



Symmetric Key Services Markup Language (SKSML) Version 1.0

Public Review Draft 03

19 March 2010

Specification URIs:

This Version:

<http://docs.oasis-open.org/ekmi/sksml/v1.0/pr02/SKSML-1.0-Specification.html>

<http://docs.oasis-open.org/ekmi/sksml/v1.0/pr02/SKSML-1.0-Specification.odt> (Authoritative)

<http://docs.oasis-open.org/ekmi/sksml/v1.0/pr02/SKSML-1.0-Specification.pdf>

Previous 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> (Authoritative)

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

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>

Technical Committee:

OASIS Enterprise Key Management Infrastructure (EKMI) TC

Chair(s):

Anil Saldhana, Red Hat Inc. (anil.saldhana@redhat.com)

Timothy Bruce, CA (timothy.bruce@ca.com)

Editor(s):

Anil Saldhana, Red Hat Inc. (anil.saldhana@redhat.com)

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

Arshad Noor, Individual (arshad.noor@strongauth.com)

Related Work:

This specification replaces or supercedes:

- None

This specification is related to:

- 32 • Advanced Encryption Standard (AES) - NIST FIPS 197 -
33 <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
- 34 • Simple Object Access Protocol (SOAP) - W3C Recommendation 08 May 2000.
35 <http://www.w3.org/TR/soap/>
- 36 • XML Encryption - W3C Recommendation 10 Dec 2002. <http://www.w3.org/TR/xmlenc-core/>
- 37 • XML Signature - W3C Recommendation 12 Feb 2002. <http://www.w3.org/TR/xmldsig-core/>
- 38 • Web Services Security - SOAP Message Security 1.0 - OASIS Standard 200401, March
39 2004 - [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
40 1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)

41 **Declared XML Namespace(s):**
42 <http://docs.oasis-open.org/ekmi/2008/01>

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

53 **Status:**
54 This document was last revised or approved by the EKMI TC on the above date. The level of
55 approval is also listed above. Check the "Latest Version" or "Latest Approved Version" location
56 noted above for possible later revisions of this document.

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

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

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

67 Notices

68 Copyright © OASIS® 2008. All Rights Reserved.

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

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

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

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

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

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

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

107 The names "OASIS", [insert specific trademarked names, abbreviations, etc. here] are trademarks of
108 OASIS, the owner and developer of this specification, and should be used only to refer to the organization
109 and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications,
110 while reserving the right to enforce its marks against misleading uses. Please see [http://www.oasis-](http://www.oasis-open.org/who/trademark.php)
111 [open.org/who/trademark.php](http://www.oasis-open.org/who/trademark.php) for above guidance.

112

113 Table of Contents

114	1 Introduction.....	6
115	1.1 Terminology.....	6
116	1.2 Glossary.....	6
117	1.3 Normative References.....	7
118	2 Background (non-normative).....	9
119	2.1 Requirements (Non-normative).....	11
120	3 Examples of use of SKSML (non-normative).....	12
121	3.1 Request for a new symmetric key.....	13
122	3.2 Response with a new symmetric key.....	15
123	3.3 Request for an existing symmetric key.....	19
124	3.4 Response with an existing symmetric key.....	20
125	3.5 Request for a new symmetric key of a specific KeyClass.....	20
126	3.6 Response with a new symmetric key of a specific KeyClass.....	20
127	3.7 Request for multiple new symmetric keys.....	21
128	3.8 Response with multiple new symmetric keys.....	22
129	3.9 Response with an SKS error.....	26
130	3.10 Response with symmetric keys and errors.....	27
131	3.11 Request for a symmetric key-caching policy.....	30
132	3.12 Response with a symmetric key-caching policy (1).....	31
133	3.13 Response with a symmetric key-caching policy (2).....	33
134	3.14 Response with multiple symmetric key-caching policies (3).....	34
135	4 Specification.....	39
136	4.1 Element <SymkeyRequest>.....	39
137	4.2 Element <GlobalKeyID>.....	41
138	4.3 Element <KeyClasses> and <KeyClass>.....	42
139	4.4 Element <SymkeyResponse>.....	44
140	4.5 Element <Symkey>.....	46
141	4.6 Element <SymkeyError>.....	48
142	4.7 Element <KeyUsePolicy>.....	49
143	4.8 Type TwoPartIDType.....	52
144	4.9 Element <KeyAlgorithm>.....	53
145	4.10 Element <KeySize>.....	54
146	4.11 Element <Status>.....	55
147	4.12 Element <Permissions>.....	56
148	4.13 Element <PermittedApplications> and <PermittedApplication>.....	62
149	4.14 Element <PermittedDates> and <PermittedDate>.....	66
150	4.15 Element <PermittedDays> and <PermittedDay>.....	68
151	4.16 Element <PermittedDuration>.....	69
152	4.17 Element <PermittedLevels> and <PermittedLevel>.....	70

153	4.18 Element <PermittedLocations> and <PermittedLocation>.....	72
154	4.19 Element <PermittedNumberOfTransactions>.....	74
155	4.20 Element <PermittedTimes> and <PermittedTime>.....	76
156	4.21 Element <PermittedUses> and <PermittedUse>.....	78
157	4.22 Element <KeyCachePolicyRequest>.....	79
158	4.23 Element <KeyCachePolicyResponse>.....	79
159	4.24 Element <KeyCachePolicy>.....	80
160	4.25 Type KeyCacheDetailType.....	83
161	5 Bindings.....	86
162	5.1 W3C Security Binding.....	86
163	5.2 Mutually Authenticated TLS Binding.....	86
164	5.3 SOAP-WSS Binding.....	86
165	6 Conformance.....	87
166		

167 1 Introduction

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

172 1.1 Terminology

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

176 1.2 Glossary

177 **3DES** – Triple Data Encryption Standard

178 **AES** – Advanced Encryption Standard

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

180 **Ciphertext** – Encrypted data

181 **Cryptographic module** – A software library or hardware module dedicated to performing cryptographic
182 operations

183 **DES** – Data Encryption Standard

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

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

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

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

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

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

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

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

201 **Plaintext** – Unencrypted data

202 **SHA** – Secure Hashing Algorithm

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

204 **SHA-256** – Secure Hashing Algorithm with a resultant size of 256-bits
 205 **SHA-384** – Secure Hashing Algorithm with a resultant size of 384-bits
 206 **SHA-512** – Secure Hashing Algorithm with a resultant size of 512-bits
 207 **SID** or **Server ID** – Server Identifier; the unique integer assigned to every **SKS** server within an
 208 enterprise's **SKMS**
 209 **SKCL** – Symmetric Key Client Library; a software library that supports the **SKSML** protocol
 210 **SKMS** – Symmetric Key Management System; a collection of hardware and software providing symmetric
 211 encryption key-management services
 212 **SKS** – Symmetric Key Services; a server that provides symmetric key management services over a
 213 network or other mechanism selected by implementers
 214 **SKSML** – Symmetric Key Services Markup Language; an XML-based protocol to request and receive
 215 symmetric encryption key-management services
 216 **SOAP** – Simple Object Access Protocol
 217 **SOAP Body** – The content part of a SOAP message
 218 **SOAP Envelope** – The SOAP message consisting of a SOAP Header and a SOAP Body, conforming to
 219 the SOAP protocol standard.
 220 **SOAP Error** – A SOAP error message response to a SOAP request
 221 **SOAP Header** – The header part of a SOAP message containing meta-information about the message,
 222 including security-related objects
 223 **Symkey** - A symmetric encryption key
 224 **unbounded** – A parameter used with the “maxOccurs” attribute to indicate an unlimited number
 225 **XMLEncryption** – Encrypted content represented in eXtensible Markup Language that conforms to the
 226 World Wide Web Consortium's XML Encryption standard
 227 **XMLSignature** – A digital signature represented in eXtensible Markup Language that conforms to the
 228 World Wide Web Consortium's XML Signature standard

229 1.3 Normative References

230 **[AES]** Advanced Encryption Standard
 231 NIST FIPS 197. <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
 232 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*.
 233 IETF RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>
 234 **[SOAP]** Simple Object Access Protocol 1.2
 235 W3C Recommendation 27 April 2007. <http://www.w3.org/TR/soap12/>
 236 **[XMLEncryption]** XML Encryption Syntax and Processing
 237 W3C Recommendation 10 Dec 2002. <http://www.w3.org/TR/xmlenc-core/>
 238 **[XMLSignature]** XML Signature Syntax and Processing
 239 W3C Recommendation 12 Feb 2002. <http://www.w3.org/TR/xmlsig-core/>
 240 **[WSS]** Web Services Security – SOAP Message Security 1.0
 241 OASIS Standard 200401, January 2004 <http://docs.oasis->
 242 open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
 243 **[RFC 2578]** K. McCloghrie, et al. *Structure of Management Information Version 2(SMIv2)*.

