) assigned by the SKS server for the key-caching policy being
1301 returned.

1302 [m06] provides a descriptive name for this key-cache policy through the **PolicyName** element, which is helpful to
1303 human readers when identifying this policy.

1304 [m07] is the start tag of the **Description** element.

1305 [m08] - [m10] provides a human-readable description about this key-cache policy.

1306 [m11] is the closing tag of the **Description** element.

1307 [m12] specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are subject to
1308 this caching policy.

1309 [m13] specifies the date and time that this **KeyCachePolicy** is effective. This is accomplished through a
1310 **StartDate** element. In this example, the policy is effective as of January 01, 2008.

1311 [m14] specifies the date and time that this **KeyCachePolicy** becomes invalid. This is accomplished through a
1312 **EndDate** element. In this example, the policy expires on December 31, 2008.

1313 [m15] specifies the frequency at which this client must check with the SKS server for updates to the key-caching
1314 policy. This is specified in seconds in the **PolicyCheckInterval** element; in this example it is a monthly interval.

1315 [m16] indicates the **Status** of this **KeyCachePolicy** and whether it is an active policy or not.

1316 [m17] is the start of the **NewKeysCacheDetail** element, which provides details about how many new symmetric
1317 keys – that haven't been used for any encryption transactions – may be cached by the client and for how long.

1318 [m18] indicates the maximum number of new (unused) symmetric keys that may be cached by the client. This is
1319 specified through the **MaximumKeys** element. When the client uses a symmetric key, this reduces the number
1320 of new symmetric keys. In this case, the SKCL connects to the SKS server (if it is on the network) and requests
1321 a new symmetric key to add to its new-key cache.

1322 [m19] indicates the maximum duration that new (unused) symmetric keys may be cached by the client. This is
1323 specified through the **MaximumDuration** element in seconds. If there are any new keys that exceed this
1324 duration limit, the SKCL deletes the specific symmetric key and replaces it with a new symmetric key from the
1325 SKS server.

1326 [m20] is the closing tag of the **NewKeysCacheDetail** element.

1327 [m21] is the start of the *UsedKeysCacheDetail* element, which provides details about how many used
1328 symmetric keys – those that HAVE been used for any encryption transactions – may be cached by the client and
1329 for how long.

1330 [m22] indicates the maximum number of used symmetric keys that may be cached by the client through the
1331 *MaximumKeys* element. If the client already has the maximum number of used-keys in its cache, using the
1332 First-In-First-Out (FIFO) method, it deletes the oldest symmetric key in the cache to replace with the key that
1333 transitioned from the “new” to “used” status.

1334 [m23] indicates the maximum duration that used symmetric keys may be cached by the client through the
1335 *MaximumDuration* element in seconds. If there are any used keys that exceed this duration limit, the SKCL
1336 deletes the specific symmetric key. While this may temporarily reduce the number of used symmetric keys in the
1337 cache, the SKCL takes the most conservative position when making this decision.

1338 [m24] is the closing tag of the *UsedKeysCacheDetail* element.

1339 [m25] is the closing tag of the *KeyCachePolicy* element.

1340 [m26] is the closing tag of the *KeyCachePolicyResponse* element.

1341 3.14 Response with multiple symmetric key-caching policies (3)

1342 This is a third example of a key-caching policy response that has multiple key-cache policies that apply to
1343 different classes of symmetric keys.

```

1344 [m01] <ekmi:KeyCachePolicyResponse
1345 [m02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1346 [m03]     xmlns:xenc='http://www.w3.org/2001/04/xmenc#'>
1347 [m04]     <ekmi:KeyCachePolicy>
1348 [m05]         <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
1349 [m06]         <ekmi:PolicyName>No Caching Policy</ekmi:PolicyName>
1350 [m07]         <ekmi:Description>
1351 [m08]             This policy is for high-risk, always-connected machines on the
1352 [m09]             network, which will never cache symmetric keys locally. This
1353 [m10]             policy never expires (but checks monthly for any updates).
1354 [m11]         </ekmi:Description>
1355 [m12]         <ekmi:KeyClass>NoCachingClass</ekmi:KeyClass>
1356 [m13]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1357 [m14]         <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
1358 [m15]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1359 [m16]         <ekmi:Status>Active</ekmi:Status>
1360 [m17]     </ekmi:KeyCachePolicy>
1361 [m18]     <ekmi:KeyCachePolicy>
1362 [m19]         <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
1363 [m20]         <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
1364 [m21]         <ekmi:Description>
1365 [m22]             This policy defines how company-issued laptops will manage symmetric
1366 [m23]             keys used for file/disk encryption in their local cache. This
1367 [m24]             policy must be used by all laptops that use the company EKMI.
1368 [m25]         </ekmi:Description>
1369 [m26]         <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
1370 [m27]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1371 [m28]         <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
1372 [m29]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1373 [m30]         <ekmi:Status>Active</ekmi:Status>
1374 [m31]         <ekmi:NewKeysCacheDetail>
1375 [m32]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1376 [m33]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1377 [m34]         </ekmi:NewKeysCacheDetail>
1378 [m35]     </ekmi:KeyCachePolicy>

```



```

1379 [m36]         <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1380 [m37]         <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1381 [m38]         </ekmi:UsedKeysCacheDetail>
1382 [m39]     </ekmi:KeyCachePolicy>
1383 [m40]     <ekmi:KeyCachePolicy>
1384 [m41]         <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
1385 [m42]         <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
1386 [m43]         <ekmi:Description>
1387 [m44]             This policy defines how company-issued laptops will manage
1388 [m45]             symmetric keys used for file/disk encryption in their local
1389 [m46]             cache. This policy must be used by all laptops.
1390 [m47]         </ekmi:Description>
1391 [m48]         <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
1392 [m49]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1393 [m50]         <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
1394 [m51]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1395 [m52]         <ekmi:Status>Active</ekmi:Status>
1396 [m53]         <ekmi:NewKeysCacheDetail>
1397 [m54]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1398 [m55]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1399 [m56]         </ekmi:NewKeysCacheDetail>
1400 [m57]         <ekmi:UsedKeysCacheDetail>
1401 [m58]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1402 [m59]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1403 [m60]         </ekmi:UsedKeysCacheDetail>
1404 [m61]     </ekmi:KeyCachePolicy>
1405 [m62] </ekmi:KeyCachePolicyResponse>

```

1406

1407 [m01] is the start of the *KeyCachePolicyResponse* element.

1408 [m02] and [m03] identify the namespaces to which this XML conforms, and the location of their XML Schema
1409 Definitions (XSD).

1410 [m04] is the start of the first of three *KeyCachePolicy* elements in this response.

1411 [m05] identifies the *KeyCachePolicyID* (KCPIID) assigned to this policy by the SKS server. In this example, the
1412 concatenated values of the Domain ID and Policy ID indicate that the key belongs to the organization
1413 represented by the PEN, 10514; and is the first key-caching policy within the SKMS.

1414 [m06] provides a descriptive name for this key-cache policy through the *PolicyName* element, which is helpful to
1415 human readers when identifying this policy.

1416 [m07] is the start tag of the *Description* element.

1417 [m08] - [m10] provides a human-readable description about this key-cache policy.

1418 [m11] is the closing tag of the *Description* element.

1419 [m12] specifies the *KeyClass* to which this policy applies. Only keys that belong to this key-class are subject to
1420 this caching policy.

1421 [m13] specifies the date and time that this *KeyCachePolicy* is effective. This is accomplished through a
1422 *StartDate* element. In this example, the policy is effective as of January 01, 2008.

1423 [m14] specifies the date and time that this *KeyCachePolicy* becomes invalid. This is accomplished through a
1424 *EndDate* element. In this example, the use of the UNIX "epoch" date (January 01, 1969) indicates that this
1425 policy never expires.

1426 [m15] specifies the frequency at which this client must check with the SKS server for updates to the key-caching
1427 policy. This is specified in seconds in the *PolicyCheckInterval* element; in this example it is a monthly interval.

1428 [m16] indicates the **Status** of this *KeyCachePolicy* and whether it is an active policy or not.

1429 [m17] is the closing tag of the first *KeyCachePolicy* element.

1430 [m18] is the start of the second of three *KeyCachePolicy* elements in this response.

1431 [m19] identifies the *KeyCachePolicyID* (KCPID) assigned by the SKS server for the key-caching policy being
1432 returned.

1433 [m20] provides the descriptive name for this key-cache policy through the *PolicyName* element.

1434 [m21] is the start tag of the *Description* element.

1435 [m22] - [m24] provides a human-readable description about this key-cache policy.

1436 [m25] is the closing tag of the *Description* element.

1437 [m26] specifies the *KeyClass* to which this policy applies. Only keys that belong to this key-class are subject to
1438 this caching policy.

1439 [m27] specifies the date and time that this *KeyCachePolicy* is effective. This is accomplished through a
1440 *StartDate* element. In this example, the policy is effective as of January 01, 2008.

1441 [m28] specifies the date and time that this *KeyCachePolicy* becomes invalid. This is accomplished through a
1442 *EndDate* element. In this example, the policy expires on December 31, 2008.

1443 [m29] specifies the frequency at which this client must check with the SKS server for updates to the key-caching
1444 policy. This is specified in seconds in the *PolicyCheckInterval* element; in this example it is a monthly interval.

1445 [m30] indicates the **Status** of this *KeyCachePolicy* and whether it is an active policy or not.

1446 [m31] is the start of the *NewKeysCacheDetail* element, which provides details about how many new symmetric
1447 keys – that haven't been used for any encryption transactions – may be cached by the client and for how long.

1448 [m32] indicates the maximum number of new (unused) symmetric keys that may be cached by the client. This is
1449 specified through the *MaximumKeys* element. When the client uses a symmetric key, this reduces the number
1450 of new symmetric keys. In this case, the SKCL connects to the SKS server (if it is on the network) and requests
1451 a new symmetric key to add to its new-key cache.

1452 [m33] indicates the maximum duration that new (unused) symmetric keys may be cached by the client. This is
1453 specified through the *MaximumDuration* element in seconds. If there are any new keys that exceed this
1454 duration limit, the SKCL deletes the specific symmetric key and replaces it with a new symmetric key from the
1455 SKS server.

1456 [m34] is the closing tag of the *NewKeysCacheDetail* element.

1457 [m35] is the start of the *UsedKeysCacheDetail* element, which provides details about how many used
1458 symmetric keys – those that HAVE been used for any encryption transactions – may be cached by the client and
1459 for how long.

1460 [m36] indicates the maximum number of used symmetric keys that may be cached by the client through the
1461 *MaximumKeys* element. If the client already has the maximum number of used-keys in its cache, using the
1462 First-In-First-Out (FIFO) method, it deletes the oldest symmetric key in the cache to replace with the key that
1463 transitioned from the “new” to “used” status.

1464 [m37] indicates the maximum duration that used symmetric keys may be cached by the client through the
1465 *MaximumDuration* element in seconds. If there are any used keys that exceed this duration limit, the SKCL
1466 deletes the specific symmetric key. While this may temporarily reduce the number of used symmetric keys in the
1467 cache, the SKCL takes the most conservative position when making this decision.

1468 [m38] is the closing tag of the *UsedKeysCacheDetail* element.

1469 [m39] is the closing tag of the second *KeyCachePolicy* element.

1470 [m40] is the start of the third *KeyCachePolicy* element in this response.

1471 [m41] identifies the *KeyCachePolicyID* (KCPID) assigned by the SKS server for the key-caching policy being
1472 returned.

1473 [m42] provides the descriptive name for this key-cache policy through the *PolicyName* element.

1474 [m43] is the start tag of the *Description* element.

1475 [m44] - [m46] provides a human-readable description about this key-cache policy.

1476 [m47] is the closing tag of the *Description* element.

1477 [m48] specifies the *KeyClass* to which this policy applies. Only keys that belong to this key-class are subject to
1478 this caching policy.

1479 [m49] specifies the date and time that this *KeyCachePolicy* is effective.

1480 [m50] specifies the date and time that this *KeyCachePolicy* becomes invalid.

1481 [m51] specifies the frequency at which this client must check with the SKS server for updates to the key-caching
1482 policy.

1483 [m52] indicates the *Status* of this *KeyCachePolicy* and whether it is an active policy or not.

1484 [m53] is the start of the *NewKeysCacheDetail* element, which provides details about how many new symmetric
1485 keys may be cached by the client and for how long.

1486 [m54] indicates the maximum number of new (unused) symmetric keys that may be cached by the client. This is
1487 specified through the *MaximumKeys* element. When the client uses a symmetric key, this reduces the number
1488 of new symmetric keys. In this case, the SKCL connects to the SKS server (if it is on the network) and requests
1489 a new symmetric key to add to its new-key cache.

1490 [m55] indicates the maximum duration that new (unused) symmetric keys may be cached by the client. This is
1491 specified through the *MaximumDuration* element in seconds. If there are any new keys that exceed this
1492 duration limit, the SKCL deletes the specific symmetric key and replaces it with a new symmetric key from the
1493 SKS server.

1494 [m56] is the closing tag of the *NewKeysCacheDetail* element.

1495 [m57] is the start of the *UsedKeysCacheDetail* element, which provides details about how many used
1496 symmetric keys – those that HAVE been used for any encryption transactions – may be cached by the client and
1497 for how long.

1498 [m58] indicates the maximum number of used symmetric keys that may be cached by the client through the
1499 *MaximumKeys* element. If the client already has the maximum number of used-keys in its cache, using the
1500 First-In-First-Out (FIFO) method, it deletes the oldest symmetric key in the cache to replace with the key that
1501 transitioned from the “new” to “used” status.

1502 [m59] indicates the maximum duration that used symmetric keys may be cached by the client through the
1503 *MaximumDuration* element in seconds. If there are any used keys that exceed this duration limit, the SKCL
1504 deletes the specific symmetric key. While this may temporarily reduce the number of used symmetric keys in the
1505 cache, the SKCL takes the most conservative position when making this decision.

1506 [m60] is the closing tag of the *UsedKeysCacheDetail* element.

1507 [m61] is the closing tag of the third and final *KeyCachePolicy* element in this response.

1508 [m62] is the closing tag of the *KeyCachePolicyResponse* element.

4 Specification

1509

4.1 Element <SymkeyRequest>

1510

1511 The <SymkeyRequest> element identifies one or more *GlobalKeyID*'s of symmetric encryption keys needed by
1512 the client application. The request may also specify one or more *KeyClass* elements for the requested key
1513 when the request is for a new symmetric key.

1514 While it is a top-level element within this specification, a <SymkeyRequest> element MUST be enclosed within a
1515 *SOAP Body* element of a *SOAP Envelope* to conform to the security requirements of this specification. The
1516 *SOAP Header* of the *SOAP Envelope* MUST enclose a *Security* element conforming to [WSS] with a
1517 *ValueType* attribute containing the value <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3>. The *Security* element must conform to all other requirements of the specified security profile
1518 in [WSS] to form a well-formed, secure message.
1519

Schema Definition:

1520

```
1521 <xsd:element name="SymkeyRequest">  
1522   <xsd:complexType>  
1523     <xsd:sequence>  
1524       <xsd:element  
1525         name="GlobalKeyID"  
1526         type="ekmi:GlobalKeyIDType"  
1527         minOccurs="1"  
1528         maxOccurs="unbounded">  
1529       </xsd:element>  
1530       <xsd:element  
1531         name="KeyClasses"  
1532         type="ekmi:KeyClassesType"  
1533         minOccurs="0">  
1534       </xsd:element>  
1535     </xsd:sequence>  
1536   </xsd:complexType>  
1537 </xsd:element>
```

1538 The <SymkeyRequest> element consists of a sequence of two child elements:

1. <GlobalKeyID> [Required]

1539

1540

1541 This element of type *GlobalKeyIDType*, identifies the unique global key identifier of the requested
1542 symmetric key within the target Symmetric Key Management System (SKMS) the client is
1543 communicating with. There MUST be at least one <GlobalKeyID> element in a <SymkeyRequest>,
1544 but there may be an unbounded (unlimited) number of <GlobalKeyID> elements specified.

1545

1546

The <GlobalKeyID> element is specified in Section 4.2.

2. <KeyClasses> [Optional]

1547

1548

1549 This element of type *KeyClassesType*, when specified, identifies at least one <KeyClass> element, but
1550 may specify an unbounded (unlimited) number of <KeyClass> elements within the <KeyClasses> set.
1551 Client applications may request one or more symmetric keys conforming to one or more key classes
1552 required by the application. If the client application is authorized to receive keys conforming to such key
1553 classes, the SKS server will generate and supply them.

1554

1555

When more than one <GlobalKeyID> for a new symmetric key is specified in the request, there MAY
1556 be only one <KeyClass> element within the <KeyClasses> set.