244
245
246
247
248

IETF RFC 2578, April 1999. <http://www.ietf.org/html/rfc2578>

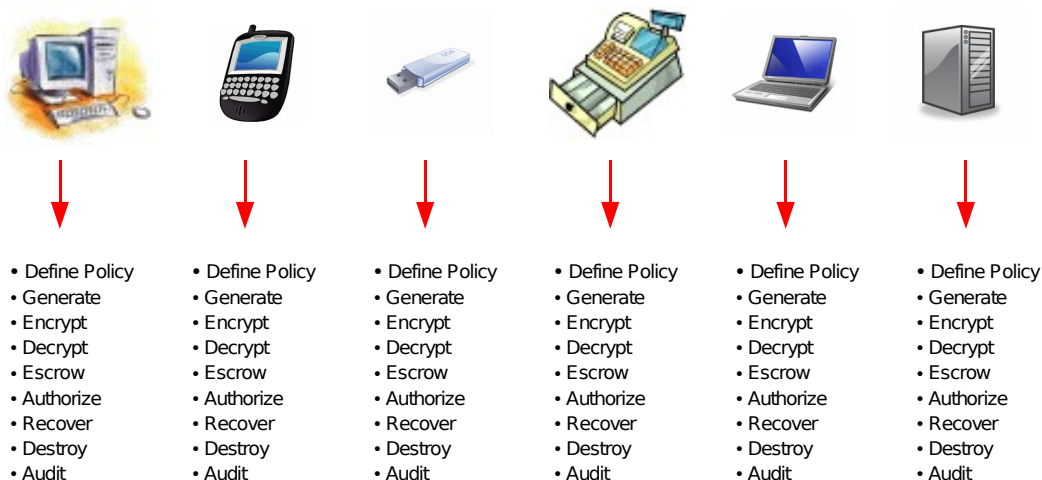
2 Background (non-normative)

249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285

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

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

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



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

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

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

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

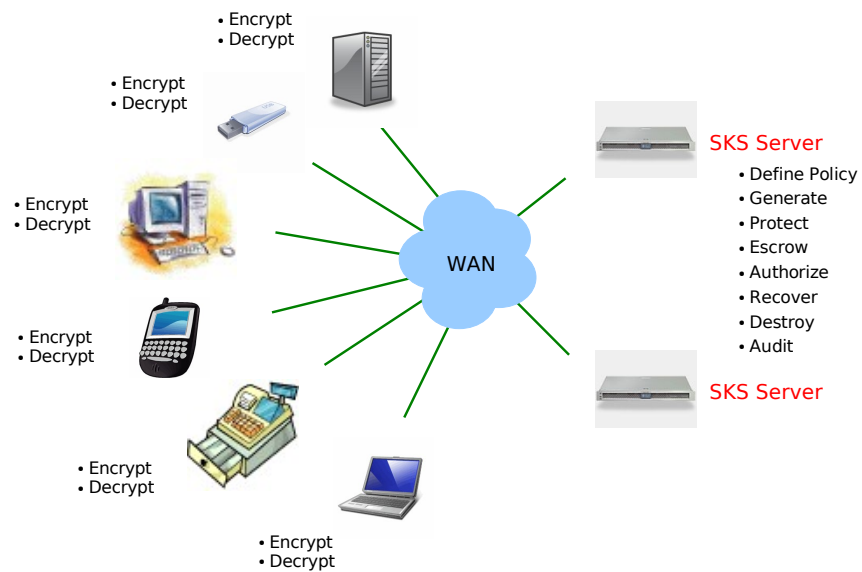
316

317

318

319

320



321 With an architecture similar to the Domain Name Service (DNS), an SKMS becomes the focal point for all
322 symmetric encryption key-management services.

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

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

328 2.1 Requirements (Non-normative)

329 The requirements of the SKSML protocol are that:

- 330 • It must be platform independent;
- 331 • It must support the request of new and previously escrowed symmetric encryption keys;
- 332 • It must support the unique identification of every symmetric encryption key on the internet;
- 333 • It must provide message authenticity, confidentiality and integrity even when used over insecure
334 networks;
- 335 • It must support the use of encryption/decryption services by a client even when disconnected
336 from the network;
- 337 • It must provide flexibility in defining key-usage policies;

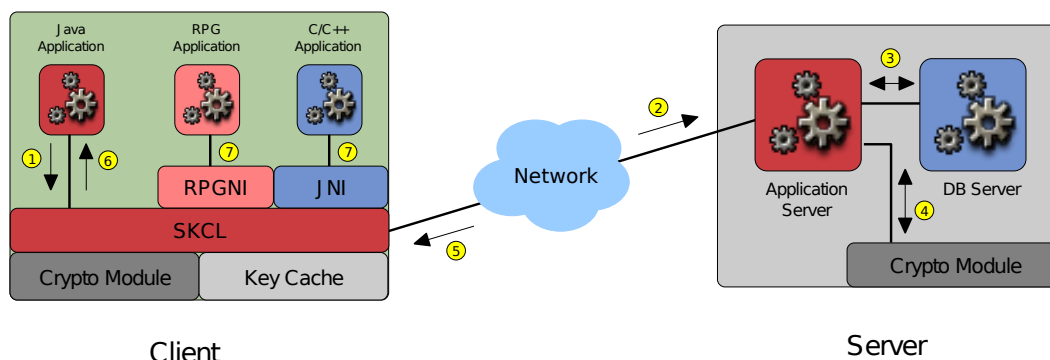
338 SKSML meets the above requirements in the following manner:

- 339 • SKSML uses XML for encapsulating its requests and responses and can thus, be used on any
340 platform that supports this protocol;
- 341 • Using a scheme that concatenates unique Domain identifiers (Private Enterprise Numbers issued
342 by the IANA), unique SKS Server identifiers within a domain and unique Key identifiers within an
343 SKS server, SKSML creates Global Key Identifiers (GKID) that can uniquely identify symmetric
344 keys across the internet;
- 345 • SKSML relies on RSA cryptographic key-pairs and digital certificates enabling XML encryption
346 (confidentiality) and XML signatures (for authenticity and message integrity);
- 347 • Using secure key-caching enabled through centrally-defined policies, SKSML supports the
348 request and receipt of **KeyCachePolicy** elements by clients for the use of symmetric encryption
349 keys even when the client is disconnected from the network and an SKS server;
- 350 • SKSML provides significant flexibility for defining policies on how symmetric encryption keys may
351 be used by client applications. The **KeyUsePolicy** element allows Security Officers to define
352 which applications may use a specific key, days and times of use, location of use, purpose of
353 use, key-sizes, encryption algorithms, etc.