1557

1558

1559

When the client requires more than one new symmetric key, and each key is required to be of a different
key class, there MUST be only one <GlobalKeyID> element followed by as many <KeyClass>

1560 elements inside the <KeyClasses> set, as needed by the client application.
1561
1562 When a client requires multiple symmetric keys of two or more key classes, the client MUST send
1563 multiple requests to the SKS server. See examples 4 and 5 below in this section.
1564

1565 The <KeyClasses> and <KeyClass> elements are specified in Section 4.3.

1566 Some examples of the <SymkeyRequest> element are as follows:

1567 **Example 1 – A single new symmetric key request of a default key class:**

```
1568 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1569 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1570 </ekmi:SymkeyRequest>
```

1571 **Example 2 – A request for three new symmetric keys of a default key class for each symmetric key:**

```
1572 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1573 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1574 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1575 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1576 </ekmi:SymkeyRequest>
```

1577 **Example 3 – A request for a single new symmetric key of a specific key class:**

```
1578 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1579 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1580 <ekmi:KeyClasses>  
1581 <ekmi:KeyClass>HR-Class</ekmi:KeyClass>  
1582 </ekmi:KeyClasses>  
1583 </ekmi:SymkeyRequest>
```

1584 **Example 4 – A request for a two new symmetric keys with the same key class for each symmetric key:**

```
1585 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1586 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1587 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1588 <ekmi:KeyClasses>  
1589 <ekmi:KeyClass>FIN-FX</ekmi:KeyClass>  
1590 </ekmi:KeyClasses>  
1591 </ekmi:SymkeyRequest>
```

1592 **Example 5 – A request for a nine new symmetric keys of different key classes:**

```
1593 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1594 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1595 <ekmi:KeyClasses>  
1596 <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>  
1597 <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>  
1598 <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>  
1599 <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>  
1600 <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>  
1601 <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>  
1602 <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>  
1603 <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>  
1604 <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>  
1605 </ekmi:KeyClasses>  
1606 </ekmi:SymkeyRequest>
```

1607 4.2 Element <GlobalKeyID>

1608 The <GlobalKeyID> element is the unique identifier of a symmetric encryption key within an SKMS. Every
1609 symmetric key generated by the SKS server MUST be assigned a unique <GlobalKeyID> as specified in this
1610 section.

1611 Schema Definition:

```
1612 <xsd:simpleType name="GlobalKeyIDType">  
1613   <xsd:restriction base="xsd:string">  
1614     <xsd:minLength value="5"/>  
1615     <xsd:maxLength value="62"/>  
1616     <xsd:pattern value="[0-9]{1,20}-[0-9]{1,20}-[0-9]{1,20}"/>  
1617     <xsd:whiteSpace value="collapse"/>  
1618   </xsd:restriction>  
1619 </xsd:simpleType>
```

1620 The <GlobalKeyID> element is of the *GlobalKeyIDType*, and is a string identifier of a symmetric key
1621 consisting of five parts concatenated together:

- 1622 1. A positive integer identifying the *Domain ID*. The *DomainID* identifies the IANA-issued Private
1623 Enterprise Number (PEN) as published at <http://www.iana.org/assignments/enterprise-numbers> and is
1624 used by the SKS server to constrain the ownership of objects within the SKMS;
- 1625 2. A literal hyphen ("-") without surrounding spaces;
- 1626 3. A positive integer identifying the Server ID of the server that originally generated the key;
- 1627 4. Another literal hyphen ("-") without surrounding spaces;
- 1628 5. A positive integer identifying the Key ID;

1629 Combined, the five components of this element make up a unique identifier for a symmetric key within the SKMS.
1630 Since all enterprises are expected to use only the PENs assigned to them, technically the <GlobalKeyID> is
1631 unique across the internet.

1632 The *DomainID* part of the <GlobalKeyID> element MUST be a positive integer in the range of 0 (zero) to
1633 18446744073709551615 (20-byte ASCII decimal).

1634 When an SKMS manages the symmetric keys for a single enterprise, the *DomainID* part of the <GlobalKeyID>
1635 element in a <SymkeyRequest> MAY be zero ("0"). When an SKMS manages symmetric keys for multiple
1636 enterprises, the *DomainID* in the <GlobalKeyID> of a <SymkeyRequest> MUST be positive and non-zero. In
1637 such a situation, the client application will request a symmetric key for the domain in which it is authorized to
1638 request and receive keys.

1639 The *DomainID* in the <GlobalKeyID> element of a <SymkeyResponse> MUST always be positive and non-
1640 zero. It will typically contain the PEN of the domain to which the symmetric key belongs.

1641 The *ServerID* part of the <GlobalKeyID> element MUST be a positive integer in the range of 0 (zero) to
1642 18446744073709551615 (20-byte ASCII decimal).

1643 The *ServerID* part of the <GlobalKeyID> element of a <SymkeyRequest> MUST always be zero ("0").

1644 The *ServerID* part of the <GlobalKeyID> element of a <SymkeyResponse> MUST always be positive and non-
1645 zero. It will typically contain the unique server identifier of the SKS server where the symmetric key was
1646 generated.

1647 The *KeyID* part of the <GlobalKeyID> element MUST be a positive integer in the range of 0 (zero) to
1648 18446744073709551615 (20-byte ASCII decimal).

1649 The *KeyID* part of the <GlobalKeyID> element of a <SymkeyRequest> MUST always be zero ("0").

1650 The **KeyID** part of the <GlobalKeyID> element of a <SymkeyResponse> MUST always be positive and non-
1651 zero. It will typically contain the unique key identifier of the symmetric key within the SKS server where the key
1652 was generated.

1653 **Example 1 – A <GlobalKeyID> value for a new symmetric key from an SKMS that serves a single domain:**

```
1654 <ekmi:GlobalKeyID>0-0-0</ekmi:GlobalKeyID>
```

1655 **Example 2 – A <GlobalKeyID> value for a new symmetric key for the domain with the PEN 10514, from an**
1656 **SKMS that serves multiple domains:**

```
1657 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
```

1658 **Example 3 – A <GlobalKeyID> value for the 16,777,215th symmetric key generated on 2nd SKS server for**
1659 **an enterprise with the PEN 10514, in either a <SymkeyRequest> or a <SymkeyResponse>:**

```
1660 <ekmi:GlobalKeyID>10514-2-16777215</ekmi:GlobalKeyID>
```

1661 **Example 4 – The maximum <GlobalKeyID> value possible (a 62-byte ASCII decimal), in a**
1662 **<SymkeyRequest> or <SymkeyResponse>:**

```
1663 <ekmi:GlobalKeyID>  
1664 18446744073709551615-18446744073709551615-18446744073709551615  
1665 </ekmi:GlobalKeyID>
```

1666 4.3 Element <KeyClasses> and <KeyClass>

1667 The <KeyClasses> element of type **KeyClassesType**, when specified, identifies at least one <KeyClass>
1668 element, but may specify an unbounded (unlimited) number of <KeyClass> elements within the <KeyClasses>
1669 set.

1670 Schema Definition:

```
1671 <xsd:complexType name="KeyClassesType">  
1672 <xsd:sequence>  
1673 <xsd:element  
1674 name="KeyClass"  
1675 type="tns:KeyClassType"  
1676 minOccurs="1"  
1677 maxOccurs="unbounded"/>  
1678 </xsd:sequence>  
1679 </xsd:complexType>
```

1680

```
1681 <xsd:simpleType name="KeyClassType">  
1682 <xsd:restriction base="xsd:string">  
1683 <xsd:maxLength value="255"/>  
1684 </xsd:restriction>  
1685 </xsd:simpleType>
```

1686 Client applications may request one or more symmetric keys conforming to one or more key classes required by
1687 the application. If the client application is authorized to receive keys conforming to such key classes, the SKS
1688 server will generate and supply them.

1689

1690 The <KeyClasses> element is useful only when requesting new symmetric keys, i.e. symmetric encryption keys
1691 that have previously NOT been used for encrypting data. There is little reason for a client application to specify
1692 the <KeyClasses> element when requesting an existing (escrowed) symmetric key, since the SKS server will
1693 return the requested key to authorized clients with whatever key class is associated with the key regardless of
1694 what key class is specified in the request. The key class will have been associated with the symmetric key at the
1695 time of its generation and cannot be changed once associated with a key.

1696

1697 When more than one <GlobalKeyID> is specified in the request, there MAY be only one <KeyClass> element
1698 within the <KeyClasses> set. When a key class is not specified in a request, it implies a request for symmetric
1699 key(s) of a default key class configured at the **SKS** server. The default key class for a site is site-specific.
1700

1701 When the client requires more than one symmetric key, and each key needs to be of a different key class, there
1702 MUST be only one <GlobalKeyID> element followed by as many <KeyClass> elements inside the
1703 <KeyClasses> set as needed by the client application. (Example 5 in this section).
1704

1705 When a client requires many symmetric keys – say five keys – and two or more keys belong to the same key
1706 class, the client MUST send multiple requests to the **SKS** server. One request will contain multiple
1707 <GlobalKeyID> elements with one <KeyClass> element in the <KeyClasses> set, and the other request will
1708 contain one <GlobalKeyID> element and multiple <KeyClass> elements within the <KeyClasses> set.
1709 (Examples 4 and 5 in this section).

1710 **Example 1 – A symmetric key request of a default key class (when no KeyClass is specified):**

```
1711 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1712 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1713 </ekmi:SymkeyRequest>
```

1714 **Example 2 – A request for multiple new symmetric keys, each of a default key class:**

```
1715 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1716 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1717 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1718 </ekmi:SymkeyRequest>
```

1719 **Example 3 – A request for a new symmetric key of a specific key class:**

```
1720 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1721 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1722 <ekmi:KeyClasses>  
1723 <ekmi:KeyClass>256-Bit-Class</ekmi:KeyClass>  
1724 </ekmi:KeyClasses>  
1725 </ekmi:SymkeyRequest>
```

1726 **Example 4 – A request for two new symmetric keys of the same key class for each symmetric key. Note**
1727 **that if the FIN-FX key class was the default key class, a request as shown in Example 2 of this section**
1728 **would result in the same response:**

```
1729 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1730 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1731 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1732 <ekmi:KeyClasses>  
1733 <ekmi:KeyClass>FIN-FX</ekmi:KeyClass>  
1734 </ekmi:KeyClasses>  
1735 </ekmi:SymkeyRequest>
```

1736 **Example 5 – A request for a four new symmetric keys of different key classes:**

```
1737 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1738 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1739 <ekmi:KeyClasses>  
1740 <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>  
1741 <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>  
1742 <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>  
1743 <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>  
1744 </ekmi:KeyClasses>  
1745 </ekmi:SymkeyRequest>
```


1746 4.4 Element <SymkeyResponse>

1747 The <SymkeyResponse> element is one of two results returned by an **SKS** server upon being sent a valid and
1748 authorized <SymkeyRequest> by a client application. The other result is a <SymkeyError> which will be
1749 discussed in the next section.

1750 While <SymkeyResponse> is a top-level element within this specification, it **MUST** be enclosed within a **SOAP**
1751 **Body** element of a **SOAP Envelope** to conform to the security requirements of this specification. The **SOAP**
1752 **Header** of the **SOAP Envelope** **MUST** enclose a **Security** element conforming to [WSS] with a **ValueType**
1753 attribute containing the value <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3>. The **Security** element must conform to all other requirements of the specified security profile
1754 in [WSS] to form a well-formed, secure message.
1755

1756 Schema Definition:

```
1757 <xsd:element name="SymkeyResponse">  
1758 <xsd:complexType>  
1759 <xsd:sequence>  
1760 <xsd:element  
1761 name="Symkey"  
1762 type="tns:SymkeyType"  
1763 minOccurs="0"  
1764 maxOccurs="unbounded"/>  
1765 <xsd:element  
1766 name="SymkeyError"  
1767 type="tns:SymkeyErrorType"  
1768 minOccurs="0"  
1769 maxOccurs="unbounded"/>  
1770 </xsd:sequence>  
1771 </xsd:complexType>  
1772 </xsd:element>
```

1773 The <SymkeyResponse> element consists of a sequence of two types of child elements - <Symkey> or
1774 <SymkeyError>. The <SymkeyResponse> element **MAY** consist of either type of element or both types of
1775 elements. When both elements are contained in a <SymkeyResponse>, all <Symkey> elements **MUST** precede
1776 the first <SymkeyError> element.

1777 1. <Symkey> [Optional]

1778
1779 This element of type **SymkeyType**, is returned by the **SKS** server in response to a successful
1780 processing of a <SymkeyRequest>. There **MAY** be more than one <Symkey> element in the
1781 <SymkeyResponse> if the client application made a request for multiple symmetric keys.
1782

1783 The <Symkey> element and the **SymkeyType** are specified in Section 4.5.

1784 2. <SymkeyError> [Optional]

1785
1786 This element of type **SymkeyErrorType**, contains a response to a failed attempt in processing a
1787 request for one or more symmetric keys. There **MAY** be more than one <SymkeyError> element in
1788 the <SymkeyResponse> if the client application made a request for multiple symmetric keys and the
1789 request resulted in multiple errors.
1790

1791 The <SymkeyError> element is specified in Section 4.6.

1792 Some high-level examples of the <SymkeyResponse> element are as follows:

1793 Example 1 – A response with a single symmetric key:

```
1794 <ekmi:SymkeyResponse  
1795 xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1796 <ekmi:Symkey>.....</ekmi:Symkey>  
1797 </ekmi:SymkeyResponse>
```

1798 **Example 2 – A response with three symmetric keys:**

```
1799 <ekmi:SymkeyResponse
1800     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1801     <ekmi:Symkey>.....</ekmi:Symkey>
1802     <ekmi:Symkey>.....</ekmi:Symkey>
1803     <ekmi:Symkey>.....</ekmi:Symkey>
1804 </ekmi:SymkeyResponse>
```

1805 **Example 3 – A response with an error:**

```
1806 <ekmi:SymkeyResponse
1807     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1808     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
1809 </ekmi:SymkeyResponse>
```

1810 **Example 4 – A response with multiple errors:**

```
1811 <ekmi:SymkeyResponse
1812     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1813     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
1814     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
1815 </ekmi:SymkeyResponse>
```

1816 **Example 5 – A response with one symmetric key and one error:**

```
1817 <ekmi:SymkeyResponse
1818     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1819     <ekmi:Symkey>.....</ekmi:Symkey>
1820     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
1821 </ekmi:SymkeyResponse>
```

1822 **Example 6 – A response with multiple symmetric keys and multiple error:**

```
1823 <ekmi:SymkeyResponse
1824     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1825     <ekmi:Symkey>.....</ekmi:Symkey>
1826     <ekmi:Symkey>.....</ekmi:Symkey>
1827     <ekmi:Symkey>.....</ekmi:Symkey>
1828     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
1829     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
1830 </ekmi:SymkeyResponse>
```

1831 **4.5 Element <Symkey>**

1832 The <Symkey> element is the *raison d'etre* of the SKSML protocol. The element of type *SymkeyType*, contains
1833 the symmetric key returned by the SKS server, in response to a successful processing of a <SymkeyRequest>
1834 from a client application.

1835 **Schema Definition:**

```
1836 <xsd:complexType name="SymkeyType">
1837     <xsd:sequence>
1838         <xsd:element name="GlobalKeyID" type="ekmi:GlobalKeyIDType"/>
1839         <xsd:element name="KeyUsePolicy" type="ekmi:KeyUsePolicyType"/>
1840         <xsd:element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>
1841         <xsd:element ref="xenc:CipherData"/>
1842     </xsd:sequence>
1843 </xsd:complexType>
```

1844 When a request for a symmetric key is successful, there MUST be at least one <Symkey> element in a
1845 <SymkeyResponse> element. There MAY be more than one <Symkey> element in the response if the client
1846 application made a request for multiple symmetric keys and the SKS server processed the request successfully.

1847 In the event of an error in processing the request, there SHALL be no <Symkey> element in the response; there
1848 SHALL be a <SymkeyError> element, instead. The <SymkeyError> element is specified in Section 4.6.

1849 The <Symkey> element consists of a sequence of the following child elements:

1850 1. <GlobalKeyID> [Required]

1851 This element of type **GlobalKeyIDType** identifies the unique identifier of the symmetric key within an
1852 SKMS. There SHALL be only one <GlobalKeyID> within a <Symkey> element.

1853 The **GlobalKeyIDType** is specified in Section 4.2.

1856 2. <KeyUsePolicy> [Required]

1857 This element of type **KeyUsePolicyType**, defines how the symmetric key in this <Symkey> element may
1858 be used by applications. There SHALL be only one <KeyUsePolicy> element within a <Symkey>
1859 element.

1860 The <KeyUsePolicy> element is specified in Section 4.7.

1863 3. <EncryptionMethod> [Required]

1864 This element of type **EncryptionMethodType** from [XMLEncryption] describes how the symmetric key
1865 in this <Symkey> element is encrypted for transport between the SKS Server and the client application.

1866 The <EncryptionMethod> MUST specify one of the following two transport algorithms in the
1867 **Algorithm** attribute of the element:

1868 - http://www.w3.org/2001/04/xmlenc#rsa-1_5
1869 - <http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p>

1873 4. <CipherData> [Required]

1874 This element of **CipherDataType** from [XMLEncryption] contains the encrypted symmetric-key. As
1875 specified in [XMLEncryption], the content of this element is Base-64 encoded and is of the XML
1876 Schema **base64Binary** type.