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

3 Examples of use of SKSML (non-normative)

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

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

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

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

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

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

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

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

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

399

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

```
402 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
403   <SOAP-ENV:Header>
404     <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
405     1.0.xsd" SOAP-ENV:mustUnderstand="1">
406       <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
407       utility-1.0.xsd"
408         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
409       security-1.0#Base64Binary"
410         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
411       1.0#X509v3" wsu:Id="XWSSGID-1172790302111-1738806553">
412
413
414       MIIDfDCCAmSgAwIBAgIIAe/AvliGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdU3Ryb25nab
415
416       S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvciBTdHJvbmdLZXkgREVNTyBVc2Ug1d
417       T25seTEXMBUGA1UEChMOU3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTcxMDMwWWhcNMDcwNzla64dd3k
418       A1UECXMbRm9yIFN0cm9uZ0tleSBERU1PIFVzZSBPbmx5MRcwFQYDVQQKEw5TdHJvbmdBdXR0eI2da
419       S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvciBTdHJvbmdLZXkgREVNTyBVc2Ugia
420       T25seTEXMBUGA1UEChMOU3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTY0NjEwWWhcNMDcwNzI1s34wdd
421       NjEwWjBpMREwDwYKZCIzImZlPyLGQBARMBOTEVMBMGA1UEAxMMU0tTIFNlcnZlci0xMSQwIlgYDVQsdw2
422       ExtGb3IglU3Ryb25nS2V5IERFTU8gVXNlIE9ubHkxZzAvbG9uZDQwZD4wZD4wZD4wZD4wZD4wZD4wZD4w
423
424       NBGkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAztpqRoU5A8plxx1Rz1QEUnIAAM1D5g9+islr3wxa
425       hbwjtFSMYilnY4iv77xU/nsMOnMZ7RxsLYKdCzQ1ODVYqQwqmAvaJ5Z6SVy34gZ51YG+rSWE3NjFsd
426       bOXW8RJYA/Tn6Lmht/qngrcqqmtP0cAAiMRZOWtCTmC2K/LEqDabXSyU6Hh8ySNE3njbvmWpresf
427       zsYokTdvWQqT6tKo1OwJsdJ1+hxM7DnMLvMNq5reINfsKhDdX17wzhrBUx+hiYA/qo8tMXkL6wsd
428       4PN5dYugtZpSzldUO5tlg58Avhzo7hy5oofBIKFY22CeljQ36u0bMjuyGj6UYHs3rdfsds32rda
429       YzCBnzANBgkqhkiG9w0BAQEFAAOBjQAwYkCgYEAyAmxMZHyA8wHJ4UE4b61s51JVWe4Fygj4MCf3a
430       hvcNAQELBQADggEBACK05PtvZD4WPgI0e=
431     </wsse:BinarySecurityToken>
432     <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
433       <ds:SignedInfo>
434         <ds:CanonicalizationMethod
435           Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
436         <InclusiveNamespaces
437           xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"
438           PrefixList="wsse SOAP-ENV"/>
439         </ds:CanonicalizationMethod>
440         <ds:SignatureMethod
441           Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
442         <ds:Reference URI="#XWSSGID-1172790300636-653454040">
443         <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
444         <ds:DigestValue>IU4m+rp4oebgl9g+t3nRaZYqUIE=</ds:DigestValue>
445         </ds:Reference>
446         <ds:Reference URI="#XWSSGID-1172790300637708871805">
447         <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
448         <ds:DigestValue>WCpOmTCbffcEHXhGf5rlEYWLrZg=</ds:DigestValue>
449         </ds:Reference>
450       </ds:SignedInfo>
451     </ds:SignatureValue>
```

```

452 svStAvBRRrF+g2biPI7uWHkJTQPll8t4phMbOZQsZlQcn36tcMSj/a4+4LPNfOB3Y8yO2lr1Oa1
453
454
455 fGqCPAWZNuEH34VQEM196rRwV258mgp8uwpXEYJlgPJqg89w8+/NdaODccLQ2Bizu7QM/HSM2ab
456 ogNjwqmbSylazOsnOcU=
457 </ds:SignatureValue>
458 <ds:KeyInfo>
459 <wsse:SecurityTokenReference xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
460 200401-wss-wssecurity-utility-1.0.xsd"
461 wsu:Id="XWSSGID-1172790300633-442423344">
462 <wsse:Reference URI="#XWSSGID-1172790302111-1738806553"
463 ValueTypes="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-
464 token-profile-1.0#X509v3"/>
465 </wsse:SecurityTokenReference>
466 </ds:KeyInfo>
467 </ds:Signature>
468 <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
469 1.0.xsd" wsu:Id="XWSSGID-1172790300637708871805">
470 <wsu:Created>2007-03-01T23:05:00Z</wsu:Created>
471 <wsu:Expires>2007-03-01T23:05:05Z</wsu:Expires>
472 </wsu:Timestamp>
473 </wsse:Security>
474 </SOAP-ENV:Header>
475 <SOAP-ENV:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
476 1.0.xsd" wsu:Id="XWSSGID-1172790300636-653454040">
477 <ekmi:SymkeyRequest
478 xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
479 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
480 </ekmi:SymkeyRequest>
481 </SOAP-ENV:Body>
482 </SOAP-ENV:Envelope>

```

483 3.2 Response with a new symmetric key

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

```

489 [b01] <ekmi:SymkeyResponse
490 [b02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
491 [b03]     xmlns:xenc='http://www.w3.org/2001/04/xmlenc#' >
492 [b04] <ekmi:Symkey>
493 [b05]     <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
494 [b06]     <ekmi:KeyUsePolicy>
495 [b07]         <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>
496 [b08]         <ekmi:PolicyName>DES-EDE KeyUsePolicy</ekmi:PolicyName>
497 [b09]         <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
498 [b10]         <ekmi:KeyAlgorithm>
499 [b11]             http://www.w3.org/2001/04/xmlenc#tripleDES-cbc
500 [b12]         </ekmi:KeyAlgorithm>
501 [b13]         <ekmi:KeySize>192</ekmi:KeySize>
502 [b14]         <ekmi:Status>Active</ekmi:Status>
503 [b15]     <ekmi:Permissions>
504 [b16]         <ekmi:PermittedApplications ekmi:any="false">
505 [b17]             <ekmi:PermittedApplication>
506 [b18]                 <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
507 [b19]                 <ekmi:ApplicationName>
508 [b20]                     Payroll Application

```

```

509 [b21] </ekmi:ApplicationName>
510 [b22] <ekmi:ApplicationVersion>1.0</ekmi:ApplicationVersion>
511 [b23] <ekmi:ApplicationDigestAlgorithm>
512 [b24] http://www.w3.org/2000/09/xmlsig#sha1
513 [b25] </ekmi:ApplicationDigestAlgorithm>
514 [b26] <ekmi:ApplicationDigestValue>
515 [b27] NIG4bKkt4cziEqFFu0oBTM81efU=
516 [b28] </ekmi:ApplicationDigestValue>
517 [b29] </ekmi:PermittedApplication>
518 [b30] </ekmi:PermittedApplications>
519 [b31] <ekmi:PermittedDates ekmi:any="false">
520 [b32] <ekmi:PermittedDate>
521 [b33] <ekmi:StartDate>2008-01-01</ekmi:StartDate>
522 [b34] <ekmi:EndDate>2008-12-31</ekmi:EndDate>
523 [b35] </ekmi:PermittedDate>
524 [b36] </ekmi:PermittedDates>
525 [b37] <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
526 [b38] <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
527 [b39] <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
528 [b40] <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
529 [b41] <ekmi:PermittedNumberOfTransactions ekmi:any="true"
530 xsi:nil="true"/>
531 [b42] <ekmi:PermittedTimes ekmi:any="false">
532 [b43] <ekmi:PermittedTime>
533 [b44] <ekmi:StartTime>07:00:00</ekmi:StartTime>
534 [b45] <ekmi:EndTime>19:00:00</ekmi:EndTime>
535 [b46] </ekmi:PermittedTime>
536 [b47] </ekmi:PermittedTimes>
537 [b48] <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
538 [b49] </ekmi:Permissions>
539 [b50] </ekmi:KeyUsePolicy>
540 [b51] <ekmi:EncryptionMethod
541 [b52] Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
542 [b53] <xenc:CipherData>
543 [b54] <xenc:CipherValue>
544 [b55] E9zWB/y93hVSzeTLiDcQoDxmLNxTuxSffmNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJM
545 [b56] hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/2fw
546 [b57] UJow/qimxi8+huUYJMtaGhtXuLlWtx27STRcRpIsY=
547 [b58] </xenc:CipherValue>
548 [b59] </xenc:CipherData>
549 [b60] </ekmi:Symkey>
550 [b61] </ekmi:SymkeyResponse>

```

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

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

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

556 [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.

560 [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.