1878 Some high-level examples of the <Symkey> element are as follows. Details about the <KeyUsePolicy>
1879 element have been elided for brevity:

1880 **Example 1 – A response with a symmetric key:**

```
1881 <ekmi:SymkeyResponse
1882   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1883   <ekmi:Symkey>
1884     <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
1885     <ekmi:KeyUsePolicy>.....</ekmi:KeyUsePolicy>
1886     <ekmi:EncryptionMethod
1887       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
1888     <xenc:CipherData>
1889       <xenc:CipherValue>
1890         E9zWB/y93hVSzeTLiDcQoDxmNlNxTux+SffmNwCJmt1dIqzQHBnpdQ8
1891         1g6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxj
1892         fg1pU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
1893       </xenc:CipherValue>
1894     </xenc:CipherData>
1895   </ekmi:Symkey>
1896 </ekmi:SymkeyResponse>
```

1897 **Example 2 – A response with multiple symmetric keys:**

```
1898 <ekmi:SymkeyResponse
1899   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1900   <ekmi:Symkey>
1901     <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
1902     <ekmi:KeyUsePolicy>.....</ekmi:KeyUsePolicy>
1903     <ekmi:EncryptionMethod
1904       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
1905     <xenc:CipherData>
1906       <xenc:CipherValue>
1907         E9zWB/y93hVSzeTLiDcQoDxmLNxTux+SffMNwCJmt1dIqzQHBnpdQ8
1908         1g6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxj
1909         fg1pU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
1910       </xenc:CipherValue>
1911     </xenc:CipherData>
1912   </ekmi:Symkey>
1913   <ekmi:Symkey>
1914     <ekmi:GlobalKeyID>10514-1-236</ekmi:GlobalKeyID>
1915     <ekmi:KeyUsePolicy>.....</ekmi:KeyUsePolicy>
1916     <ekmi:EncryptionMethod
1917       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
1918     <xenc:CipherData>
1919       <xenc:CipherValue>
1920         Qbg65cy93hVSzeTLiDcQoDxmLNxTux+SffMNwCJmt1dIqzQHBnpdQ8
1921         7k6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxj
1922         uyecU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
1923       </xenc:CipherValue>
1924     </xenc:CipherData>
1925   </ekmi:Symkey>
1926 </ekmi:SymkeyResponse>
```

1927 **4.6 Element <SymkeyError>**

1928 The <SymkeyError> element of type *SymkeyErrorType*, contains the error returned by the SKS server, in
1929 response to a failure in processing of a <SymkeyRequest> from a client application.

1930 **Schema Definition:**

```
1931 <xsd:complexType name="SymkeyErrorType">
1932   <xsd:sequence>
1933     <xsd:element name="RequestedGlobalKeyID" type="ekmi:GlobalKeyIDType"/>
1934     <xsd:element
1935       name="RequestedKeyClass"
1936       type="ekmi:KeyClassType"
1937       minOccurs="0"/>
1938     <xsd:element name="ErrorCode">
1939       <xsd:simpleType>
1940         <xsd:restriction base="xsd:string">
1941           <xsd:maxLength value="255"/>
1942         </xsd:restriction>
1943       </xsd:simpleType>
1944     </xsd:element>
1945     <xsd:element name="ErrorMessage">
1946       <xsd:simpleType>
1947         <xsd:restriction base="xsd:string">
1948           <xsd:maxLength value="1024"/>
1949         </xsd:restriction>
1950       </xsd:simpleType>
1951     </xsd:element>
```

1952 </xsd:sequence>
1953 </xsd:complexType>

1954 When a request for a symmetric key fails despite successfully being processed by the SOAP layer, there MUST
1955 be at least one <SymkeyError> element in a <SymkeyResponse> element. When a <SymkeyRequest> fails
1956 at the SOAP layer, the response SHALL consist of a **SOAPFault**.

1957 There MAY be more than one <SymkeyError> element in the response if the client application made a request
1958 for multiple symmetric keys and the **SKS** server failed in processing the request for more than one symmetric
1959 key.

1960 The <SymkeyError> element consists of a sequence of the following child elements:

1961 1. <RequestedGlobalKeyID> [Required]

1962 This element of type **GlobalKeyIDType** identifies the unique identifier of the symmetric key requested by
1963 the client application. There SHALL be only one <RequestedGlobalKeyID> within a
1964 <SymkeyError> element.
1965

1966 The **GlobalKeyIDType** is specified in Section 4.2.
1967

1968 2. <RequestedKeyClass> [Optional]

1969 This element of type **KeyClassType** identifies the key-class of the symmetric key requested by the
1970 client application. If the <RequestedKeyClass> element is not embedded in the <SymkeyError>
1971 element, this implies that the requested symmetric key was for the default key-class of the SKMS.
1972

1973 The **KeyClassType** is specified in Section 4.3.
1974

1975 3. <ErrorCode> [Required]

1976 This element of type **String** identifies a mnemonic code identifying the error the **SKS** Server
1977 experienced in processing the client's symmetric key request.
1978

1979 The <ErrorCode> element SHALL return one of the codes identified in Appendix D of this specification.
1980

1981 4. <ErrorMessage> [Required]

1982 This element of type **String** identifies a localized message describing the error the **SKS** Server
1983 experienced in processing the client's symmetric key request.
1984

1985 The <ErrorMessage> element SHALL return the appropriate localized version of the message
1986 corresponding to the <ErrorCode> element from Appendix D of this specification.
1987

1988 Some high-level examples of the <SymkeyError> element are as follows.

1989 **Example 1 – An error within a response:**

```
1990 <ekmi:SymkeyResponse  
1991   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1992   <ekmi:SymkeyError>  
1993     <ekmi:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>  
1994     <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>  
1995     <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>  
1996   </ekmi:SymkeyError>  
1997 </ekmi:SymkeyResponse>
```

1998 **Example 2 – Multiple errors within a response:**

```
1999 <ekmi:SymkeyResponse  
2000   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
2001   <ekmi:SymkeyError>  
2002     <ekmi:RequestedGlobalKeyID>10514-1-0</ekmi:RequestedGlobalKeyID>
```

```

2003         <ekmi:ErrorCode>SKS-100001</ekmi:ErrorCode>
2004         <ekmi:ErrorMessage>Invalid GlobalKeyID</ekmi:ErrorMessage>
2005     </ekmi:SymkeyError>
2006     <ekmi:SymkeyError>
2007         <ekmi:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>
2008         <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
2009         <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
2010     </ekmi:SymkeyError>
2011 </ekmi:SymkeyResponse>

```

2012 4.7 Element <KeyUsePolicy>

2013 The <KeyUsePolicy> element defines rules that conforming implementations of the **SKCL** MUST adhere to
2014 when using the symmetric key sent by the **SKS** Server. It is an integral part of the <Symkey> element .

2015 Schema Definition:

```

2016     <xsd:complexType name="KeyUsePolicyType" mixed="true">
2017         <xsd:sequence>
2018             <xsd:element name="KeyUsePolicyID" type="tns:TwoPartIDType"/>
2019             <xsd:element name="PolicyName">
2020                 <xsd:simpleType>
2021                     <xsd:restriction base="xsd:string">
2022                         <xsd:maxLength value="255"/>
2023                     </xsd:restriction>
2024                 </xsd:simpleType>
2025             </xsd:element>
2026             <xsd:element name="KeyClass" type="tns:KeyClassType"/>
2027             <xsd:element name="KeyAlgorithm" type="tns:EncryptionAlgorithmType"/>
2028             <xsd:element name="KeySize" type="tns:KeySizeType"/>
2029             <xsd:element name="Status" type="tns:StatusType"/>
2030             <xsd:element name="Permissions" type="tns:PermissionsType"/>
2031         </xsd:sequence>
2032     </xsd:complexType>

```

2033 The <KeyUsePolicy> element is of the **KeyUsePolicyType** and consists of the following child elements:

- 2034 1. <KeyUsePolicyID> [Required]
2035
2036 The <KeyUsePolicyID> element, of type **TwoPartIDType** , identifies the unique policy object within
2037 the SKMS. There SHALL be only one <KeyUsePolicyID> element within a <KeyUsePolicy>
2038 element.
2039
2040 The **TwoPartIDType** is specified in Section 4.8.
- 2041 2. <PolicyName> [Required]
2042
2043 The <PolicyName> element, of type XSD **String**, with a maximum length of 255 characters,
2044 identifies a unique name given to this <KeyUsePolicy>. There SHALL be only one <PolicyName>
2045 element within a <KeyUsePolicy> element.
- 2046 3. <KeyClass> [Required]
2047
2048 The <KeyClass> element ,of type **KeyClassType** , identifies a key-class assigned to this
2049 <KeyUsePolicy>. There SHALL be only one <KeyUsePolicyID> element within a
2050 <KeyUsePolicy> element.
2051
2052 The **KeyClassType** is specified in Section 4.3.

- 2053 4. <KeyAlgorithm> [Required]
 2054
 2055 The <KeyAlgorithm> element , of type **EncryptionAlgorithmType** , identifies encryption algorithm to
 2056 be used by applications when using this symmetric key. There SHALL be only one <KeyAlgorithm>
 2057 element within a <KeyUsePolicy> element.
 2058
 2059 The <KeyAlgorithm> element is specified in Section 4.9.
- 2060 5. <KeySize> [Required]
 2061
 2062 The <KeySize> element , of type **KeySizeType**, defines the size of the symmetric key, in bits (binary
 2063 digits). There SHALL be only one <KeySize> element within a <KeyUsePolicy> element.
 2064
 2065 Note: It is possible to determine the size of a symmetric key in an **SKCL** implementation without having
 2066 to send the size in the response. So, why include it? It is our belief that while network bandwidth and
 2067 compute performance of devices are increasing steadily, encryption is desired in many small and
 2068 portable devices. Consequently, it will speed up applications in cryptographic processing if they do not
 2069 have to determine the size of each key they use. While “protocol purity” demands that implementation
 2070 issues do not show up in protocol design, we believe it is justified in this case.
 2071
 2072 The **KeySizeType** is specified in Section 4.10.
- 2073 6. <Status> [Required]
 2074
 2075 The <Status> element, of type **StatusType**, identifies the current status of the symmetric key. There
 2076 SHALL be only one <Status> element within a <KeyUsePolicy> element.
 2077
 2078 The **StatusType** is specified in Section 4.11.
- 2079 7. <Permissions> [Required]
 2080
 2081 The <Permissions> element, of type **PermissionsType**, defines what is permissible to client
 2082 applications with the symmetric key this element is associated with. It is the responsibility of the
 2083 conforming **SKCL** implementation to enforce these rules.
 2084
 2085 An important distinction of this element – unlike most access control rules – is that the absence of sub-
 2086 elements in the <Permissions> element implies that all permissions are allowed. The presence of
 2087 sub-elements in this element provide rules to the **SKCL** about what actions are permitted.
 2088
 2089 There SHALL be only one <Permissions> element in a <KeyUsePolicy> element.
 2090
 2091 The **PermissionsType** is specified in Section 4.12.

2092 Some examples of the <KeyUsePolicy> element are as follows.

2093 **Example 1 – A <KeyUsePolicy> with some permission restrictions:**

```

2094 <ekmi:KeyUsePolicy>
2095   <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>
2096   <ekmi:PolicyName>DES-EDE KeyUsePolicy</ekmi:PolicyName>
2097   <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
2098   <ekmi:KeyAlgorithm>
2099     http://www.w3.org/2001/04/xmlenc#tripleDES-cbc
2100   </ekmi:KeyAlgorithm>
2101   <ekmi:KeySize>192</ekmi:KeySize>
2102   <ekmi:Status>Active</ekmi:Status>
2103   <ekmi:Permissions>
2104     <ekmi:PermittedApplications ekmi:any="false">
2105       <ekmi:PermittedApplication>
2106         <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
2107         <ekmi:ApplicationName>Payroll Application</ekmi:ApplicationName>

```



```

2108         <ekmi:Version>1.0</ekmi:Version>
2109         <ekmi:DigestAlgorithm>
2110             http://www.w3.org/2000/09/xmlsig#sha1
2111         </ekmi:DigestAlgorithm>
2112         <ekmi:DigestValue>
2113             229ea73a5e76eabd183663d332b283948a9202a1
2114         </ekmi:DigestValue>
2115     </ekmi:PermittedApplication>
2116 </ekmi:PermittedApplications>
2117 <ekmi:PermittedDates ekmi:any="false">
2118     <ekmi:PermittedDate>
2119         <ekmi:StartDate>2008-01-01</ekmi:StartDate>
2120         <ekmi:EndDate>2008-12-31</ekmi:EndDate>
2121     </ekmi:PermittedDate>
2122 </ekmi:PermittedDates>
2123 <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2124 <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2125 <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2126 <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2127 <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
2128 <ekmi:PermittedTimes ekmi:any="false">
2129     <ekmi:PermittedTime>
2130         <ekmi:StartTime>07:00:00</ekmi:StartTime>
2131         <ekmi:EndTime>19:00:00</ekmi:EndTime>
2132     </ekmi:PermittedTime>
2133 </ekmi:PermittedTimes>
2134 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2135 </ekmi:Permissions>
2136 </ekmi:KeyUsePolicy>

```

2137 **Example 2 – A <KeyUsePolicy> with no restrictions on the key:**

```

2138 <ekmi:KeyUsePolicy>
2139     <ekmi:KeyUsePolicyID>10514-2</ekmi:KeyUsePolicyID>
2140     <ekmi:PolicyName>Laptop KeyUsePolicy</ekmi:PolicyName>
2141     <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
2142     <ekmi:KeyAlgorithm>
2143         http://www.w3.org/2001/04/xmlenc#aes256-cbc
2144     </ekmi:KeyAlgorithm>
2145     <ekmi:KeySize>256</ekmi:KeySize>
2146     <ekmi:Status>Active</ekmi:Status>
2147     <ekmi:Permissions>
2148         <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
2149         <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
2150         <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2151         <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2152         <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2153         <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2154         <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
2155         <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
2156         <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2157     </ekmi:Permissions>
2158 </ekmi:KeyUsePolicy>

```

2159 **4.8 Type TwoPartIDType**

2160 The *TwoPartIDType* is used to create identifiers for many elements within the SKSML. It is a simple
2161 concatenation of two integers with a hyphen between them ("-") to create an XML Schema *String* type.

2162 The *TwoPartIDType* has a minimum length of three (3) characters, and a maximum length of 41 characters.

2163 **Schema Definition:**

```
2164 <xsd:simpleType name="TwoPartIDType">
2165   <xsd:restriction base="xsd:string">
2166     <xsd:minLength value="3"/>
2167     <xsd:maxLength value="41"/>
2168     <xsd:pattern value="[1-9][0-9]{0,19}-[1-9][0-9]{0,19}"/>
2169     <xsd:whiteSpace value="collapse"/>
2170   </xsd:restriction>
2171 </xsd:simpleType>
```

2172 The *TwoPartIDType* is used in the <ApplicationID>, the <KeyCachePolicyID> and the <KeyUsePolicyID>
2173 elements within the SKSML.

2174 Some examples of the <KeyUsePolicy> element are as follows.

2175 **Example 1 – A *TwoPartIDType* used to identify an ApplicationID:**

```
2176 <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
```

2177 **Example 2 – A *TwoPartIDType* used to identify a KeyUsePolicyID:**

```
2178 <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>
```

2179 **Example 3 – A minimum-length *TwoPartIDType* :**

```
2180 <ekmi:KeyCachePolicyID>5-4</ekmi:KeyCachePolicyID>
```

2181 **Example 4 – A maximum-length *TwoPartIDType* :**

```
2182 <ekmi:ApplicationID>
2183   18446744073709551615-18446744073709551615
2184 </ekmi:ApplicationID>
```

2185 4.9 Element <KeyAlgorithm>

2186 The element <KeyAlgorithm> , of type *EncryptionAlgorithmType*, is used to identify the cryptographic
2187 algorithm to be used with the symmetric keys in the <SymkeyResponse>.

2188 **Schema Definition:**

```
2189 <xsd:simpleType name="EncryptionAlgorithmType">
2190   <xsd:restriction base="xsd:anyURI">
2191     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
2192     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
2193     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes192-cbc"/>
2194     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
2195   </xsd:restriction>
2196 </xsd:simpleType>
```

2197 The algorithms currently supported by this specification are the algorithms defined in [XMLEncryption]. As new
2198 algorithms are added to [XMLEncryption], they will be added to the enumerated list in this element. Currently,
2199 the following four algorithms are supported:

2200 1. Triple Data Encryption Standard (3DES)

2201

2202 Within the context of this specification, and as specified in [XMLEncryption], the form of 3DES
2203 supported within SKSML is a 192-bit key with a 64-bit Initialization Vector. Of the key bits, the first 64
2204 are used in the first DES operation, the second 64 bits in the second (middle) DES operation, and the
2205 third 64 bits in the third (last) DES operation. Each of these 64 bits of key contain 56 effective bits and
2206 8 parity bits.

2207 The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#tripleDES-cbc>.

2208

2209 2. Advanced Encryption Standard (AES) – 128-bit

2210

2211 Within the context of this specification, and as specified in [AES], this is a 128-bit symmetric key used in

2212 the Cipher Block Chaining (CBC) mode.