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

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

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

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

570 [b11] identifies the cryptographic algorithm that this symmetric key must be used with to for cryptographic
571 operations.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

690 3.3 Request for an existing symmetric key

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

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

```
702 [c01] <ekmi:SymkeyRequest  
703 [c02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
704 [c03]     <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>  
705 [c04] </ekmi:SymkeyRequest>
```

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

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

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

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

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

715 3.4 Response with an existing symmetric key

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

721 3.5 Request for a new symmetric key of a specific KeyClass

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

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

```
730 [e01] <ekmi:SymkeyRequest  
731 [e02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
732 [e03]     <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
733 [e04]     <ekmi:KeyClasses>
```

```
734 [e05]      <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
735 [e06]      </ekmi:KeyClasses>
736 [e07]      </ekmi:SymkeyRequest>
```

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

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

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

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

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

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

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

746

747

748 3.6 Response with a new symmetric key of a specific **KeyClass**

749 After an SKS server has performed its operations of authenticating the request, identifying the requester and determining whether the requester is authorized to receive a symmetric key of the requested **KeyClass**, the SKS server generates, escrows, encrypts and sends back the symmetric key using a **SymkeyResponse** message secured within a WSS envelope. This **SymkeyResponse** is identical to the message described in Section 3.2 (since the symmetric key identified there is of the **KeyClass** requested here) and is therefore, not repeated here for brevity.

755 3.7 Request for multiple new symmetric keys

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

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

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

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

```

776 [g01] <ekmi:SymkeyRequest
777 [g02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
778 [g03]     <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
779 [g04]     <ekmi:KeyClasses>
780 [g05]         <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
781 [g06]         <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
782 [g07]         <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
783 [g08]         <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
784 [g09]         <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
785 [g10]         <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
786 [g11]         <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
787 [g12]         <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
788 [g13]         <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>
789 [g14]     </ekmi:KeyClasses>
790 [g15] </ekmi:SymkeyRequest>

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

824 3.8 Response with multiple new symmetric keys

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

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

```
835 [h01] <ekmi:SymkeyResponse
836 [h02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
837 [h03]     xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
838 [h04]     <ekmi:Symkey>
839 [h05]         <ekmi:GlobalKeyID>10514-4-3792</ekmi:GlobalKeyID>
840 [h06]         <ekmi:KeyUsePolicy>
841 [h07]             <ekmi:KeyUsePolicyID>10514-9</ekmi:KeyUsePolicyID>
842 [h08]             <ekmi:PolicyName>DES-EDE KeyUsePolicy for EHR-CDC</ekmi:PolicyName>
843 [h09]             <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
844 [h10]             <ekmi:KeyAlgorithm>
845 [h11]                 http://www.w3.org/2001/04/xmlenc#tripleDES-cbc
846 [h12]             </ekmi:KeyAlgorithm>
847 [h13]             <ekmi:KeySize>192</ekmi:KeySize>
848 [h14]             <ekmi>Status>Active</ekmi>Status>
849 [h15]             <ekmi:Permissions>
850 [h16]                 <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
851 [h17]                 <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
852 [h18]                 <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
853 [h19]                 <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
854 [h20]                 <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
855 [h21]                 <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
856 [h22]                 <ekmi:PermittedNumberOfTransactions
857 [h23]                     ekmi:any="true" xsi:nil="true"/>
858 [h24]                 <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
859 [h25]                 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
860 [h26]             </ekmi:Permissions>
861 [h27]         </ekmi:KeyUsePolicy>
862 [h28]         <ekmi:EncryptionMethod
863 [h29]             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
864 [h30]         <xenc:CipherData>
865 [h31]             <xenc:CipherValue>
866 [h32]                 E9zWB/y93hVSzeTLiDcQoDxmLNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJMa1w
867 [h33]                 hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfglpU8tlWtx27STRcR/2fw2ava
868 [h34]                 UlWtx27STRcRJMtaGhtXuLlWtx27STRcRpIsY=
869 [h35]             </xenc:CipherValue>
870 [h36]         </xenc:CipherData>
871 [h37]     </ekmi:Symkey>
872 [h38]     <ekmi:Symkey>
873 [h39]         <ekmi:GlobalKeyID>10514-4-3793</ekmi:GlobalKeyID>
874 [h40]         <ekmi:KeyUsePolicy>
875 [h41]             <ekmi:KeyUsePolicyID>10514-12</ekmi:KeyUsePolicyID>
876 [h42]             <ekmi:PolicyName>DES-EDE KeyUsePolicy for EHR-CRO</ekmi:PolicyName>
877 [h43]             <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
```

```

878 [h44]         <ekmi:KeyAlgorithm>
879 [h45]         http://www.w3.org/2001/04/xmlenc#tripledes-cbc
880 [h46]         </ekmi:KeyAlgorithm>
881 [h47]         <ekmi:KeySize>192</ekmi:KeySize>
882 [h48]         <ekmi:Status>Active</ekmi:Status>
883 [h49]         <ekmi:Permissions>
884 [h50]         <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
885 [h51]         <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
886 [h52]         <ekmi:PermittedDate>
887 [h53]         <ekmi:StartDate>2008-01-01</ekmi:StartDate>
888 [h54]         <ekmi:EndDate>2009-12-31</ekmi:EndDate>
889 [h55]         </ekmi:PermittedDate>
890 [h56]         </ekmi:PermittedDates>
891 [h57]         <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
892 [h58]         <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
893 [h59]         <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
894 [h60]         <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
895 [h61]         <ekmi:PermittedNumberOfTransactions
896 [h62]         ekmi:any="true" xsi:nil="true"/>
897 [h63]         <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
898 [h64]         <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
899 [h65]         </ekmi:Permissions>
900 [h66]     </ekmi:KeyUsePolicy>
901 [h67]     <ekmi:EncryptionMethod
902 [h68]         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
903 [h69]     <xenc:CipherData>
904 [h70]         <xenc:CipherValue>
905 [h71]         qUiXG0ca8EU871zBoXBjDoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJM1
906 [h72]         hQhywCx9sfYjv9h5FDqUiXG0ca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/2fw1
907 [h73]         UJow/qimxi8+ huUYJMtaGhtXuLWtx27STRcRpIsY=
908 [h74]         </xenc:CipherValue>
909 [h75]     </xenc:CipherData>
910 [h76] </ekmi:Symkey>
911 [h77] <ekmi:Symkey>
912 [h78]     <ekmi:GlobalKeyID>10514-4-3795</ekmi:GlobalKeyID>
913     ...
914 [h79]     <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
915     ...
916 [h80] </ekmi:Symkey>
917 [h81] <ekmi:Symkey>
918 [h82]     <ekmi:GlobalKeyID>10514-4-3797</ekmi:GlobalKeyID>
919     ...
920 [h83]     <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
921     ...
922 [h84] </ekmi:Symkey>
923 [h85] <ekmi:Symkey>
924 [h86]     <ekmi:GlobalKeyID>10514-4-3798</ekmi:GlobalKeyID>
925     ...
926 [h87]     <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
927     ...
928 [h88] </ekmi:Symkey>
929 [h89] <ekmi:Symkey>
930 [h90]     <ekmi:GlobalKeyID>10514-4-3799</ekmi:GlobalKeyID>
931     ...
932 [h91]     <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
933     ...
934 [h92] </ekmi:Symkey>
935 [h93] <ekmi:Symkey>
936 [h94]     <ekmi:GlobalKeyID>10514-4-3801</ekmi:GlobalKeyID>
937     ...

```

```

938 [h95]          <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
939          ...
940 [h96]          </ekmi:Symkey>
941 [h97]          <ekmi:Symkey>
942 [h98]          <ekmi:GlobalKeyID>10514-4-3803</ekmi:GlobalKeyID>
943          ...
944 [h99]          <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
945          ...
946 [h100]         </ekmi:Symkey>
947 [h101]         <ekmi:Symkey>
948 [h102]         <ekmi:GlobalKeyID>10514-4-3805</ekmi:GlobalKeyID>
949          ...
950 [h103]         <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>
951          ...
952 [h104]         </ekmi:Symkey>
953 [h105]         </ekmi:SymkeyResponse>

```

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

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

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

959 **[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.

963 **[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.

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

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

969 **[h09]** identifies the **KeyClass** to which this symmetric key belongs. In the case of this example, the first symmetric key in the response conforms to the **EHR-CDC** class which, presumably, might be a key that covers data encrypted for/by the Center for Disease Control (CDC) within an Electronic Health Record (EHR) system. 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.

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

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

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

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

982 **[h14]** indicates the **Status** of this **KeyUsePolicy** and whether it is an active policy or not. This is useful in situations where an application may wish to re-use a symmetric key to encrypt related data to the data originally encrypted with the symmetric key. While it is possible for the symmetric key object to be active

985 in the database, it is conceivable that the **KeyUsePolicy** used by the key has changed and the
986 application technically needs to use a new symmetric key to encrypt new data.

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

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

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

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

1000 **[h19]** is the start and end of the null **PermittedDuration** element. This implies that there are no
1001 restrictions to how long this symmetric key may be used.

1002 **[h20]** is the start and end of the null **PermittedLevels** element. This implies that there are no restrictions
1003 on the MLS security level in which this symmetric key can be used.

1004 **[h21]** is the start and end of the null **PermittedLocations** element. This implies that there are no
1005 geophysical restrictions where this symmetric key can be used.

1006 **[h22] - [h23]** is the start and end of the null **PermittedNumberOfTransactions** element. This implies that
1007 there are no restrictions on how many encryption transactions that can be performed by this symmetric
1008 key.

1009 **[h24]** is the start and end of the null **PermittedTimes** element. This implies that there are no time-of-day
1010 restrictions when this symmetric key can be used.

1011 **[h25]** is the start and end of the null **PermittedUses** element. This implies that there are no restrictions
1012 how this symmetric key can be used by applications.

1013 **[h26]** is the closing tag of the **Permissions** element.

1014 **[h27]** is the closing tag of the **KeyUsePolicy** element.

1015 **[h28]** and **[h29]** identify the encryption algorithm used in the **EncryptionMethod** element to encrypt the
1016 symmetric encryption key itself, to transport to the requesting client. The symmetric key is encrypted
1017 using the PublicKey or the requesting client. The **Algorithm** attribute uses the W3C-specified URLs for
1018 identifying the encryption and padding algorithms.

1019 **[h30]** is the start of the **CipherData** element. This element is from the W3C XML Encryption namespace
1020 (as identified by the “xenc” qualifier in the element name).

1021 **[h31]** is the start of the **CipherValue** element. This element contains the Base64-encoded ciphertext of
1022 the symmetric encryption key.

1023 **[h32] – [h34]** is the Base64-encoded ciphertext of the symmetric encryption key.

1024 **[h35]** is the closing tag of the **CipherValue** element.

1025 **[h36]** is the closing tag of the **CipherData** element.

1026 **[h37]** is the closing tag of the first **Symkey** element within this **SymkeyResponse**.

1027 [h38] - [h76] represents the **second Symkey** element in this **SymkeyResponse**. The differences in this
1028 symmetric key element from the first, can be summarized as follows:

- 1029 • [h39] identifies a different **GlobalKeyID** (10514-4-3793) for this symmetric key;
- 1030 • [h41] identifies a different **KeyUsePolicy** (10514-12) for this symmetric key;
- 1031 • [h43] identifies a different **KeyClass** (EHR-CRO) for this symmetric key;
- 1032 • [h49] - [h65] defines a different **Permissions** element for this symmetric key;
- 1033 • [h71] - [h73] contains a different **CipherValue** for this symmetric key;

1034 [h78] – [h80] is the container for the **third Symkey** element in this response. For the sake of brevity, all
1035 the usual **Symkey** elements have been dispensed with, but the unique **GlobalKeyID** and **KeyClass** are
1036 shown to indicate the SKS server's response to the request. In this example, they are 10514-4-3795 and
1037 EHR-DEF respectively.

1038 [h81] – [h104] contain the remaining **Symkey** elements of this **SymkeyResponse**. Once again, for
1039 brevity, all the details of the **Symkey** elements are dispensed with, except for the unique GKIDs and
1040 KeyClasses.

1041 [h105] is the closing tag of the **SymkeyResponse** element.

1042 Note that it is possible for a request to contain the same **KeyClass** multiple times; there is no requirement
1043 that they need to be unique within a request if an application has a legitimate business need for multiple
1044 symmetric keys of the same **KeyClass**. The SKS server will respond with unique symmetric keys, all
1045 belonging to the **KeyClass** requested by the client application.

1046 3.9 Response with an SKS error

1047 While one hopes that all authorized requesters will get favorable responses from the SKS server, there
1048 are situations in which the client application can receive an error to a request for a symmetric key. The
1049 following XML shows one example of such an error response. Depending on the type of error, the actual
1050 message content might be different.

```
1051 [i01] <ekmi:SymkeyResponse  
1052 [i02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'  
1053 [i03]     xmlns:xenc='http://www.w3.org/2001/04/xmldsig#'>  
1054 [i04]     <ekmi:SymkeyError>  
1055 [i05]         <ekmi:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>  
1056 [i06]         <ekmi:RequestedKeyClass>Payroll</ekmi:RequestedKeyClass>  
1057 [i07]         <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>  
1058 [i08]         <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>  
1059 [i09]     </ekmi:SymkeyError>  
1060 [i10] </ekmi:SymkeyResponse>
```

1061 [i01] is the start of the **SymkeyResponse** element.

1062 [i02] and [i03] identify the namespaces to which this XML conforms, and the location of their XML
1063 Schema Definitions (XSD).

1064 [i04] is the start of the **SymkeyError** element, which tells the Symmetric Key Client Library (SKCL) that
1065 the request for a symmetric key resulted in an error.

1066 [i05] indicates the **RequestedGlobalKeyID** the client requested. Returning the GKID in the error
1067 response allows the SKCL to associate the error message with the requesting application on the client
1068 machine.

1069 [i06] indicates the *RequestedKeyClass* the client requested. Returning the key-class in the error
1070 response allows the SKCL to associate the error message with the requesting application on the client
1071 machine.

1072 [i07] is an **ErrorCode** returned by the SKS server. The code may be one of the standard error codes
1073 defined in this specification, or may be a vendor-specific error code.

1074 [i08] is the text of the **ErrorMessage**, localized to the region and language of the requesting client
1075 application.

1076 [i09] is the closing tag of the **SymkeyError** tag.

1077 [i10] is the closing tag of the **SymkeyResponse** tag.

1078

1079 3.10 Response with symmetric keys and errors

1080 When a client application requests multiple symmetric keys, the SKS server may respond in one of three
1081 ways. The SKS server may:

- 1082 i. Return all symmetric keys as requested;
- 1083 ii. Return no symmetric keys – i.e. it returns all errors;
- 1084 iii. Return some symmetric keys and some errors.

1085 The following SKSML shows the third case, where the server returns some symmetric keys and errors in
1086 response to a request for multiple keys (such as the one shown in **Section 3.7 Request for multiple**
1087 **new symmetric keys**).

1088 In a response that contains a mix of symmetric keys and errors, all symmetric keys precede all errors –
1089 i.e. the **SymkeyResponse** element will not consist of **Symkeys** interspersed with **SymkeyErrors** in
1090 between; all **Symkeys** (if any) will start from the top of the response till the first **SymkeyError** element.