2213

2214 The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#aes128-cbc>.

2215 3. Advanced Encryption Standard (AES) – 192-bit

2216

2217 Within the context of this specification, and as specified in [AES], this is a 192-bit symmetric key used in

2218 the Cipher Block Chaining (CBC) mode.

2219

2220 The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#aes192-cbc>.

2221 4. Advanced Encryption Standard (AES) – 256-bit

2222

2223 Within the context of this specification, and as specified in [AES], this is a 256-bit symmetric key used in

2224 the Cipher Block Chaining (CBC) mode.

2225

2226 The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#aes256-cbc>.

2227 There SHALL be only one <KeyAlgorithm> element within a <KeyUsePolicy> element.

2228 Some examples of the <KeyAlgorithm> element are as follows; other elements of the <KeyUsePolicy>

2229 element are not displayed for brevity:

2230 **Example 1 – An example using the Triple-DES key algorithm:**

```
2231 <ekmi:KeyUsePolicy>
2232   ...
2233   <ekmi:KeyAlgorithm>
2234     http://www.w3.org/2001/04/xmlenc#tripleDES-cbc
2235   </ekmi:KeyAlgorithm>
2236   ...
2237 </ekmi:KeyUsePolicy>
```

2238 **Example 2 – An example using the AES-128 key algorithm:**

```
2239 <ekmi:KeyUsePolicy>
2240   ...
2241   <ekmi:KeyAlgorithm>
2242     http://www.w3.org/2001/04/xmlenc#aes128-cbc
2243   </ekmi:KeyAlgorithm>
2244   ...
2245 </ekmi:KeyUsePolicy>
```

2246 **4.10 Element <KeySize>**

2247 The element <KeySize>, of type *KeySizeType*, is used to identify the size of the symmetric key, in binary digits

2248 (bits) in the <SymkeyResponse>.

2249 **Schema Definition:**

```
2250 <xsd:simpleType name="KeySizeType">
2251   <xsd:restriction base="xsd:unsignedShort">
2252     <xsd:totalDigits value="3"/>
2253     <xsd:fractionDigits value="0"/>
2254     <xsd:enumeration value="128"/>
```

```
2255         <xsd:enumeration value="192"/>
2256         <xsd:enumeration value="256"/>
2257     </xsd:restriction>
2258 </xsd:simpleType>
```

2259 There SHALL be only one <KeySize> element within a <KeyUsePolicy> element.

2260 Currently, the following three key-sizes are supported:

- 2261 1. 128-bits when used with the *AES-192* algorithm
- 2262 2. 192-bits when used with the *AES-192* or the *3DES* algorithms
- 2263 3. 256-bits when used with the *AES-256* algorithm

2264 Some examples of the <KeySize> element are as follows; other elements of the <KeyUsePolicy> element are
2265 not displayed for brevity:

2266 **Example 1 – An example using a 128-bit key size:**

```
2267 <ekmi:KeyUsePolicy>
2268     ...
2269     <ekmi:KeySize>128</ekmi:KeySize>
2270     ...
2271 </ekmi:KeyUsePolicy>
```

2272 **Example 2 – An example using a 192-bit key size:**

```
2273 <ekmi:KeyUsePolicy>
2274     ...
2275     <ekmi:KeySize>192</ekmi:KeySize>
2276     ...
2277 </ekmi:KeyUsePolicy>
```

2278 **Example 3 – An example using a 256-bit key size:**

```
2279 <ekmi:KeyUsePolicy>
2280     ...
2281     <ekmi:KeySize>256</ekmi:KeySize>
2282     ...
2283 </ekmi:KeyUsePolicy>
```

2284 4.11 Element <Status>

2285 The element <Status>, of type *StatusType*, is used to identify the current status of an object . It is used in
2286 almost every element within the SKMS.

2287 **Schema Definition:**

```
2288 <xsd:simpleType name="StatusType">
2289     <xsd:restriction base="xsd:string">
2290         <xsd:enumeration value="Active"/>
2291         <xsd:enumeration value="Default"/>
2292         <xsd:enumeration value="Inactive"/>
2293         <xsd:enumeration value="Other"/>
2294     </xsd:restriction>
2295 </xsd:simpleType>
```

2296 Where it exists within an element, there SHALL be only one <Status> element within the enclosing element.

2297 The <Status> element can contain one of four *String* type values:

- 2298 1. The **Active** value indicates that the element that makes up the document-root is currently active in the
2299 SKMS and conforming **SKCL** implementations may use it within applications.
- 2300 2. The **Default** value indicates that the element that makes up the document root is the default element in
2301 the SKMS, is also active, and conforming **SKCL** implementations may use it within applications.
- 2302 3. The **Inactive** value indicates that the element that makes up the document root is not active in the
2303 SKMS, and conforming **SKCL** implementations may NOT use it within applications.
- 2304 4. The **Other** value indicates that the element that makes up the document root has a meaning that is
2305 application-specific. However, conforming **SKCL** implementations may NOT use it within applications.

2306 Some examples of the <Status> element are shown below; other parts of their enclosing elements are not
2307 shown for brevity:

2308 **Example 1 – An example with an Active status within a <KeyUsePolicy> element:**

```
2309 <ekmi:KeyUsePolicy>
2310   ...
2311   <ekmi:Status>Active</ekmi:Status>
2312   ...
2313 </ekmi:KeyUsePolicy>
```

2314 **Example 2 – An example with an Inactive status within a <KeyUsePolicy> element:**

```
2315 <ekmi:KeyUsePolicy>
2316   ...
2317   <ekmi:Status>Inactive</ekmi:Status>
2318   ...
2319 </ekmi:KeyUsePolicy>
```

2320 **Example 3 – An example with a Default status within a <KeyUsePolicy> element:**

```
2321 <ekmi:KeyUsePolicy>
2322   ...
2323   <ekmi:Status>Default</ekmi:Status>
2324   ...
2325 </ekmi:KeyUsePolicy>
```

2326 4.12 Element <Permissions>

2327 The <Permissions> element, of the type *PermissionsType* is at the heart of the <KeyUsePolicy> element. It
2328 provides guidance to conforming **SKCL** implementations on who may use the symmetric key, when they may use
2329 it, for what purposes, for how long and in which locations. For applications that conform to the Multi-Level
2330 Security (MLS) model, there is a provision for specifying which levels are permitted use of the key. There is also
2331 an element that allows for extending the <Permissions> element to accommodate rules that have not been
2332 envisioned in the current specification.

2333 There SHALL be only one <Permissions> element within a <KeyUsePolicy> element.

2334 **Schema Definition:**

```
2335 <xsd:complexType name="PermissionsType">
2336 <xsd:sequence>
2337   <xsd:element
2338     name="PermittedApplications"
2339     type="tns:PermittedApplicationsType"
2340     minOccurs="1"
2341     nillable="true"/>
2342   <xsd:element
2343     name="PermittedDates"
2344     type="tns:PermittedDatesType"
```

```

2345         minOccurs="1"
2346         nillable="true"/>
2347     <xsd:element
2348         name="PermittedDays"
2349         type="tns:PermittedDaysType"
2350         minOccurs="1"
2351         nillable="true"/>
2352     <xsd:element
2353         name="PermittedDuration"
2354         type="tns:PermittedDurationType"
2355         minOccurs="1"
2356         nillable="true"/>
2357     <xsd:element
2358         name="PermittedLevels"
2359         type="tns:PermittedLevelsType"
2360         minOccurs="1"
2361         nillable="true"/>
2362     <xsd:element
2363         name="PermittedLocations"
2364         type="tns:PermittedLocationsType"
2365         minOccurs="1"
2366         nillable="true"/>
2367     <xsd:element
2368         name="PermittedNumberOfTransactions"
2369         type="tns:PermittedNumberOfTransactionsType"
2370         minOccurs="1"
2371         nillable="true"/>
2372     <xsd:element
2373         name="PermittedTimes"
2374         type="tns:PermittedTimesType"
2375         minOccurs="1"
2376         nillable="true"/>
2377     <xsd:element
2378         name="PermittedUses"
2379         type="tns:PermittedUsesType"
2380         minOccurs="1"
2381         nillable="true"/>
2382     <xsd:element
2383         name="Other"
2384         type="xsd:anyType"
2385         minOccurs="0"/>
2386 </xsd:sequence>
2387 </xsd:complexType>

```

2388
2389 The <Permissions> element consists of the following sub-elements:

2390 1. A required <PermittedApplications> element which identifies applications that are permitted use of
2391 the symmetric key in question. While the <PermittedApplications> element is required, it may be
2392 empty (NULL). The absence of sub-elements in the <PermittedApplications> element implies that
2393 all applications are permitted to use the key. Identifying a specific application restricts the use of the key
2394 to only the identified applications.

2395
2396 The <PermittedApplications> element is specified in Section 4.13.

2397 2. A required <PermittedDates> element which identifies the date ranges during which applications are
2398 permitted use of the symmetric key in question. While the <PermittedDates> element is required, it
2399 may be empty (NULL). The absence of sub-elements in the <PermittedDates> element implies that
2400 applications are permitted to use the key on any date. Identifying specific date ranges restricts the use
2401 of the key to only the duration between the identified dates.

- 2402
2403 The <PermittedDates> element is specified in Section 4.12.
- 2404 3. A required <PermittedDays> element which identifies the days of week during which applications are
2405 permitted use of the symmetric key in question. While the <PermittedDays> element is required, it
2406 may be empty (NULL). The absence of sub-elements in the <PermittedDays> element implies that
2407 applications are permitted to use the key on any day of the week. Identifying specific days restricts the
2408 use of the key to only the identified days.
2409
2410 The <PermittedDays> element is specified in Section 4.15.
- 2411 4. A required <PermittedDuration> element which identifies the duration (in seconds) in which
2412 applications are permitted use of the symmetric key in question *once the SKCL starts using the*
2413 *symmetric key*. While the <PermittedDuration> element is required, it may be empty (NULL). The
2414 absence of any content – the duration time - in the <PermittedDuration> element implies that
2415 applications are permitted to use the key for any duration after it has been used. Identifying a non-zero,
2416 positive duration value restricts the use of the key to only the period after the start of the use of the key.
2417
2418 A distinction between <PermittedDates> and <PermittedDuration> is that the former has fixed
2419 start and end-dates for the use of the key, whereas the latter has a fixed end-date-and-time after the
2420 key has begun to be used without a fixed start-date-and-time. Thus, an application with a
2421 <PermittedDuration> can begin the use of a symmetric key at any time, but must stop its use at the
2422 end of the <PermittedDuration> once it has begun. With <PermittedDates>, an application can
2423 continue using the symmetric key until the fixed date-and-time have been reached.
2424
2425 The <PermittedDuration> element is specified in Section 4.16
- 2426 5. Within a Multi-Level Security (MLS) system, the required <PermittedLevels> element identifies the
2427 security levels at which applications are permitted use of the symmetric key in question. While the
2428 <PermittedLevels> element is required, it may be empty (NULL). The absence of sub-elements in
2429 the <PermittedLevels> element implies that applications are permitted to use the key at any level of
2430 security. Identifying specific MLS level(s) restricts the use of the key to only the identified security
2431 level(s).
2432
2433 The <PermittedLevels> element is specified in Section 4.17
- 2434 6. A required <PermittedLocations> element which identifies physical geographic locations where
2435 applications are permitted use of the symmetric key in question. While the <PermittedLocations>
2436 element is required, it may be empty (NULL). The absence of sub-elements in the
2437 <PermittedLocations> element implies that applications are permitted to use the key at any physical
2438 location. Identifying specific locations restricts the use of the key to only the identified locations.
2439
2440 The <PermittedLocations> element is specified in Section 4.18.
- 2441 7. A required <PermittedNumberOfTransactions> element which identifies the number of encryption
2442 transactions that applications are permitted, with the use of the symmetric key in question. While the
2443 <PermittedNumberOfTransactions> element is required, it may be empty (NULL). The absence of
2444 content – the number of transactions – in the <PermittedNumberOfTransactions> element implies
2445 that applications are permitted to use the key for as many encryption transactions as necessary.
2446 Identifying a specific, non-zero, positive number of transactions in this element restricts the use of the
2447 key to only the limit identified in the element.
2448
2449 The <PermittedNumberOfTransactions> element is specified in Section 4.19.
- 2450 8. A required <PermittedTimes> element which identifies the times of day during which applications are
2451 permitted the use of the symmetric key in question. While the <PermittedTimes> element is required,
2452 it may be empty (NULL). The absence of sub-elements in the <PermittedTimes> element implies that
2453 applications are permitted to use the key at any time of day. Identifying specific times restricts the use
2454 of the key to only the duration of the identified times.
2455
2456 The <PermittedTimes> element is specified in Section 4.20.

2457 9. A required <PermittedUses> element which identifies application-uses that applications are permitted
2458 with the symmetric key in question. While the <PermittedUses> element is required, it may be empty
2459 (NULL). The absence of sub-elements in the <PermittedUses> element implies that applications are
2460 permitted to use the key for any purpose. Identifying specific uses restricts the use of the key to only
2461 the identified uses.

2462 The <PermittedUses> element is specified in Section 4.21.

2464 10. The optional <Other> element allows implementers to specify permissions that cannot be addressed
2465 with the above-mentioned categories, for restricting the use of the symmetric key in question.

2466 While the <Other> element provides flexibility for implementations, the disadvantage of the element is
2467 that it may render a specific implementation incompatible with other SKMS implementations that use the
2468 SKSML standard.

2469 **It is strongly recommended that implementers avoid the use of the <Other> element unless they**
2470 **definitely do not expect to inter-operate with other SKCL implementations. If there is a strong**
2471 **need for capability that does not exist within the current specification of the <Permissions>**
2472 **element, implementers are encouraged to contact the OASIS EKMI TC with their requirements.**
2473

2474 When all sub-elements of the <Permissions> element are empty, there are no restrictions on the use of the
2475 symmetric key other than that the application calling on the SKCL be authorized to access the key in question.
2476 However, when there are elements defined within the sub-elements of the <Permissions> element, conforming
2477 SKCL implementations must comply with all the permission elements, evaluating the most restrictive permissions
2478 first and in decreasing order of restriction, before allowing the use of the key.
2479

2480 For example, if a <Permissions> element specifies that a key may be used on Weekdays, between the hours
2481 of 0900 and 1700 Hours, then a request for a symmetric key on a Saturday at 1105 would deny use of the key in
2482 question, since it violates the more restrictive permission of being allowed for use only on weekdays.

2483 **It should be noted that it is the primary responsibility of a conforming SKCL to enforce the**
2484 **<Permission> elements' rules. The SKS server will generate the key – or return an existing key - when**
2485 **an authorized client with appropriate access requests it. However, it is up to the SKCL implementation to**
2486 **comply with the rules in the <Permissions> element.**

2487 In another example, if a <Permissions> element specifies a <PermittedDuration> of 600 seconds from the
2488 start of use of the key, and there is also present a <PermittedNumberOfTransactions> element with a value
2489 of 10 (encryption transactions), conforming SKCL implementations must evaluate both permissions before each
2490 transaction and determine if they are both within the specified thresholds before using the key. If the 600
2491 seconds expire before the 10 encryption transactions have been completed, or if the 10 encryption transactions
2492 are completed before 600 seconds have expired, conforming SKCL implementations MUST not use the key in
2493 question anymore.