```
1091 [j01] <ekmi:SymkeyResponse
1092 [j02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1093 [j03]     xmlns:xenc='http://www.w3.org/2001/04/xmenc#'>
1094 [j04]   <ekmi:Symkey>
1095 [j05]     <ekmi:GlobalKeyID>10514-4-3792</ekmi:GlobalKeyID>
1096 [j06]     <ekmi:KeyUsePolicy>
1097 [j07]       <ekmi:KeyUsePolicyID>10514-9</ekmi:KeyUsePolicyID>
1098 [j08]       <ekmi:PolicyName>DES-EDE Policy for EHR-CDC</ekmi:PolicyName>
1099 [j09]       <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
1100 [j10]       <ekmi:KeyAlgorithm>
1101 [j11]         http://www.w3.org/2001/04/xmenc#tripleDES-cbc
1102 [j12]       </ekmi:KeyAlgorithm>
1103 [j13]     <ekmi:KeySize>192</ekmi:KeySize>
1104 [j14]     <ekmi:Status>Active</ekmi:Status>
1105 [j15]     <ekmi:Permissions>
1106 [j16]       <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
1107 [j17]       <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
1108 [j18]       <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
1109 [j19]       <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
1110 [j20]       <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
1111 [j21]       <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
1112 [j22]       <ekmi:PermittedNumberOfTransactions
1113 [j23]         ekmi:any="true" xsi:nil="true"/>
1114 [j24]       <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
1115 [j25]       <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
```

```

1116 [j26]         </ekmi:Permissions>
1117 [j27]     </ekmi:KeyUsePolicy>
1118 [j28]     <ekmi:EncryptionMethod
1119 [j29]         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
1120 [j30]     <xenc:CipherData>
1121 [j31]         <xenc:CipherValue>
1122 [j32]             E9zWB/y93hVSzeTLiDcQoDxmLNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJ
1123 [j33]             hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfg1pU8tLWtx27STRcR/2fw
1124 [j34]             UlWtx27STRcRJMtaGHtXuLlWtx27STRcRpIsY=
1125 [j35]         </xenc:CipherValue>
1126 [j36]     </xenc:CipherData>
1127 [j37] </ekmi:Symkey>
1128 [j38] <ekmi:Symkey>
1129 [j39]     <ekmi:GlobalKeyID>10514-4-3793</ekmi:GlobalKeyID>
1130 [j40]     <ekmi:KeyUsePolicy>
1131 [j41]         <ekmi:KeyUsePolicyID>10514-12</ekmi:KeyUsePolicyID>
1132 [j42]         <ekmi:PolicyName>DES-EDE Policy for EHR-CRO</ekmi:PolicyName>
1133 [j43]         <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
1134 [j44]         <ekmi:KeyAlgorithm>
1135 [j45]             http://www.w3.org/2001/04/xmlenc#tripleDES-cbc
1136 [j46]         </ekmi:KeyAlgorithm>
1137 [j47]         <ekmi:KeySize>192</ekmi:KeySize>
1138 [j48]         <ekmi:Status>Active</ekmi:Status>
1139 [j49]         <ekmi:Permissions>
1140 [j50]             <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
1141 [j51]             <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
1142 [j52]                 <ekmi:PermittedDate>
1143 [j53]                     <ekmi:StartDate>2008-01-01</ekmi:StartDate>
1144 [j54]                     <ekmi:EndDate>2009-12-31</ekmi:EndDate>
1145 [j55]                 </ekmi:PermittedDate>
1146 [j56]             </ekmi:PermittedDates>
1147 [j57]             <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
1148 [j58]             <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
1149 [j59]             <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
1150 [j60]             <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
1151 [j61]             <ekmi:PermittedNumberOfTransactions
1152 [j62]                 ekmi:any="true" xsi:nil="true"/>
1153 [j63]             <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
1154 [j64]             <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
1155 [j65]         </ekmi:Permissions>
1156 [j66]     </ekmi:KeyUsePolicy>
1157 [j67]     <ekmi:EncryptionMethod
1158 [j68]         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
1159 [j69]     <xenc:CipherData>
1160 [j70]         <xenc:CipherValue>
1161 [j71]             qUiQXG0ca8EU871zBoXBjDoDxmLNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCF
1162 [j72]             hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/
1163 [j73]             UJow/qimxi8+ huUYJMtaGHtXuLlWtx27STRcRpIsY=
1164 [j74]         </xenc:CipherValue>
1165 [j75]     </xenc:CipherData>
1166 [j76] </ekmi:Symkey>
1167 [j77] <ekmi:Symkey>
1168 [j78]     <ekmi:GlobalKeyID>10514-4-3795</ekmi:GlobalKeyID>
1169 [j79]         ...
1170 [j79]         <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
1171 [j79]         ...
1172 [j80] </ekmi:Symkey>
1173 [j81] <ekmi:Symkey>
1174 [j82]     <ekmi:GlobalKeyID>10514-4-3797</ekmi:GlobalKeyID>
1175 [j82]         ...

```

```

1176 [j83]          <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
1177          ...
1178 [j84]      </ekmi:Symkey>
1179 [j85]      <ekmi:Symkey>
1180 [j86]          <ekmi:GlobalKeyID>10514-4-3798</ekmi:GlobalKeyID>
1181          ...
1182 [j87]          <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
1183          ...
1184 [j88]      </ekmi:Symkey>
1185 [j89]      <ekmi:Symkey>
1186 [j90]          <ekmi:GlobalKeyID>10514-4-3799</ekmi:GlobalKeyID>
1187          ...
1188 [j91]          <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
1189          ...
1190 [j92]      </ekmi:Symkey>
1191 [j93]      <ekmi:Symkey>
1192 [j94]          <ekmi:GlobalKeyID>10514-4-3801</ekmi:GlobalKeyID>
1193          ...
1194 [j95]          <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
1195          ...
1196 [j96]      </ekmi:Symkey>
1197 [j97]      <ekmi:SymkeyError>
1198 [j98]          <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
1199 [j99]          <ekmi:RequestedKeyClass>EHR-PAT</ekmi:RequestedKeyClass>
1200 [j100]         <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
1201 [j101]         <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
1202 [j102]      </ekmi:SymkeyError>
1203 [j103]      <ekmi:SymkeyError>
1204 [j104]         <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
1205 [j105]         <ekmi:RequestedKeyClass>EHR-PHY</ekmi:RequestedKeyClass>
1206 [j106]         <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
1207 [j107]         <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
1208 [j108]      </ekmi:SymkeyError>
1209 [j109] </ekmi:SymkeyResponse>

```

1210 [j01] – [j96] is no different from the response shown in **Section 3.8 Response with multiple new**
1211 **symmetric keys**.

1212 [j97] - [j102] identifies the first **SymkeyError** returned by the SKS server. It is not unlike the error
1213 described in **Section 3.9 Response with an SKS error**. However, there is one difference in this element
1214 on line [j79]. This element includes the **RequestedKeyClass** of the request.

1215 [j103] - [j108] identifies the second **SymkeyError** returned by the SKS server. It is similar to the error
1216 described in lines [j77] through [j82] including the **RequestedKeyClass** of the request.

1217 [j109] is the closing tag of the **SymkeyResponse** tag.

1218 3.11 Request for a symmetric key-caching policy

1219 When a client application (that has been linked to the SKCL) needs to encrypt sensitive data, it will call an
1220 API method within the SKCL for a new symmetric key. After the SKCL has ensured that the application is
1221 authorized to make such a request (by verifying that the configured/passed-in credentials can access the
1222 cryptographic key-store module on the client containing the PrivateKey used for signing SKSML
1223 requests), the SKCL assembles the following SKSML request:

```

1224 [k01] <ekmi:KeyCachePolicyRequest
1225 [k02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01"/>

```

1226 [k01] is the start of the **KeyCachePolicyRequest** element.

1227 **[k02]** identifies the namespace to which this XML conforms, and the location of its XML Schema
1228 Definition (XSD).

1229 The **KeyCachePolicyRequest** is an “empty” request. It does not need to specify any requesting
1230 parameter or element, since the SKS server only needs to know who requested the policy. The server
1231 derives this information from the SOAP Header and consequently has everything it needs to know – the
1232 digital signature to establish the identity of the requester, as well as to ensure message integrity in the
1233 request.

1234 While the **KeyCachePolicyRequest** element is very simple, the Web Service Security (WSS) envelope –
1235 which provides security for all SKSML messages – expands the size of the message. The same request
1236 shown above, is displayed below in its entirety, with its WSS envelope. Please note that some content –
1237 such as Base64-encoded binary content - has been reformatted for aesthetics and clarity of the XML
1238 elements. The actual elements and data-types have been preserved from actual SKSML messages.

1239 For an interpretation of the XML elements shown below, please refer to **[WSS]**.

1240 For the sake of brevity, this specification will dispense with showing the SOAP envelope and the WSS
1241 elements in all other examples, when discussing SKSML.