2494 Some examples of the <Permissions> element are as shown below; the enclosing <KeyUsePolicy> element,
2495 <Symkey> element and <SymkeyResponse> elements are not displayed for brevity:

2496 **Example 1 – A <Permissions> element that permits a single application the use of the symmetric key in**
2497 **question, between January 01, 2008 and December 31, 2008 and between the hours of 0700 and 1900.**
2498 **There are, however, no restrictions on what days of the week the key may be used, the locations where it**
2499 **may be used, at which MLS level it may be used (if it applies), the number of data files/transactions that**
2500 **may be encrypted with the key or the uses of the key within that application:**

```
2501 <ekmi:Permissions>  
2502   <ekmi:PermittedApplications ekmi:any="false">  
2503     <ekmi:PermittedApplication>  
2504       <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>  
2505       <ekmi:ApplicationName>Payroll Application</ekmi:ApplicationName>  
2506       <ekmi:Version>1.0</ekmi:Version>  
2507       <ekmi:DigestAlgorithm>  
2508         http://www.w3.org/2000/09/xmldsig#sha1  
2509       </ekmi:DigestAlgorithm>  
2510       <ekmi:DigestValue>
```



```

2511         229ea73a5e76eabd183663d332b283948a9202a1
2512         </ekmi:DigestValue>
2513     </ekmi:PermittedApplication>
2514 </ekmi:PermittedApplications>
2515 <ekmi:PermittedDates ekmi:any="false">
2516     <ekmi:PermittedDate>
2517         <ekmi:StartDate>2008-01-01</ekmi:StartDate>
2518         <ekmi:EndDate>2008-12-31</ekmi:EndDate>
2519     </ekmi:PermittedDate>
2520 </ekmi:PermittedDates>
2521 <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2522 <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2523 <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2524 <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2525 <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
2526 <ekmi:PermittedTimes ekmi:any="false">
2527     <ekmi:PermittedTime>
2528         <ekmi:StartTime>07:00:00</ekmi:StartTime>
2529         <ekmi:EndTime>19:00:00</ekmi:EndTime>
2530     </ekmi:PermittedTime>
2531 </ekmi:PermittedTimes>
2532 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2533 </ekmi:Permissions>

```

2534 **Example 2 – A <Permissions> element that permits two specific applications the use of the symmetric**
2535 **key in question, between January 01, 2009 and January 31, 2009.**

```

2536 <ekmi:Permissions>
2537     <ekmi:PermittedApplications ekmi:any="false">
2538         <ekmi:PermittedApplication>
2539             <ekmi:ApplicationID>10514-24</ekmi:ApplicationID>
2540             <ekmi:ApplicationName>
2541                 Employee Tax Reporting Application
2542             </ekmi:ApplicationName>
2543             <ekmi:Version>3.3</ekmi:Version>
2544             <ekmi:DigestAlgorithm>
2545                 http://www.w3.org/2000/09/xmldsig#sha1
2546             </ekmi:DigestAlgorithm>
2547             <ekmi:DigestValue>
2548                 af96d65a7a2415239c8eb8be1347f704322957a4
2549             </ekmi:DigestValue>
2550         </ekmi:PermittedApplication>
2551     <ekmi:PermittedApplication>
2552         <ekmi:ApplicationID>10514-25</ekmi:ApplicationID>
2553         <ekmi:ApplicationName>
2554             IRS Tax Reporting Application
2555         </ekmi:ApplicationName>
2556         <ekmi:Version>2.1</ekmi:Version>
2557         <ekmi:DigestAlgorithm>
2558             http://www.w3.org/2000/09/xmldsig#sha1
2559         </ekmi:DigestAlgorithm>
2560         <ekmi:DigestValue>
2561             a4f5925185ffe12c1a91ea3de90fc086b34b34b2
2562         </ekmi:DigestValue>
2563     </ekmi:PermittedApplication>
2564 </ekmi:PermittedApplications>
2565 <ekmi:PermittedDates ekmi:any="false">
2566     <ekmi:PermittedDate>
2567         <ekmi:StartDate>2009-01-01</ekmi:StartDate>
2568         <ekmi:EndDate>2009-12-31</ekmi:EndDate>

```

```

2569     </ekmi:PermittedDate>
2570 </ekmi:PermittedDates>
2571 <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2572 <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2573 <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2574 <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2575 <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
2576 <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
2577 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2578 </ekmi:Permissions>

```

2579 **Example 3 – A <Permissions> element that permits all applications the use of the symmetric key in**
2580 **question, for 100 transactions for encrypting credit card numbers.**

```

2581 <ekmi:Permissions>
2582   <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
2583   <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
2584   <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2585   <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2586   <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2587   <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2588   <ekmi:PermittedNumberOfTransactions ekmi:any="false">
2589     100
2590   </ekmi:PermittedNumberOfTransactions>
2591   <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
2592   <ekmi:PermittedUses ekmi:any="false">
2593     <ekmi:PermittedUse>CCN</ekmi:PermittedUse>
2594   </ekmi:PermittedUses>
2595 </ekmi:Permissions>

```

2596 **Example 4 – A <Permissions> element that permits all applications the use of the symmetric key in**
2597 **question, for 600 seconds once the SKCL starts using the key.**

```

2598 <ekmi:Permissions>
2599   <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
2600   <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
2601   <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2602   <ekmi:PermittedDuration ekmi:any="false">600</ekmi:PermittedDuration>
2603   <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2604   <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2605   <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
2606   <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
2607   <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2608 </ekmi:Permissions>

```

2609 **Example 5 – A <Permissions> element that permits a specific application the use of the symmetric key**
2610 **in question, at specific geographic locations only on weekdays between the hours of 0800 and 1700, and**
2611 **only when the application is operating at the Secret security level within an MLS system.**

```

2612 <ekmi:Permissions>
2613   <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
2614   <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
2615   <ekmi:PermittedDays ekmi:any="false">
2616     <ekmi:PermittedDay>Weekday</ekmi:PermittedDay>
2617   </ekmi:PermittedDays>
2618   <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2619   <ekmi:PermittedLevels ekmi:any="false">
2620     <ekmi:PermittedLevel>Secret</ekmi:PermittedLevel>
2621   </ekmi:PermittedLevels>
2622   <ekmi:PermittedLocations ekmi:any="false">
2623     <ekmi:PermittedLocation>

```

```

2624         <ekmi:LocationName>Facility A51</ekmi:LocationName>
2625         <ekmi:Latitude>37.385562</ekmi:Latitude>
2626         <ekmi:Longitude>-121.993387</ekmi:Longitude>
2627     </ekmi:PermittedLocation>
2628     <ekmi:PermittedLocation>
2629         <ekmi:LocationName>Facility DC-VA01</ekmi:LocationName>
2630         <ekmi:Latitude>88.485362</ekmi:Latitude>
2631         <ekmi:Longitude>-21.453648</ekmi:Longitude>
2632     </ekmi:PermittedLocation>
2633 </ekmi:PermittedLocations>
2634 <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
2635 <ekmi:PermittedTimes ekmi:any="false">
2636     <ekmi:PermittedTime>
2637         <ekmi:StartTime>08:00:00</ekmi:StartTime>
2638         <ekmi:EndTime>17:00:00</ekmi:EndTime>
2639     </ekmi:PermittedTime>
2640 </ekmi:PermittedTimes>
2641 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2642 </ekmi:Permissions>

```

2643 **4.13 Element <PermittedApplications> and <PermittedApplication>**

2644 The element <PermittedApplications>, of type *PermittedApplicationsType* and its only child-element
2645 <PermittedApplication> of type *ApplicationsType* are used to define the list of applications that are
2646 permitted to use a symmetric key within a specific <Symkey> element.

2647 **Schema Definition:**

```

2648     <xsd:complexType name="PermittedApplicationsType">
2649         <xsd:sequence>
2650             <xsd:element
2651                 name="PermittedApplication"
2652                 type="tns:ApplicationsType"
2653                 minOccurs="0"
2654                 maxOccurs="unbounded"/>
2655         </xsd:sequence>
2656         <xsd:attribute ref="tns:any" use="required"/>
2657     </xsd:complexType>

```

2658 **Schema Definition:**

```

2659     <xsd:complexType name="ApplicationsType">
2660         <xsd:sequence>
2661             <xsd:element name="ApplicationID" type="tns:TwoPartIDType"/>
2662             <xsd:element name="ApplicationName">
2663                 <xsd:simpleType>
2664                     <xsd:restriction base="xsd:string">
2665                         <xsd:maxLength value="256"/>
2666                         <xsd:whiteSpace value="preserve"/>
2667                     </xsd:restriction>
2668                 </xsd:simpleType>
2669             </xsd:element>
2670             <xsd:element name="Version" minOccurs="0">
2671                 <xsd:simpleType>
2672                     <xsd:restriction base="xsd:string">
2673                         <xsd:maxLength value="32"/>
2674                         <xsd:whiteSpace value="preserve"/>
2675                     </xsd:restriction>
2676                 </xsd:simpleType>
2677             </xsd:element>
2678         </xsd:sequence>

```

```

2679         <xsd:group ref="tns:MessageDigestGroup" minOccurs="0"/>
2680         <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
2681     </xsd:sequence>
2682 </xsd:complexType>

```

2683 **Schema Definition:**

```

2684
2685 <xsd:group name="MessageDigestGroup">
2686     <xsd:sequence>
2687         <xsd:element name="DigestAlgorithm">
2688             <xsd:simpleType>
2689                 <xsd:restriction base="xsd:anyURI">
2690                     <xsd:enumeration value="http://www.w3.org/2000/09/xmlsig#sha1"/>
2691                     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#sha256"/>
2692                     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#sha512"/>
2693                 </xsd:restriction>
2694             </xsd:simpleType>
2695         </xsd:element>
2696         <xsd:element name="DigestValue">
2697             <xsd:simpleType>
2698                 <xsd:restriction base="xsd:base64Binary">
2699                     <xsd:maxLength value="1024"/>
2700                 </xsd:restriction>
2701             </xsd:simpleType>
2702         </xsd:element>
2703     </xsd:sequence>
2704 </xsd:group>

```

2705 **Schema Definition:**

```

2706     <xsd:attribute name="any">
2707         <xsd:simpleType>
2708             <xsd:restriction base="xsd:string">
2709                 <xsd:enumeration value="false"/>
2710                 <xsd:enumeration value="true"/>
2711             </xsd:restriction>
2712         </xsd:simpleType>
2713     </xsd:attribute>

```

2714 There SHALL be only one <PermittedApplications> element within the <Permissions> element.
2715 However, there MAY be an unbounded (unlimited) number of <PermittedApplication> elements within a
2716 <PermittedApplications> element.

2717 The <PermittedApplications> element SHALL have one attribute named "any", that will have a "false" or
2718 "true" value, based on the following:

- 2719 • When the <PermittedApplications> element is null (i.e. it does not have a single
2720 <PermittedApplication> sub-element in it), the value of the "any" attribute SHALL be set to "true"
2721 AND the XML Schema Instance (XSI) "nil" attribute SHALL be set to "true".
- 2722 • When the <PermittedApplications> element is not-null (i.e. it has at least one
2723 <PermittedApplication> sub-element in it), the value of the "any" attribute SHALL be set to "false"
2724 AND the XML Schema Instance (XSI) "nil" attribute SHALL NOT be present.

2725 A null <PermittedApplications> element specifies that ALL applications are permitted use of the symmetric
2726 key, subject to complying with all other permission clauses in the <KeyUsePolicy> element.

2727 The <PermittedApplication> sub-element of type **ApplicationsType**, provides details of the application
2728 which is permitted use of the symmetric key in question. The <PermittedApplication> element consists of
2729 the following sub-elements:

- 2730 1. The <ApplicationID> element identifies the unique identifier assigned to this application within the
 2731 SKMS. It is a **TwoPartIDType** as specified in Section 4.8. There SHALL be only one
 2732 <ApplicationID> element within a <PermittedApplication> element.
- 2733 2. The <ApplicationName> element identifies the name assigned to this application within the SKMS. It
 2734 is an XSD **String** with a maximum length of 256 characters. There SHALL be only one
 2735 <ApplicationName> element within a <PermittedApplication> element.
- 2736 3. An optional <Version> element identifying the version number of this application within the SKMS. It
 2737 is an XSD **String** with a maximum length of 32 characters. There MAY be only one <Version>
 2738 element within a <PermittedApplication> element.
- 2739 4. The <MessageDigestGroup> group which identifies the message digest value of the application's
 2740 binary image, along with the message digest algorithm used to calculate the digest value. The
 2741 <MessageDigestGroup> consists of the following elements:
- 2742 a) The <DigestAlgorithm> element of the XSD type **anyURI**, which supports one of the
 2743 following three digest algorithms:
- 2744 i. <http://www.w3.org/2000/09/xmldsig#sha1>
- 2745 ii. <http://www.w3.org/2001/04/xmlenc#sha256>
- 2746 iii. <http://www.w3.org/2001/04/xmlenc#sha512>
- 2747 b) The <DigestValue> element of the XSD type **base64Binary**.
- 2748 There SHALL be only one <MessageDigestGroup> group within a <PermittedApplication>
 2749 element.
- 2750 5. An optional <Other> element that provides implementers the ability to carry other information about
 2751 the application that may be relevant to their SKMS. Implementers are cautioned that the use of the
 2752 <Other> element may not be supported by other SKCL implementations, and may break
 2753 interoperability between two SKMS implementations. Should there be a strong need for additional
 2754 features in the <PermittedApplication> element, implementers are encouraged to contact the
 2755 OASIS EKMI TC with their requirements.

2756 NOTE: The SKSML specification does not specify how an SKCL implementation will determine the message
 2757 digest of an application that needs to use the symmetric key in question. It is left to the implementers of the
 2758 SKCL to determine the message digest of the calling application using the specified algorithm, and verify that
 2759 the digest values match before the SKCL uses the symmetric key on behalf of the application.

2760 Some examples of the <PermittedApplications> element are shown below; other parts of their enclosing
 2761 elements are not shown for brevity:

2762 **Example 1 – An example of a <PermittedApplications> element with two child**
 2763 **<PermittedApplication> elements with specific version numbers and message digest values:**

```

2764 <ekmi:PermittedApplications ekmi:any="false">
2765   <ekmi:PermittedApplication>
2766     <ekmi:ApplicationID>10514-24</ekmi:ApplicationID>
2767     <ekmi:ApplicationName>Employee Tax Reporting</ekmi:ApplicationName>
2768     <ekmi:Version>3.3</ekmi:Version>
2769     <ekmi:DigestAlgorithm>
2770       http://www.w3.org/2000/09/xmldsig#sha1
2771     </ekmi:DigestAlgorithm>
2772     <ekmi:DigestValue>G4bsdfKkt4cziEqFFu0oBTM81efU=</ekmi:DigestValue>
2773   </ekmi:PermittedApplication>
2774   <ekmi:PermittedApplication>
2775     <ekmi:ApplicationID>10514-25</ekmi:ApplicationID>
2776     <ekmi:ApplicationName>IRS Tax Reporting Application</ekmi:ApplicationName>
2777     <ekmi:Version>2.1</ekmi:Version>
2778     <ekmi:DigestAlgorithm>
  
```

```

2779         http://www.w3.org/2001/04/xmlenc#sha256
2780     </ekmi:DigestAlgorithm>
2781     <ekmi:DigestValue>
2782         ab7b85c9410d48c54fc7939c391be4028e7305085191c56e7b3740f2cbdbbc79
2783     </ekmi:DigestValue>
2784 </ekmi:PermittedApplication>
2785 </ekmi:PermittedApplications>

```

2786 **Example 2 – An example of a <PermittedApplications> element with one child**
2787 **<PermittedApplication> element that applies to all versions of the application; the message digest**
2788 **value and algorithm are not used in this example:**

```

2789     <ekmi:PermittedApplications ekmi:any="false">
2790         <ekmi:PermittedApplication>
2791             <ekmi:ApplicationID>10514-14</ekmi:ApplicationID>
2792             <ekmi:ApplicationName>E-Commerce Payment</ekmi:ApplicationName>
2793         </ekmi:PermittedApplication>
2794     </ekmi:PermittedApplications>

```

2795 **Example 3 – An example of a null <PermittedApplications> element specifying that ALL applications**
2796 **are permitted the use of the symmetric key:**

```

2797     <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true" />

```

2798 4.14 Element <PermittedDates> and <PermittedDate>

2799 The element <PermittedDates>, of type *PermittedDatesType* and its only child-element <PermittedDate>,
2800 which is an anonymous XSD *ComplexType*, are used to define ranges of dates between which applications are
2801 permitted to use the symmetric key within a specific <Symkey> element.

2802 Schema Definition:

```

2803     <xsd:complexType name="PermittedDatesType">
2804         <xsd:sequence>
2805             <xsd:element name="PermittedDate" minOccurs="0" maxOccurs="unbounded">
2806                 <xsd:complexType>
2807                     <xsd:sequence>
2808                         <xsd:element name="StartDate">
2809                             <xsd:simpleType>
2810                                 <xsd:restriction base="xsd:date">
2811                                     <xsd:pattern value="\p{Nd}{4}-\p{Nd}{2}-\p{Nd}{2}" />
2812                                 </xsd:restriction>
2813                             </xsd:simpleType>
2814                         </xsd:element>
2815                         <xsd:element name="EndDate">
2816                             <xsd:simpleType>
2817                                 <xsd:restriction base="xsd:date">
2818                                     <xsd:pattern value="\p{Nd}{4}-\p{Nd}{2}-\p{Nd}{2}" />
2819                                 </xsd:restriction>
2820                             </xsd:simpleType>
2821                         </xsd:element>
2822                     </xsd:sequence>
2823                 </xsd:complexType>
2824             </xsd:element>
2825         </xsd:sequence>
2826         <xsd:attribute ref="tns:any" use="required" />
2827     </xsd:complexType>

```

2828 There SHALL be only one <PermittedDates> element within the <Permissions> element. However, there
2829 MAY be an unbounded number of <PermittedDate> elements within a <PermittedDates> element.

2830 The <PermittedDates> element SHALL have one attribute named "any", that will have a "false" or "true" value,
2831 based on the following:

2832 • When the <PermittedDates> element is null (i.e. it does not have a single <PermittedDate> sub-
2833 element in it), the value of the "any" attribute SHALL be set to "true" AND the XML Schema Instance
2834 (XSI) "nil" attribute SHALL be set to "true".

2835 • When the <PermittedDates> element is not-null (i.e. it has at least one <PermittedDate> sub-
2836 element in it), the value of the "any" attribute SHALL be set to "false" AND the XML Schema Instance
2837 (XSI) "nil" attribute SHALL NOT be present.

2838 A null <PermittedDates> element specifies that applications are permitted use of the symmetric key on any
2839 calendar date of the year, subject to complying with all other permission clauses in the <Permissions>
2840 element.