```
1242 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
1243   <SOAP-ENV:Header>
1244     <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1245     1.0.xsd" SOAP-ENV:mustUnderstand="1">
1246       <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
1247       utility-1.0.xsd"
1248         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1249         1.0#Base64Binary"
1250         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
1251         wsu:id="XWSSGID-1172790302111-1738806553">
1252           MIIDfDCCAmSgAwIBAgIIAe/AvliGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdU3Ryb25nab
1253           S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvciBTdHJvbmddLZXkgREVNTyBVc2Ug1d
1254           T25seTEXMBUGA1UEChMOU3Ryb25nQXV0aCBJbmMwHhcNMDYwNz11MTcxMDMwWhcNMDcwNzla64dd3k
1255           A1UECXMbRm9yIFN0cm9uZ0tleSBERU1PIFVzZSBPbmx5MRcwFQYDVQQKEW5TdHJvbmddBdXRolEI2da
1256           S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvciBTdHJvbmddLZXkgREVNTyBVc2Ugia
1257           T25seTEXMBUGA1UEChMOU3Ryb25nQXV0aCBJbmMwHhcNMDYwNz11MTY0NjEwWkcNMDcwNz1s34wdd
1258           NjEwWjBpMREwDwYKZCZlmiZPYLQGBARMBOTEVMBMGA1UEAxMUMU0tTIFNlcnZlci0xMSQwWlgYDVQsdsW2
1259           ExtGb3IgdU3Ryb25nS2V5IERFTU8gVXNlIE9ubHkxZzAvbG9uZ0t1dGggSW5jMlIBd2
1260           NBGkqhkiG9w0BAQEFAAOCQA8AMIIBCgKCAQEAztpqRoU5A8plxx1Rz1QEUnIAAM1D5g9+islr3wxa
1261           hbwtjFSMYilnY4iV77xU/nsMOnMZ7RxsLYkDcZQ1ODVYqQwqmAvaJ5Z6SVy34gZ51YG+rSWE3NjFsd
1262           bOXW8RJA/Tn6Lmht/qngrcaqgmtP0cAAiMRZOWtCTmC2K/LEqDabXSyU6Hh8ySNE3njyvmWpresf
1263           zsYokTdnvWQqT6tKo1OwJsdJ1+hxM7DrnMLvMNq5relNfsKhDdX17wzhrBUx+hiYA/qo8tMXkL6wsd
1264           4PN5dYugtzpSzldUO5tlg58Avhzw07hy5oofBIKFY22CeljQ36u0bMjuyGj6UYHs3rdfsds32rda
1265           YzCBnzANBkgqhkiG9w0BAQEFAAQBQAwgYkCgYEAyAmxMZhYA8wHJ4UE4b61s51JVWe4Fygj4Mcf3a
1266           hvcNAQELBQADggEBACK05PtvZD4WPgIOe=
1267       </wsse:BinarySecurityToken>
1268       <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
1269         <ds:SignedInfo>
1270           <ds:CanonicalizationMethod
1271             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1272             <InclusiveNamespaces
1273               xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"
1274               PrefixList="wsse SOAP-ENV"/>
1275             </ds:CanonicalizationMethod>
1276             <ds:SignatureMethod
1277               Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
1278               <ds:Reference URI="#XWSSGID-1172790300636-653454040">
1279                 <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1280                 <ds:DigestValue>IU4m+rp4oebgl9g+t3nRaZYqUIE=</ds:DigestValue>
1281                 </ds:Reference>
1282                 <ds:Reference URI="#XWSSGID-1172790300637708871805">
1283                 <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
```

```

1284         <ds:DigestValue>WCpOmTCbffcEHXhGf5rIEYWLRZg=</ds:DigestValue>
1285     </ds:Reference>
1286 </ds:SignedInfo>
1287 <ds:SignatureValue>
1288 svStAvBRRrF+g2biPI7uWHkJTQPII8t4phMbOZQsZlQcn36tcMSj/a4+4LPNfOB3Y8yO2lr1Oa1
1289 fGqCPAWZNuEH34VQEM196rRwV258mpg8uwpXEYJlgPJqg89w8+/NdaODccLQ2Bizu7QM/HSM2ab
1290 ogNJwqmbSylazOsnOcU=
1291 </ds:SignatureValue>
1292 <ds:KeyInfo>
1293 <wsse:SecurityTokenReference xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1294 wss-wssecurity-utility-1.0.xsd"
1295 wsu:Id="XWSSGID-1172790300633-442423344">
1296 <wsse:Reference URI="#XWSSGID-1172790302111-1738806553"
1297 ValueURI="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
1298 profile-1.0#X509v3"/>
1299 </wsse:SecurityTokenReference>
1300 </ds:KeyInfo>
1301 </ds:Signature>
1302 <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1303 1.0.xsd" wsu:Id="XWSSGID-1172790300637708871805">
1304 <wsu:Created>2007-03-01T23:05:00Z</wsu:Created>
1305 <wsu:Expires>2007-03-01T23:05:05Z</wsu:Expires>
1306 </wsu:Timestamp>
1307 </wsse:Security>
1308 </SOAP-ENV:Header>
1309 <SOAP-ENV:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1310 1.0.xsd" wsu:Id="XWSSGID-1172790300636-653454040">
1311 <ekmi:KeyCachePolicyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01"/>
1312 </SOAP-ENV:Body>
1313 </SOAP-ENV:Envelope>

```

1314 3.12 Response with a symmetric key-caching policy (1)

1315 After an SKS server has performed its operations of authenticating a **KeyCachePolicyRequest**,
1316 identifying the requester, determining policies that apply to the requester, it assembles the following
1317 response and returns it to the client. (The SOAP message, as indicated earlier, is secured using WSS,
1318 but only the actual SKSMML content is displayed and discussed here).

```

1319 [m01] <ekmi:KeyCachePolicyResponse
1320 [m02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1321 [m03]     xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
1322 [m04]     <ekmi:KeyCachePolicy>
1323 [m05]         <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
1324 [m06]         <ekmi:PolicyName>No Caching Policy</ekmi:PolicyName>
1325 [m07]         <ekmi:Description>
1326 [m08]             This policy is for high-risk, always-connected machines on the
1327 [m09]             network, which will never cache symmetric keys locally. This
1328 [m10]             policy never expires (but checks monthly for any updates).
1329 [m11]         </ekmi:Description>
1330 [m12]         <ekmi:KeyClass>NoCachingClass</ekmi:KeyClass>
1331 [m13]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1332 [m14]         <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
1333 [m15]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1334 [m16]         <ekmi:Status>Active</ekmi:Status>
1335 [m17]     </ekmi:KeyCachePolicy>
1336 [m18] </ekmi:KeyCachePolicyResponse>

```

1337 [m01] is the start of the **KeyCachePolicyResponse** element.

1338 [m02] and [m03] identify the namespaces to which this XML conforms, and the location of their XML
1339 Schema Definitions (XSD).

1340 [m04] is the start of the first – and only - **KeyCachePolicy** element.

1341 [m05] identifies the **KeyCachePolicyID** (KCPID) assigned to this policy by the SKS server. In this
 1342 example, the concatenated values of the Domain ID and Policy ID indicate that the key belongs to the
 1343 organization represented by the PEN, 10514; and is the first key-caching policy within the SKMS.

1344 [m06] provides a descriptive name for this key-cache policy through the **PolicyName** element, which is
 1345 helpful to human readers when identifying this policy.

1346 [m07] is the start tag of the **Description** element.

1347 [m08] - [m10] provides a human-readable description about this key-cache policy.

1348 [m11] is the closing tag of the **Description** element.

1349 [m12] specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are
 1350 subject to this caching policy.

1351 [m13] specifies the date and time that this **KeyCachePolicy** is effective. This is accomplished through a
 1352 **StartDate** element. In this example, the policy is effective as of January 01, 2008.

1353 [m14] specifies the date and time that this **KeyCachePolicy** becomes invalid. This is accomplished
 1354 through a **EndDate** element. In this example, the use of the UNIX “epoch” date (January 01, 1969)
 1355 indicates that this policy never expires.

1356 [m15] specifies the frequency at which this client must check with the SKS server for updates to the key-
 1357 caching policy. This is specified in seconds in the **PolicyCheckInterval** element; in this example it is a
 1358 monthly interval.

1359 [m16] indicates the **Status** of this **KeyCachePolicy** and whether it is an active policy or not.

1360 [m17] is the closing tag of the **KeyCachePolicy** element.

1361 [m18] is the closing tag of the **KeyCachePolicyResponse** element.

1362

1363

1364 3.13 Response with a symmetric key-caching policy (2)

1365 This is a second example of a key-caching policy response that has additional elements in the policy
 1366 permitting caching and specify the number of unused and used (for encryption) symmetric keys that may
 1367 be cached by the client machine.