2841 The <PermittedDate> sub-element identifies an individual set of dates between which application are
2842 permitted use of the symmetric key in question. The <PermittedDate> element consists of the following sub-
2843 elements:

2844 1. The <StartDate> element identifies the date from which applications MAY start using the symmetric
2845 key in question. It is an XSD **Date** type that MUST be specified in a specific pattern (see examples)
2846 where the first four digits specify the year, the second two digits specify the calendar month in the year,
2847 and the last two digits specify the calendar date of the month.

2848 There SHALL be only one <StartDate> element within a <PermittedDate> element.

2849 Conforming SKCL implementations SHALL NOT start using the symmetric before the onset of the
2850 <StartDate> on the client machine. Unless further constrained by the <PermittedTimes> element,
2851 the onset of the <StartDate> is specified to be the first second of the day – 00:00:01 Hours – on the
2852 client machine.

2853 2. The <EndDate> element identifies the date until which applications may use the symmetric key in
2854 question. It is an XSD **Date** type that MUST be specified in a specific pattern (see examples) where the
2855 first four digits specify the year, the second two digits specify the calendar month in the year, and the
2856 last two digits specify the calendar date of the month.

2857 There SHALL be only one <EndDate> element within a <PermittedDate> element.

2858 Conforming SKCL implementations SHALL NOT use the symmetric after the end of the <EndDate> on
2859 the client machine. Unless further constrained by the <PermittedTimes> element, the end of the
2860 <EndDate> is specified to be the last second of the day – 23:59:59 Hours – on the client machine.

2861 Examples of the <PermittedDates> element are shown below; other required parts of their enclosing elements
2862 are not shown for brevity:

2863 **Example 1 – An example of a <PermittedDates> element with a single <PermittedDate> element. The**
2864 **<StartDate> specifies January 01, 2009 while the <EndDate> specifies January 31, 2009:**

```
2869 <ekmi:PermittedDates ekmi:any="false">  
2870 <ekmi:PermittedDate>  
2871 <ekmi:StartDate>2009-01-01</ekmi:StartDate>  
2872 <ekmi:EndDate>2009-01-31</ekmi:EndDate>  
2873 </ekmi:PermittedDate>  
2874 </ekmi:PermittedDates>
```

2875 **Example 2 – An example of a <PermittedDates> element with two <PermittedDate> elements. For the**
2876 **first <PermittedDate> element, the <StartDate> element specifies July 01, 2008 while the <EndDate>**
2877 **element specifies July 03, 2008. For the second <PermittedDate> element, the <StartDate> element**
2878 **specifies July 07, 2008 while the <EndDate> element specifies July 12, 2008. This policy would restrict a**
2879 **symmetric key with this <PermittedDates> element so it cannot be used on the July 4th weekend of**
2880 **2008:**


```

2881     <ekmi:PermittedDates ekmi:any="false">
2882         <ekmi:PermittedDate>
2883             <ekmi:StartDate>2008-07-01</ekmi:StartDate>
2884             <ekmi:EndDate>2008-07-03</ekmi:EndDate>
2885         </ekmi:PermittedDate>
2886         <ekmi:PermittedDate>
2887             <ekmi:StartDate>2008-07-07</ekmi:StartDate>
2888             <ekmi:EndDate>2009-07-12</ekmi:EndDate>
2889         </ekmi:PermittedDate>
2890     </ekmi:PermittedDates>

```

2891 **Example 3 – An example of a null <PermittedDates> element, specifying that applications are permitted**
2892 **use of the symmetric key on any date:**

```

2893     <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>

```

2894 4.15 Element <PermittedDays> and <PermittedDay>

2895 The element <PermittedDays> , of the type *PermittedDaysType* and its only child-element <PermittedDay>
2896 of the *PermittedDayType*, are used to define days of the week on which applications are permitted to use a
2897 symmetric key within a specific <Symkey> element.

2898 Schema Definition:

```

2899     <xsd:complexType name="PermittedDaysType">
2900         <xsd:sequence>
2901             <xsd:element
2902                 name="PermittedDay"
2903                 type="tns:PermittedDayType"
2904                 minOccurs="0"
2905                 maxOccurs="unbounded">
2906             </xsd:element>
2907         </xsd:sequence>
2908         <xsd:attribute ref="tns:any" use="required"/>
2909     </xsd:complexType>

```

2910 Schema Definition:

```

2911     <xsd:simpleType name="PermittedDayType">
2912         <xsd:restriction base="xsd:string">
2913             <xsd:enumeration value="Sunday"/>
2914             <xsd:enumeration value="Monday"/>
2915             <xsd:enumeration value="Tuesday"/>
2916             <xsd:enumeration value="Wednesday"/>
2917             <xsd:enumeration value="Thursday"/>
2918             <xsd:enumeration value="Friday"/>
2919             <xsd:enumeration value="Saturday"/>
2920             <xsd:enumeration value="Weekday"/>
2921             <xsd:enumeration value="Weekend"/>
2922         </xsd:restriction>
2923     </xsd:simpleType>

```

2924 There SHALL be only one <PermittedDays> element within the <Permissions> element. However, there
2925 MAY be an unbounded (unlimited) number of <PermittedDay> elements within a <PermittedDays> element.

2926 The <PermittedDays> element SHALL have one attribute named “any”, that will have a “false” or “true” value,
2927 based on the following:

- 2928 • When the <PermittedDays> element is null (i.e. it does not have a single <PermittedDay> sub-
2929 element in it), the value of the “any” attribute SHALL be set to “true” AND the XML Schema Instance
2930 (XSI) “nil” attribute SHALL be set to “true”.

- 2931 • When the <PermittedDays> element is not-null (i.e. it has at least one <PermittedDay> sub-
2932 element in it), the value of the "any" attribute SHALL be set to "false" AND the XML Schema Instance
2933 (XSI) "nil" attribute SHALL NOT be present.

2934 A null <PermittedDays> element specifies that applications are permitted use of the symmetric key on all days
2935 of the week, subject to complying with all other permission clauses in the <Permissions> element.

2936 The <PermittedDay> element, of the XSD *String* type, identifies individual days of the week from an
2937 enumerated list on which application are permitted to use the symmetric key in question.

2938 Examples of the <PermittedDays> element are shown below; other parts of their enclosing elements are not
2939 shown for brevity:

2940 **Example 1 – An example of a <PermittedDays> element with a single<PermittedDay> child element,**
2941 **specifying that the symmetric key may be used only on weekdays:**

```
2942 <ekmi:PermittedDays ekmi:any="false">  
2943 <ekmi:PermittedDay>Weekday</ekmi:PermittedDay>  
2944 </ekmi:PermittedDays>
```

2945 **Example 2 – An example of a <PermittedDays> element with three <PermittedDay> child elements,**
2946 **specifying that the symmetric key may be used only on Mondays, Wednesdays and Fridays:**

```
2947 <ekmi:PermittedDays ekmi:any="false">  
2948 <ekmi:PermittedDay>Monday</ekmi:PermittedDay>  
2949 <ekmi:PermittedDay>Wednesday</ekmi:PermittedDay>  
2950 <ekmi:PermittedDay>Friday</ekmi:PermittedDay>  
2951 </ekmi:PermittedDays>
```

2952 **Example 3 – An example of a null <PermittedDays> element, specifying that the symmetric key may be**
2953 **used on any day of the week:**

```
2954 <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
```

2955 4.16 Element <PermittedDuration>

2956 The element <PermittedDuration>, of the type *PermittedDurationType* is used to define the number of
2957 seconds, applications are permitted to use a symmetric key, once the SKCL has started using the symmetric key
2958 in question.

2959 Schema Definition:

```
2960 <xsd:complexType name="PermittedDurationType">  
2961 <xsd:simpleContent>  
2962 <xsd:extension base="tns:DurationType">  
2963 <xsd:attribute ref="tns:any" use="required"/>  
2964 </xsd:extension>  
2965 </xsd:simpleContent>  
2966 </xsd:complexType>
```

2967 Schema Definition:

```
2968 <xsd:simpleType name="DurationType">  
2969 <xsd:restriction base="xsd:positiveInteger">  
2970 <xsd:minInclusive value="1"/>  
2971 <xsd:maxInclusive value="18446744073709551615"/>  
2972 </xsd:restriction>  
2973 </xsd:simpleType>
```

2974 There SHALL be only one <PermittedDuration> element within the <Permissions> element.

2975 The <PermittedDuration> element SHALL have one attribute, named “any” that will have a “false” or “true”
2976 value, based on the following:

- 2977 • When the <PermittedDuration> element is null (i.e. it does not have any content in it), the value of
2978 the “any” attribute SHALL be set to “true” AND the XML Schema Instance (XSI) “nil” attribute SHALL be
2979 set to “true”.
- 2980 • When the <PermittedDuration> element is not-null (i.e. it has content in it), the value of the “any”
2981 attribute SHALL be set to “false” AND the XML Schema Instance (XSI) “nil” attribute SHALL NOT be
2982 present.

2983 A null <PermittedDuration> element specifies that applications are permitted use of the symmetric key
2984 indefinitely, subject to complying with all other permission clauses in the <Permissions> element.

2985 The <PermittedDuration> element, of the XSD *positiveInteger* type, identifies the number of seconds for
2986 which the symmetric key in question may be used, ONCE the key has been used by conforming SKCL
2987 implementations for the first time. The values for <PermittedDuration> may range between 1 and
2988 18446744073709551615.

2989 As long as the symmetric has not been used by an SKCL on a client device (it might be cached for many
2990 days/weeks/months depending on the <KeyCachePolicy> in effect within the SKMS for that device) the
2991 effective lifetime of the symmetric key may be well past the number of seconds specified in
2992 <PermittedDuration> when calculated from the time of the key's generation time on the SKS server. It is the
2993 responsibility of the SKCL implementation, when presented with a <PermittedDuration> element in a
2994 <KeyUsePolicy> of a symmetric key, to keep track of the date/time when the symmetric key in question was
2995 first used on the client device, and how long the key will last after that.

2996 Examples of the <PermittedDuration> element are shown below; other parts of their enclosing elements are
2997 not shown for brevity:

2998 **Example 1 – An example of a <PermittedDuration> element specifying that the symmetric key may be
2999 used only for a single 24-hour period from the moment it is first used by an SKCL:**

```
3000 <ekmi:PermittedDuration ekmi:any="false">86400</ekmi:PermittedDuration>
```

3001 **Example 2 – An example of a <PermittedDuration> element specifying that the symmetric key may be
3002 used only for week from the moment it is first used by an SKCL:**

```
3003 <ekmi:PermittedDuration ekmi:any="false">604800</ekmi:PermittedDuration>
```

3004 **Example 3 – An example of a <PermittedDuration> element specifying that the symmetric key may be
3005 used only 5 minutes from the moment it is first used by an SKCL:**

```
3006 <ekmi:PermittedDuration ekmi:any="false">300</ekmi:PermittedDuration>
```

3007 **Example 4 – An example of a null <PermittedDuration> element specifying that the symmetric key may
3008 be used indefinitely by an SKCL:**

```
3009 <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
```

3010 4.17 Element <PermittedLevels> and <PermittedLevel>

3011 The element <PermittedLevels>, of the type *LevelClassificationType*, is used to define the security level at
3012 which applications are permitted use of a symmetric key. This element is useful only to applications and systems
3013 that conform to the multi-level security (MLS) system as defined in the Bell-LaPadula model.

3014 Schema Definition:

```
3015 <xsd:complexType name="PermittedLevelsType">  
3016 <xsd:sequence>  
3017 <xsd:element  
3018 name="PermittedLevel"  
3019 type="tns:LevelClassificationType">
```

```

3020         minOccurs="0"
3021         maxOccurs="unbounded">
3022     </xsd:element>
3023     <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
3024 </xsd:sequence>
3025 <xsd:attribute ref="tns:any" use="required"/>
3026 </xsd:complexType>

```

3027 **Schema Definition:**

```

3028 <xsd:simpleType name="LevelClassificationType">
3029     <xsd:restriction base="xsd:string">
3030         <xsd:enumeration value="Unclassified"/>
3031         <xsd:enumeration value="Confidential"/>
3032         <xsd:enumeration value="Secret"/>
3033         <xsd:enumeration value="Top-Secret"/>
3034     </xsd:restriction>
3035 </xsd:simpleType>

```

3036 There SHALL be only one <PermittedLevels> element within the <Permissions> element. However, there
3037 MAY be an unbounded (unlimited) number of <PermittedLevel> elements within the <PermittedLevels>
3038 element. (Practically, it makes no sense to have more than the known levels; however, this specification leaves
3039 itself open to the possibility that other levels may be defined).

3040 The <PermittedLevels> element SHALL have one attribute named "any", that will have a "false" or "true"
3041 value, based on the following:

- 3042 • When the <PermittedLevels> element is null (i.e. it does not have a single <PermittedLevel> sub-
3043 element in it), the value of the "any" attribute SHALL be set to "true" AND the XML Schema Instance
3044 (XSI) "nil" attribute SHALL be set to "true".
- 3045 • When the <PermittedLevels> element is not-null (i.e. it has at least one <PermittedLevel> sub-
3046 element in it), the value of the "any" attribute SHALL be set to "false" AND the XML Schema Instance
3047 (XSI) "nil" attribute SHALL NOT be present.

3048 A null <PermittedLevels> element specifies that applications at ANY level are permitted use of the symmetric
3049 key, subject to complying with all other permission clauses in the <Permissions> element.

3050 The <PermittedLevel> sub-element, of the *LevelClassificationType*, identifies the precise MLS level at
3051 which the symmetric key in question may be used. The <PermittedLevel> SHALL contain one of the following
3052 four (4) enumerated values:

- 3053 1. Unclassified
- 3054 2. Confidential
- 3055 3. Secret
- 3056 4. Top-Secret

3057 Examples of the <PermittedLevels> element are shown below; other parts of their enclosing elements are
3058 not shown for brevity:

3059 **Example 1 – An example of a <PermittedLevels> element specifying that the symmetric key may be
3060 used only by applications at the Confidential level:**

```

3061 <ekmi:PermittedLevels ekmi:any="false">
3062     <ekmi:PermittedLevel>Confidential</ekmi:PermittedLevel>
3063 </ekmi:PermittedLevels>

```

3064 **Example 2 – An example of a <PermittedLevels> element specifying that the symmetric key may be
3065 used only by applications at the Secret or Top-Secret level:**

```

3066     <ekmi:PermittedLevels ekmi:any="false">
3067         <ekmi:PermittedLevel>Secret</ekmi:PermittedLevel>
3068         <ekmi:PermittedLevel>Top-Secret</ekmi:PermittedLevel>
3069     </ekmi:PermittedLevels>

```

3070 **Example 3 – An example of a null <PermittedLevels> element specifying that the symmetric key may**
3071 **be used at any level:**

```

3072     <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>

```

3073 4.18 Element <PermittedLocations> and <PermittedLocation>

3074 The element <PermittedLocations>, of the type *PermittedLocationsType*, is used to define the
3075 geographically physical locations where applications are permitted use of a symmetric key. This element is
3076 useful only to applications that have the ability to determine the Global Positioning System (GPS) location of the
3077 client device intending to use the symmetric key.

3078 Schema Definition:

```

3079     <xsd:complexType name="PermittedLocationsType">
3080         <xsd:sequence>
3081             <xsd:element name="PermittedLocation" minOccurs="1" maxOccurs="unbounded">
3082                 <xsd:complexType>
3083                     <xsd:sequence>
3084                         <xsd:element name="LocationName">
3085                             <xsd:simpleType>
3086                                 <xsd:restriction base="xsd:string">
3087                                     <xsd:maxLength value="256"/>
3088                                     <xsd:whiteSpace value="preserve"/>
3089                                 </xsd:restriction>
3090                             </xsd:simpleType>
3091                         </xsd:element>
3092                         <xsd:group
3093                             ref="tns:LocationCoordinateGroup"
3094                             minOccurs="0"
3095                             maxOccurs="unbounded"/>
3096                         <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
3097                     </xsd:sequence>
3098                 </xsd:complexType>
3099             </xsd:element>
3100         </xsd:sequence>
3101     <xsd:attribute ref="tns:any" use="required"/>
3102 </xsd:complexType>

```

3103 Schema Definition:

```

3104     <xsd:group name="LocationCoordinateGroup">
3105         <xsd:sequence>
3106             <xsd:element name="Latitude">
3107                 <xsd:simpleType>
3108                     <xsd:restriction base="xsd:decimal">
3109                         <xsd:totalDigits value="10"/>
3110                         <xsd:fractionDigits value="7"/>
3111                     </xsd:restriction>
3112                 </xsd:simpleType>
3113             </xsd:element>
3114             <xsd:element name="Longitude">
3115                 <xsd:simpleType>
3116                     <xsd:restriction base="xsd:decimal">
3117                         <xsd:totalDigits value="10"/>

```

```

3118         <xsd:fractionDigits value="7"/>
3119     </xsd:restriction>
3120 </xsd:simpleType>
3121 </xsd:element>
3122 </xsd:sequence>
3123 </xsd:group>

```

3124 There SHALL be only one <PermittedLocations> element within the <Permissions> element. However,
3125 there MAY be an unbounded (unlimited) number of <PermittedLocation> sub-elements within the
3126 <PermittedLocations> element.

3127 The <PermittedLocations> element SHALL have one attribute named "any", that will have a "false" or "true"
3128 value, based on the following:

- 3129 • When the <PermittedLocations> element is null (i.e. it does not have a single
3130 <PermittedLocation> sub-element in it), the value of the "any" attribute SHALL be set to "true" AND
3131 the XML Schema Instance (XSI) "nil" attribute SHALL be set to "true".
- 3132 • When the <PermittedLocations> element is not-null (i.e. it has at least one
3133 <PermittedLocation> sub-element in it), the value of the "any" attribute SHALL be set to "false"
3134 AND the XML Schema Instance (XSI) "nil" attribute SHALL NOT be present.

3135 A null <PermittedLocations> element specifies that applications are permitted use of the symmetric key at
3136 ANY physical location, subject to complying with all other permission clauses in the <Permissions> element.

3137 The <PermittedLocation> element, of the **PermittedLocationType**, identifies the precise geographical
3138 location where the symmetric key in question may be used. The <PermittedLocation> SHALL contain the
3139 following elements:

- 3140 1. The <LocationName> element identifies a human-readable name of the physical location. It is an
3141 XSD **String** type element, with a maximum length of 256 characters.

3142 There SHALL be only one <LocationName> element within a <PermittedLocation> element.
3143

- 3144 2. An optional **LocationCoordinateGroup** which, when present, SHALL contain the following two
3145 elements:

- a) 3146 The <Latitude> element of XSD **Decimal** type, that identifies the horizontal coordinate
3147 location of the client device on the Earth, measured in *degrees* and expressed as a decimal
3148 with the *minutes* and *seconds* part of the measurement expressed as a single fraction.

3149 When used, there SHALL be only one <Latitude> element within the
3150 <PermittedLocation> element.
3151

- b) 3152 The <Longitude> element of XSD **Decimal** type, that identifies the vertical coordinate location
3153 of the client device on the Earth, measured in *degrees* and expressed as a decimal with the
3154 *minutes* and *seconds* part of the measurement expressed as a single fraction.

3155 When used, there SHALL be only one <Longitude> element within the
3156 <PermittedLocation> element.
3157

3158 Some examples of the <PermittedLocations> element are shown below; other parts of their enclosing
3159 elements are not shown for brevity:

3160 **Example 1 – An example of a <PermittedLocations> element specifying that the symmetric key may be**
3161 **used only by applications at a single named location:**

```

3162     <ekmi:PermittedLocations ekmi:any="false">
3163         <ekmi:PermittedLocation>
3164             <ekmi:LocationName>StrongAuth Server Room</ekmi:LocationName>
3165         </ekmi:PermittedLocation>
3166     </ekmi:PermittedLocations>

```

3167 **Example 2 – An example of a <PermittedLocations> element specifying that the symmetric key may be**
3168 **used only by applications at a single location at the given GPS coordinates:**

```
3169     <ekmi:PermittedLocations ekmi:any="false">
3170         <ekmi:PermittedLocation>
3171             <ekmi:LocationName>StrongAuth Server Room</ekmi:LocationName>
3172             <ekmi:Latitude>37.385653 </ekmi:Latitude>
3173             <ekmi:Longitude>-121.993192 </ekmi:Longitude>
3174         </ekmi:PermittedLocation>
3175     </ekmi:PermittedLocations>
```

3176 **Example 3 – An example of a <PermittedLocations> element specifying that the symmetric key may be**
3177 **used only by applications at multiple locations:**

```
3178     <ekmi:PermittedLocations ekmi:any="false">
3179         <ekmi:PermittedLocation>
3180             <ekmi:LocationName>Humongous Headquarters</ekmi:LocationName>
3181         </ekmi:PermittedLocation>
3182         <ekmi:PermittedLocation>
3183             <ekmi:LocationName> Humongous Primary Data Center</ekmi:LocationName>
3184             <ekmi:Latitude>37.385653 </ekmi:Latitude>
3185             <ekmi:Longitude>-121.993192 </ekmi:Longitude>
3186         </ekmi:PermittedLocation>
3187         <ekmi:PermittedLocation>
3188             <ekmi:LocationName>Humongous DR Data Center</ekmi:LocationName>
3189             <ekmi:Latitude>68.845901 </ekmi:Latitude>
3190             <ekmi:Longitude>11.393385 </ekmi:Longitude>
3191         </ekmi:PermittedLocation>
3192     </ekmi:PermittedLocations>
```

3193 **Example 4 – An example of a null <PermittedLocations> element specifying that the symmetric key**
3194 **may be used at any location on the planet:**

```
3195     <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
```

3196 **4.19 Element <PermittedNumberOfTransactions>**

3197 The element <PermittedNumberOfTransactions>, of type *PermittedNumberOfTransactionsType* is used
3198 to define the number of *encryption* transactions that applications are permitted with a symmetric key within a
3199 specific <Symkey> element, once the SKCL has started using the symmetric key in question. It does not limit
3200 the number of *decryption* transactions with the same symmetric key.

3201 **Schema Definition:**

```
3202     <xsd:complexType name="PermittedNumberOfTransactionsType">
3203         <xsd:simpleContent>
3204             <xsd:extension base="tns:NumberOfTransactionsType">
3205                 <xsd:attribute ref="tns:any" use="required"/>
3206             </xsd:extension>
3207         </xsd:simpleContent>
3208     </xsd:complexType>
```

3209 **Schema Definition:**

```
3210     <xsd:simpleType name="NumberOfTransactionsType">
3211         <xsd:restriction base="xsd:positiveInteger">
3212             <xsd:minInclusive value="1"/>
3213             <xsd:maxInclusive value="18446744073709551615"/>
3214         </xsd:restriction>
3215     </xsd:simpleType>
```

3216 There SHALL be only one <PermittedNumberOfTransactions> element within the <Permissions>
 3217 element.

3218 The <PermittedNumberOfTransactions> element SHALL have one attribute named "any", that will have a
 3219 "false" or "true" value, based on the following:

- 3220 • When the <PermittedNumberOfTransactions> element is null (i.e. it does not have any content in
 3221 it), the value of the "any" attribute SHALL be set to "true" AND the XML Schema Instance (XSI) "nil"
 3222 attribute SHALL be set to "true".
- 3223 • When the <PermittedNumberOfTransactions> element is not-null (i.e. it has a positive integer
 3224 content in it), the value of the "any" attribute SHALL be set to "false" AND the XML Schema Instance
 3225 (XSI) "nil" attribute SHALL NOT be present.

3226 A null <PermittedNumberOfTransactions> element specifies that applications are permitted use of the
 3227 symmetric key for an unlimited number of encryption transactions, subject to complying with all other permission
 3228 clauses in the <Permissions> element.

3229 The value of <PermittedNumberOfTransactions> element, of the XSD *positiveInteger* type, MAY range
 3230 between 1 and 18446744073709551615.

3231 Some examples of the <PermittedNumberOfTransactions> element are shown below; other parts of their
 3232 enclosing elements are not shown for brevity:

3233 **Example 1 – An example of a <PermittedNumberOfTransactions> element specifying that the**
 3234 **symmetric key may be used only for a single encryption transaction by an SKCL:**

```
3235 <ekmi:PermittedNumberOfTransactions ekmi:any="false">
3236   1
3237 </ekmi:PermittedNumberOfTransactions>
```

3238 **Example 2 – An example of a <PermittedNumberOfTransactions> element specifying that the**
 3239 **symmetric key may be used only for 100 transactions by an SKCL:**

```
3240 <ekmi:PermittedNumberOfTransactions ekmi:any="false">
3241   100
3242 </ekmi:PermittedNumberOfTransactions>
```

3243 **Example 3 – An example of a null <PermittedNumberOfTransactions> element specifying that the**
 3244 **symmetric key may be used for an unlimited number of encryption transactions by an SKCL:**

```
3245 <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
```

3246 4.20 Element <PermittedTimes> and <PermittedTime>

3247 The element <PermittedTimes>, of the type *PermittedTimesType* and its only child-element
 3248 <PermittedTime>, which is an anonymous XSD *ComplexType*, are used to define sets of times during the day
 3249 between which applications are permitted to use a symmetric key within a specific <Symkey> element.

3250 Schema Definition:

```
3251 <xsd:complexType name="PermittedTimesType">
3252   <xsd:sequence>
3253     <xsd:element name="PermittedTime" minOccurs="0" maxOccurs="unbounded">
3254       <xsd:complexType>
3255         <xsd:sequence>
3256           <xsd:element name="StartTime">
3257             <xsd:simpleType>
3258               <xsd:restriction base="xsd:time">
3259                 <xsd:pattern value="\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}"/>
3260               </xsd:restriction>
3261             </xsd:simpleType>
```



```

3262         </xsd:element>
3263         <xsd:element name="EndTime">
3264             <xsd:simpleType>
3265                 <xsd:restriction base="xsd:time">
3266                     <xsd:pattern value="\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}"/>
3267                 </xsd:restriction>
3268             </xsd:simpleType>
3269         </xsd:element>
3270     </xsd:sequence>
3271 </xsd:complexType>
3272 </xsd:element>
3273 </xsd:sequence>
3274 <xsd:attribute ref="tns:any" use="required"/>
3275 </xsd:complexType>

```

3276 There SHALL be only one <PermittedTimes> element within the <Permissions> element. However, there
3277 MAY be an unbounded (unlimited) number of <PermittedTime> sub-elements within a <PermittedTimes>
3278 element.

3279 The <PermittedTimes> element SHALL have one attribute named "any", that will have a "false" or "true" value,
3280 based on the following:

- 3281 • When the <PermittedTimes> element is null (i.e. it does not have a single <PermittedTime> sub-
3282 element in it), the value of the "any" attribute SHALL be set to "true" AND the XML Schema Instance
3283 (XSI) "nil" attribute SHALL be set to "true".
- 3284 • When the <PermittedTimes> element is not-null (i.e. it has at least one <PermittedTime> sub-
3285 element in it), the value of the "any" attribute SHALL be set to "false" AND the XML Schema Instance
3286 (XSI) "nil" attribute SHALL NOT be present.

3287 A null <PermittedTimes> element specifies that applications are permitted use of the symmetric key at ANY
3288 time of the day or night, subject to complying with all other permission clauses in the <Permissions> element.

3289 The <PermittedTime> sub-element identifies an individual set of times between which application are
3290 permitted to use the symmetric key in question. The <PermittedTime> element consists of the following sub-
3291 elements:

- 3292 1. The <StartTime> element identifies the date from which applications may start using the symmetric
3293 key in question. It is an XSD *Time* type that MUST be specified in a specific pattern (see examples)
3294 where the first two digits specify the hour, the second two digits specify the minutes and the last two
3295 digits specify the seconds in a 24 hour format.

3296 There SHALL be only one <StartTime> element within a <PermittedTime> element.

3297 Conforming **SKCL** implementations SHALL NOT start using the symmetric before the onset of the
3298 <StartTime> on the client machine.

- 3301 2. The <EndTime> element identifies the time until which applications may use the symmetric key in
3302 question. It is an XSD *Time* type that MUST be specified in a specific pattern (see examples) where the
3303 first two digits specify the hour, the second two digits specify the minutes and the last two digits specify
3304 the seconds in a 24 hour format.

3305 There SHALL be only one <EndTime> element within a <PermittedTime> element.

3306 Conforming **SKCL** implementations SHALL NOT use the symmetric after the end of the <EndTime> on
3307 the client machine.

3310 Some examples of the <PermittedTimes> element are shown below; other parts of their enclosing elements
3311 are not shown for brevity:

3312 **Example 1 – An example of a <PermittedTimes> element with a single<PermittedTime> element. The**
3313 **<StartTime> specifies 9:00AM on the client machine while the <EndTime> specifies 5:00PM:**


```

3314     <ekmi:PermittedTimes ekmi:any="false">
3315         <ekmi:PermittedTime>
3316             <ekmi:StartTime>09:00:00</ekmi:StartTime>
3317             <ekmi:EndTime>17:00:00</ekmi:EndTime>
3318         </ekmi:PermittedTime>
3319     </ekmi:PermittedTimes>

```

3320 **Example 2 – An example of a <PermittedTimes> element with two <PermittedTime> elements. For the**
3321 **first <PermittedTime> element , the <StartTime> element specifies 6:00AM while the <EndTime>**
3322 **element specifies 12:00 Noon. For the second <PermittedTime> element, the <StartTime> element**
3323 **specifies 3:00 PM in the afternoon, while the <EndTime> element specifies 7:00PM in the evening. This**
3324 **policy might imply that a symmetric key with this <PermittedTimes> element cannot be used during a**
3325 **lunch break of 12:00 Noon to 3:00PM:**

```

3326     <ekmi:PermittedTimes ekmi:any="false">
3327         <ekmi:PermittedTime>
3328             <ekmi:StartTime>06:00:00</ekmi:StartTime>
3329             <ekmi:EndTime>12:00:00</ekmi:EndTime>
3330         </ekmi:PermittedTime>
3331         <ekmi:PermittedTime>
3332             <ekmi:StartTime>15:00:00</ekmi:StartTime>
3333             <ekmi:EndTime>19:00:00</ekmi:EndTime>
3334         </ekmi:PermittedTime>
3335     </ekmi:PermittedTimes>

```

3336 **Example 3 – An example of a null <PermittedTimes> element, specifying that the key may be used at**
3337 **any time:**

```

3338     <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>

```

3339 4.21 Element <PermittedUses> and <PermittedUse>

3340 The element <PermittedUses>, of the type *PermittedUsesType*, is used to define the specific ways in which
3341 applications are permitted to use a symmetric key within a specific <Symkey> element.

3342 Schema Definition:

```

3343     <xsd:complexType name="PermittedUsesType" mixed="true">
3344         <xsd:sequence>
3345             <xsd:element name="PermittedUse" minOccurs="0" maxOccurs="unbounded">
3346                 <xsd:simpleType>
3347                     <xsd:restriction base="xsd:string">
3348                         <xsd:maxLength value="256"/>
3349                         <xsd:whiteSpace value="preserve"/>
3350                     </xsd:restriction>
3351                 </xsd:simpleType>
3352             </xsd:element>
3353             <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
3354         </xsd:sequence>
3355         <xsd:attribute ref="tns:any" use="required"/>
3356     </xsd:complexType>

```

3357 There SHALL be only one <PermittedUses> element within the <Permissions> element. However, there
3358 MAY be an unbounded (unlimited) number of <PermittedUse> sub-elements within the <PermittedUses>
3359 element.

3360 The <PermittedUses> element SHALL have one attribute named "any", that will have a "false" or "true" value,
3361 based on the following:

- 3362 • When the <PermittedUses> element is null (i.e. it does not have a single <PermittedUse> sub-
3363 element in it), the value of the “any” attribute SHALL be set to “true” AND the XML Schema Instance
3364 (XSI) “nil” attribute SHALL be set to “true”.
- 3365 • When the <PermittedUses> element is not-null (i.e. it has at least one <PermittedUse> sub-
3366 element in it), the value of the “any” attribute SHALL be set to “false” AND the XML Schema Instance
3367 (XSI) “nil” attribute SHALL NOT be present.

3368 A null <PermittedUses> element specifies that applications are permitted use of the symmetric key for ANY
3369 purpose, subject to complying with all other permission clauses in the <Permissions> element.

3370 Examples of the <PermittedUses> element are shown below; other parts of their enclosing elements are not
3371 shown for brevity:

3372 **Example 1 – An example of a <PermittedUses> element specifying that the symmetric key may be used**
3373 **only by VPN applications for session encryption keys:**

```
3374 <ekmi:PermittedUses ekmi:any="false">
3375   <ekmi:PermittedUse>VPN</ekmi:PermittedUse>
3376 </ekmi:PermittedUses>
```

3377 **Example 2 – An example of a <PermittedUses> element specifying that the symmetric key may be used**
3378 **only by applications on laptops and Personal Digital Assistants (PDA):**

```
3379 <ekmi:PermittedUses ekmi:any="false">
3380   <ekmi:PermittedUse>Laptop</ekmi:PermittedUse>
3381   <ekmi:PermittedUse>PDA</ekmi:PermittedUse>
3382 </ekmi:PermittedUses>
```

3383 **Example 3 – An example of a null <PermittedUses> element specifying that the symmetric key may be**
3384 **used for any purpose:**

```
3385 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
```

3386 4.22 Element <KeyCachePolicyRequest>

3387 The <KeyCachePolicyRequest> element is used to request a key-cache policy from the SKS server, so the
3388 client may know if and how to cache symmetric keys locally.

3389 While it is a top-level element within this specification, a <SymkeyRequest> element MUST be enclosed within a
3390 **SOAP Body** element of a **SOAP Envelope** to conform to the security requirements of this specification. The
3391 **SOAP Header** of the **SOAP Envelope** MUST enclose a **Security** element conforming to [WSS] with a
3392 **ValueType** attribute containing the value <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3>. The **Security** element must conform to all other requirements of the specified security profile
3393 in [WSS] to form a well-formed, secure message.
3394

3395 Schema Definition:

```
3396 <xsd:element name="KeyCachePolicyRequest">
3397   <xsd:complexType>
3398     <xsd:annotation>
3399       <xsd:documentation>
3400         No elements/attributes are defined for KeyCachePolicyRequest.
3401       </xsd:documentation>
3402     </xsd:annotation>
3403   </xsd:complexType>
3404 </xsd:element>
```

3405 The <KeyCachePolicyRequest> has no child elements. The SOAP Header of the signed request provides the
3406 SKS server with all the information it needs to process the request: the identity of the requester, strong
3407 authentication and message integrity of the request.

3408 Some examples of the use of the <SymkeyRequest> element are as follows:

3409 **Example 1 – An example of a <KeyCachePolicyRequest>; the surrounding SOAP envelope is not**
3410 **displayed here for brevity:**

```
3411     <ekmi:KeyCachePolicyRequest  
3412         xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01"/>
```

3413 4.23 Element <KeyCachePolicyResponse>

3414 The <KeyCachePolicyResponse> element is the response sent by an **SKS** Server to a client that requested a
3415 key-cache policy through a <KeyCachePolicyRequest>. The <KeyCachePolicyResponse> contains
3416 policy elements, which define rules that conforming implementations of the **SKCL** MUST adhere to when
3417 caching symmetric keys sent by the **SKS** Server.

3418 Schema Definition:

```
3419     <xsd:element name="KeyCachePolicyResponse">  
3420         <xsd:complexType>  
3421             <xsd:sequence>  
3422                 <xsd:element  
3423                     name="KeyCachePolicy"  
3424                     type="ekmi:KeyCachePolicyType"  
3425                     minOccurs="1" maxOccurs="unbounded"/>  
3426             </xsd:sequence>  
3427         </xsd:complexType>  
3428     </xsd:element>
```

3429 The <KeyCachePolicyResponse> element consists of a minimum of one, but an unbounded (unlimited)
3430 number of <KeyCachePolicy> children elements.

3431 4.24 Element <KeyCachePolicy>

3432 The <KeyCachePolicy> element contains policy elements, which define rules that conforming implementations
3433 of the **SKCL** MUST adhere to when caching symmetric keys sent by the **SKS** Server.

3434 Schema Definition:

```
3435     <xsd:element name="KeyCachePolicyResponse">  
3436         <xsd:complexType>  
3437             <xsd:sequence>  
3438                 <xsd:element  
3439                     name="KeyCachePolicy"  
3440                     type="ekmi:KeyCachePolicyType"  
3441                     minOccurs="1" maxOccurs="unbounded"/>  
3442             </xsd:sequence>  
3443         </xsd:complexType>  
3444     </xsd:element>  
  
3445     <xsd:complexType name="KeyCachePolicyType" mixed="true">  
3446         <xsd:sequence>  
3447             <xsd:element name="KeyCachePolicyID" type="tns:TwoPartIDType"/>  
3448             <xsd:element name="PolicyName">  
3449                 <xsd:simpleType>  
3450                     <xsd:restriction base="xsd:string">  
3451                         <xsd:maxLength value="255"/>  
3452                         <xsd:whiteSpace value="preserve"/>  
3453                     </xsd:restriction>  
3454                 </xsd:simpleType>  
3455             </xsd:sequence>
```

```

3456     <xsd:element name="Description" nillable="true">
3457         <xsd:simpleType>
3458             <xsd:restriction base="xsd:string">
3459                 <xsd:maxLength value="2048"/>
3460                 <xsd:whiteSpace value="preserve"/>
3461             </xsd:restriction>
3462         </xsd:simpleType>
3463     </xsd:element>
3464     <xsd:element name="KeyClass" type="tns:KeyClassType"/>
3465     <xsd:element name="StartDate" type="xsd:dateTime"/>
3466     <xsd:element name="EndDate" type="xsd:dateTime" nillable="true"/>
3467     <xsd:element name="PolicyCheckInterval">
3468         <xsd:simpleType>
3469             <xsd:restriction base="xsd:nonNegativeInteger">
3470                 <xsd:minInclusive value="0"/>
3471                 <xsd:maxInclusive value="2592000"/>
3472             </xsd:restriction>
3473         </xsd:simpleType>
3474     </xsd:element>
3475     <xsd:element name="Status" type="tns:StatusType"/>
3476     <xsd:element
3477         name="NewKeysCacheDetail"
3478         type="tns:KeyCacheDetailType"
3479         minOccurs="0"/>
3480     <xsd:element
3481         name="UsedKeysCacheDetail"
3482         type="tns:KeyCacheDetailType"
3483         minOccurs="0"/>
3484     </xsd:sequence>
3485 </xsd:complexType>

```

3486 The <KeyCachePolicy> element is of the **KeyCachePolicyType** and consists of the following child elements:

3487 1. <KeyCachePolicyID> [Required]

3488 The <KeyCachePolicyID> element, of type **TwoPartIDType**, identifies the unique policy object within
3489 the **SKMS**. There SHALL be only one <KeyCachePolicyID> element within a <KeyCachePolicy>
3490 element.
3491

3492 The **TwoPartIDType** is specified in Section 4.8.
3493

3494 2. <PolicyName> [Required]

3495 The <PolicyName> element, of type XSD **String**, with a maximum length of 255 characters, identifies
3496 a unique name given to this <KeyCachePolicy>. There SHALL be only one <PolicyName> element
3497 within a <KeyCachePolicy> element.
3498

3499 3. <Description> [Required]

3500 The <Description> element, of type XSD **String**, with a maximum length of 2048 characters,
3501 provides a human-readable description of this policy. There SHALL be only one <Description>
3502 element within a <KeyCachePolicy> element.
3503

3504 The <Description> MAY be an empty element, but MUST exist within the <KeyCachePolicy>
3505 element.
3506

3507 4. <KeyClass> [Required]

3508 This element of type **KeyClassType** identifies the key-class of the symmetric key to which this policy
3509 applies.
3510

- 3511 5. <StartDate> [Required]
3512
3513 The <StartDate> element , of type XSD *dateTime*, specifies the date and time at which this policy
3514 becomes effective. There SHALL be only one <StartDate> element within a <KeyCachePolicy>
3515 element.
- 3516 6. <EndDate> [Required]
3517
3518 The <EndDate> element , of type XSD *dateTime*, specifies the date and time at which this policy
3519 expires. There SHALL be only one <EndDate> element within a <KeyCachePolicy> element.
3520
3521 The <EndDate> MAY be an empty element, but MUST exist within the <KeyCachePolicy> element.
- 3522 7. <PolicyCheckInterval> [Required]
3523
3524 The <PolicyCheckInterval> element , of type XSD *nonNegativeInteger*, specifies the frequency at
3525 which the client SHALL check the **SKS** server for updates to this policy. This frequency is specified in
3526 seconds and SHALL NOT exceed 2592000 seconds (30 calendar days). There SHALL be only one
3527 <PolicyCheckInterval> element within a <KeyCachePolicy> element.
- 3528 8. <Status> [Required]
3529
3530 The <Status> element, of type *StatusType*, identifies the current status of this policy within the SKMS.
3531 There SHALL be only one <Status> element within a <KeyCachePolicy> element.
3532
3533 The *StatusType* is specified in Section 4.11.
- 3534 9. <NewKeysCacheDetail> [Required]
3535
3536 The <NewKeysCacheDetail> element, of type *KeyCacheDetailType*, defines how many new (as yet
3537 unused for any encryption transaction) symmetric keys a client may cache, and for how long. It is the
3538 responsibility of the conforming **SKCL** implementation to enforce these rules.
3539
3540 The absence of the <NewKeysCacheDetail> element implies that new symmetric keys SHALL NEVER
3541 be cached on the client. New keys may be cached only when this element exists, and SHALL conform
3542 to the rules specified in this element.
3543
3544 When it exists, there SHALL be only one <NewKeysCacheDetail> element in a <KeyCachePolicy>
3545 element.
3546
3547 The *KeyCacheDetailType* is specified in Section 4.22.
- 3548 10. <UsedKeysCacheDetail> [Required]
3549
3550 The <UsedKeysCacheDetail> element, of type *KeyCacheDetailType*, defines how many used
3551 symmetric keys a client may cache, and for how long. It is the responsibility of the conforming **SKCL**
3552 implementation to enforce these rules.
3553
3554 The absence of the <UsedKeysCacheDetail> element implies that used symmetric keys SHALL
3555 NEVER be cached on the client. Used keys may be cached only when this element exists, and SHALL
3556 conform to the rules specified in this element.
3557
3558 When it exists, there SHALL be only one <UsedKeysCacheDetail> element in a
3559 <KeyCachePolicy> element.
3560
3561 The *KeyCacheDetailType* is specified in Section 4.22.
- 3562 Some examples of the <KeyUsePolicy> element are as follows.

3563 **Example 1 – A <KeyCachePolicy> that is valid between January 01, 2008 and December 31, 2008. It**
3564 **requires the client to check for policy updates every day and allows 3 new and 3 used keys to be cached**
3565 **for up to 90 days:**

```
3566     <ekmi:KeyCachePolicy>
3567         <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
3568         <ekmi:PolicyName>
3569             Corporate Laptop Symmetric Key Caching Policy
3570         </ekmi:PolicyName>
3571         <ekmi:Description>
3572             This policy defines how company-issued laptops will manage
3573             symmetric keys used for file/disk encryption in each laptop's
3574             local cache. This policy must be used by all laptops that
3575             use the company EKMI.
3576         </ekmi:Description>
3577         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
3578         <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
3579         <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
3580         <ekmi>Status>Active</ekmi>Status>
3581         <ekmi:NewKeysCacheDetail>
3582             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
3583             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
3584         </ekmi:NewKeysCacheDetail>
3585         <ekmi:UsedKeysCacheDetail>
3586             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
3587             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
3588         </ekmi:UsedKeysCacheDetail>
3589     </ekmi:KeyCachePolicy>
```

3590 **Example 2 – A <KeyCachePolicy> that is effective starting January 01, 2008 and never expires. It does**
3591 **NOT permit any caching of symmetric keys through the absence of the detail elements on caching:**

```
3592     <ekmi:KeyCachePolicy>
3593         <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
3594         <ekmi:PolicyName>
3595             No Caching Policy
3596         </ekmi:PolicyName>
3597         <ekmi:Description>
3598             This policy is for high-risk, always-connected machines on the
3599             network, which will never cache symmetric keys locally. This
3600             policy never expires (but checks monthly for any updates).
3601         </ekmi:Description>
3602         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
3603         <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
3604         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
3605         <ekmi>Status>Active</ekmi>Status>
3606     </ekmi:KeyCachePolicy>
```

3607 **4.25 Type *KeyCacheDetailType***

3608 The ***KeyCacheDetailType*** type allows SKS servers to specify precisely how many symmetric keys MAY be
3609 cached on the client machine, and for how long.

3610 **Schema Definition:**

```
3611     <xsd:complexType name="KeyCacheDetailType">
3612         <xsd:sequence>
3613             <xsd:element name="MaximumKeys" minOccurs="1">
3614                 <xsd:simpleType>
3615                     <xsd:restriction base="xsd:integer">
```

```

3616         <xsd:minInclusive value="0"/>
3617         <xsd:maxInclusive value="18446744073709551615"/>
3618     </xsd:restriction>
3619 </xsd:simpleType>
3620 </xsd:element>
3621 <xsd:element name="MaximumDuration" minOccurs="1">
3622     <xsd:simpleType>
3623         <xsd:restriction base="xsd:integer">
3624             <xsd:minInclusive value="0"/>
3625             <xsd:maxInclusive value="18446744073709551615"/>
3626         </xsd:restriction>
3627     </xsd:simpleType>
3628 </xsd:element>
3629 </xsd:sequence>
3630 </xsd:complexType>

```

3631 The **KeyCacheDetailType** consists of the following child elements:

3632 1. <MaximumKeys> [Required]

3633
3634 The <MaximumKeys> element, of type XSD **Integer**, specifies the maximum number of symmetric keys
3635 that MAY be cached on a client machine. It SHALL be a positive number between the values 0 and
3636 18446744073709551615. There SHALL be only one <MaximumKeys> element within an element that
3637 uses the **KeyCacheDetailType**.

3638 2. <MaximumDuration> [Required]

3639
3640 The <MaximumDuration> element, of type XSD **Integer**, specifies the maximum number of seconds
3641 that symmetric keys MAY be cached on a client machine. It SHALL be a positive number between the
3642 values 0 and 18446744073709551615. There SHALL be only one <MaximumDuration> element
3643 within an element that uses the **KeyCacheDetailType**.

3644 Examples of the **KeyCacheDetailType** when used in the <KeyCachePolicy> element are as follows.

3645 **Example 1 – A <KeyCachePolicy> that is valid between January 01, 2008 and December 31, 2008. It**
3646 **requires the client to check for policy updates every day and allows 3 new and 3 used keys to be cached**
3647 **for up to 90 days:**

```

3648 <ekmi:KeyCachePolicy>
3649   <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
3650   <ekmi:PolicyName>
3651     Corporate Laptop Symmetric Key Caching Policy
3652   </ekmi:PolicyName>
3653   <ekmi:Description>
3654     This policy defines how company-issued laptops will manage
3655     symmetric keys used for file/disk encryption in their local
3656     cache. This policy must be used by all laptops that use
3657     the company EKMI.
3658   </ekmi:Description>
3659   <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
3660   <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
3661   <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
3662   <ekmi:Status>Active</ekmi:Status>
3663   <ekmi:NewKeysCacheDetail>
3664     <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
3665     <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
3666   </ekmi:NewKeysCacheDetail>
3667   <ekmi:UsedKeysCacheDetail>
3668     <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
3669     <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>

```


3670 </ekmi:UsedKeysCacheDetail>
3671 </ekmi:KeyCachePolicy>

3672 **Example 1 – A <KeyCachePolicy> that is valid between January 01, 2008 and December 31, 2008. It**
3673 **requires the client to check for policy updates every day and allows 1 new and 0 used keys to be cached**
3674 **for upto 15 days:**

```
3675       <ekmi:KeyCachePolicy>
3676           <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
3677           <ekmi:PolicyName>
3678               Corporate Laptop Symmetric Key Caching Policy
3679           </ekmi:PolicyName>
3680           <ekmi:Description>
3681               This policy defines how company-issued laptops will manage
3682               symmetric keys used for file/disk encryption in each laptop's
3683               local cache. This policy must be used by all laptops that
3684               use the company EKMI.
3685           </ekmi:Description>
3686           <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
3687           <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
3688           <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
3689           <ekmi>Status>Active</ekmi>Status>
3690           <ekmi:NewKeysCacheDetail>
3691               <ekmi:MaximumKeys>1</ekmi:MaximumKeys>
3692               <ekmi:MaximumDuration>1296000</ekmi:MaximumDuration>
3693           </ekmi:NewKeysCacheDetail>
3694           <ekmi:UsedKeysCacheDetail>
3695               <ekmi:MaximumKeys>0</ekmi:MaximumKeys>
3696               <ekmi:MaximumDuration>1296000</ekmi:MaximumDuration>
3697           </ekmi:UsedKeysCacheDetail>
3698       </ekmi:KeyCachePolicy>
3699
```

3700

5 Conformance

3701 An implementation conforms to this specification if it satisfies all of the MUST or REQUIRED level requirements
3702 defined within this specification. An SKSML Node MUST NOT use the XML namespace identifier for this
3703 specification (listed in the Title section under Declared Namespace(s)) within SOAP Envelopes unless it is
3704 compliant with this specification.

3705 This specification references a number of other specifications (see the table above). In order to comply with this
3706 specification, an implementation MUST implement the portions of referenced specifications necessary to comply
3707 with the required provisions of this specification. Additionally, the implementation of the portions of the
3708 referenced specifications that are specifically cited in this specification MUST comply with the rules for those
3709 portions as established in the referenced specification.

3710 Additionally normative text within this specification takes precedence over normative outlines, which in turn take
3711 precedence over the XML Schema [XML Schema Part 1, Part 2] descriptions. That is, the normative text in this
3712 specification further constrains the schema part of this specification; and this specification contains further
3713 constraints on the elements defined in referenced schemas.

3714 If an OPTIONAL message is not supported, then the implementation SHOULD Fault just as it would for any other
3715 unrecognized/unsupported message. If an OPTIONAL message is supported, then the implementation MUST
3716 satisfy all of the MUST and REQUIRED sections of the message.

3717 **Appendix A. Acknowledgments**

3718 The following individuals have participated in the creation of this specification and are gratefully
3719 acknowledged

3720 **Participants:**

3721 •

3722

Appendix B. Revision History

Version	Date	Author	Notes
DRAFT 4	June 08, 2008	Arshad Noor	Initial version
DRAFT 5	June 17, 2008	Arshad Noor	Moved non-normative sections to their own document. KeyClass element was added to KeyCachePolicy. KeyCachePolicy is now embedded inside a KeyCachePolicyResponse.
DRAFT 6	July 7, 2008	Arshad Noor	Modified Permissions object to include all sub-elements on a mandatory basis. Modified all abbreviations in elements to expand to full names.
PR 1	July 22, 2008	Arshad Noor	Modified Title page information to conform with OASIS standards. Brought some Background information into this document from the information document, into Section 2. Added a Conformance section to Section 4.

3725

Appendix C. Non-Normative Text

3726

Appendix D. SKSML Error Codes and Error Messages