```

1368 [m01] <ekmi:KeyCachePolicyResponse
1369 [m02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1370 [m03]     xmlns:xenc='http://www.w3.org/2001/04/xmlenc#' >
1371 [m04]   <ekmi:KeyCachePolicy>
1372 [m05]     <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
1373 [m06]     <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
1374 [m07]     <ekmi:Description>
1375 [m08]       This policy defines how company-issued laptops will manage symmetric
1376 [m09]       keys used for file/disk encryption in their local cache. This
1377 [m10]       policy must be used by all laptops that use the company EKMI.
1378 [m11]     </ekmi:Description>
1379 [m12]     <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
1380 [m13]     <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1381 [m14]     <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
1382 [m15]     <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1383 [m16]     <ekmi>Status>Active</ekmi>Status>

```



```

1384 [m17]      <ekmi:NewKeysCacheDetail>
1385 [m18]      <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1386 [m19]      <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1387 [m20]      </ekmi:NewKeysCacheDetail>
1388 [m21]      <ekmi:UsedKeysCacheDetail>
1389 [m22]      <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1390 [m23]      <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1391 [m24]      </ekmi:UsedKeysCacheDetail>
1392 [m25]      </ekmi:KeyCachePolicy>
1393 [m26] </ekmi:KeyCachePolicyResponse>

```

1394 [m01] is the start of the **KeyCachePolicyResponse** element.

1395 [m02] and [m03] identify the namespaces to which this XML conforms, and the location of their XML Schema Definitions (XSD).

1397 [m04] is the start of the first – and only - **KeyCachePolicy** element.

1398 [m05] identifies the **KeyCachePolicyID** (KCPID) assigned by the SKS server for the key-caching policy being returned.

1400 [m06] provides a descriptive name for this key-cache policy through the **PolicyName** element, which is helpful to human readers when identifying this policy.

1402 [m07] is the start tag of the **Description** element.

1403 [m08] - [m10] provides a human-readable description about this key-cache policy.

1404 [m11] is the closing tag of the **Description** element.

1405 [m12] specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are subject to this caching policy.

1407 [m13] specifies the date and time that this **KeyCachePolicy** is effective. This is accomplished through a **StartDate** element. In this example, the policy is effective as of January 01, 2008.

1409 [m14] specifies the date and time that this **KeyCachePolicy** becomes invalid. This is accomplished through a **EndDate** element. In this example, the policy expires on December 31, 2008.

1411 [m15] specifies the frequency at which this client must check with the SKS server for updates to the key-caching policy. This is specified in seconds in the **PolicyCheckInterval** element; in this example it is a monthly interval.

1414 [m16] indicates the **Status** of this **KeyCachePolicy** and whether it is an active policy or not.

1415 [m17] is the start of the **NewKeysCacheDetail** element, which provides details about how many new symmetric keys – that haven't been used for any encryption transactions – may be cached by the client and for how long.

1418 [m18] indicates the maximum number of new (unused) symmetric keys that may be cached by the client. This is specified through the **MaximumKeys** element. When the client uses a symmetric key, this reduces the number of new symmetric keys. In this case, the SKCL connects to the SKS server (if it is on the network) and requests a new symmetric key to add to its new-key cache.

1422 [m19] indicates the maximum duration that new (unused) symmetric keys may be cached by the client. This is specified through the **MaximumDuration** element in seconds. If there are any new keys that exceed this duration limit, the SKCL deletes the specific symmetric key and replaces it with a new symmetric key from the SKS server.

1426 [m20] is the closing tag of the **NewKeysCacheDetail** element.

1427 [m21] is the start of the **UsedKeysCacheDetail** element, which provides details about how many used
1428 symmetric keys – those that HAVE been used for any encryption transactions – may be cached by the
1429 client and for how long.

1430 [m22] indicates the maximum number of used symmetric keys that may be cached by the client through
1431 the **MaximumKeys** element. If the client already has the maximum number of used-keys in its cache,
1432 using the First-In-First-Out (FIFO) method, it deletes the oldest symmetric key in the cache to replace with
1433 the key that transitioned from the “new” to “used” status.

1434 [m23] indicates the maximum duration that used symmetric keys may be cached by the client through the
1435 **MaximumDuration** element in seconds. If there are any used keys that exceed this duration limit, the
1436 SKCL deletes the specific symmetric key. While this may temporarily reduce the number of used
1437 symmetric keys in the cache, the SKCL takes the most conservative position when making this decision.

1438 [m24] is the closing tag of the **UsedKeysCacheDetail** element.

1439 [m25] is the closing tag of the **KeyCachePolicy** element.

1440 [m26] is the closing tag of the **KeyCachePolicyResponse** element.

1441

1442 3.14 Response with multiple symmetric key-caching policies (3)

1443 This is a third example of a key-caching policy response that has multiple key-cache policies that apply to
1444 different classes of symmetric keys.

```

1445 [m01] <ekmi:KeyCachePolicyResponse
1446 [m02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1447 [m03]     xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
1448 [m04]     <ekmi:KeyCachePolicy>
1449 [m05]         <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
1450 [m06]         <ekmi:PolicyName>No Caching Policy</ekmi:PolicyName>
1451 [m07]         <ekmi:Description>
1452 [m08]             This policy is for high-risk, always-connected machines on the
1453 [m09]             network, which will never cache symmetric keys locally. This
1454 [m10]             policy never expires (but checks monthly for any updates).
1455 [m11]         </ekmi:Description>
1456 [m12]         <ekmi:KeyClass>NoCachingClass</ekmi:KeyClass>
1457 [m13]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1458 [m14]         <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
1459 [m15]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1460 [m16]         <ekmi:Status>Active</ekmi:Status>
1461 [m17]     </ekmi:KeyCachePolicy>
1462 [m18]     <ekmi:KeyCachePolicy>
1463 [m19]         <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
1464 [m20]         <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
1465 [m21]         <ekmi:Description>
1466 [m22]             This policy defines how company-issued laptops will manage symmetric
1467 [m23]             keys used for file/disk encryption in their local cache. This
1468 [m24]             policy must be used by all laptops that use the company EKMI.
1469 [m25]         </ekmi:Description>
1470 [m26]         <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
1471 [m27]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1472 [m28]         <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
1473 [m29]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1474 [m30]         <ekmi:Status>Active</ekmi:Status>
1475 [m31]         <ekmi:NewKeysCacheDetail>
1476 [m32]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1477 [m33]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>

```

```

1478 [m34]         </ekmi:NewKeysCacheDetail>
1479 [m35]         <ekmi:UsedKeysCacheDetail>
1480 [m36]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1481 [m37]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1482 [m38]         </ekmi:UsedKeysCacheDetail>
1483 [m39]     </ekmi:KeyCachePolicy>
1484 [m40] <ekmi:KeyCachePolicy>
1485 [m41]     <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
1486 [m42]     <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
1487 [m43]     <ekmi:Description>
1488 [m44]         This policy defines how company-issued laptops will manage
1489 [m45]         symmetric keys used for file/disk encryption in their local
1490 [m46]         cache. This policy must be used by all laptops.
1491 [m47]     </ekmi:Description>
1492 [m48]     <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
1493 [m49]     <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1494 [m50]     <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
1495 [m51]     <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1496 [m52]     <ekmi>Status>Active</ekmi>Status>
1497 [m53]     <ekmi:NewKeysCacheDetail>
1498 [m54]         <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1499 [m55]         <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1500 [m56]     </ekmi:NewKeysCacheDetail>
1501 [m57]     <ekmi:UsedKeysCacheDetail>
1502 [m58]         <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1503 [m59]         <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1504 [m60]     </ekmi:UsedKeysCacheDetail>
1505 [m61]     <ekmi:KeyCachePolicy>
1506 [m62] </ekmi:KeyCachePolicyResponse>

```

1507

1508 [m01] is the start of the **KeyCachePolicyResponse** element.

1509 [m02] and [m03] identify the namespaces to which this XML conforms, and the location of their XML
1510 Schema Definitions (XSD).

1511 [m04] is the start of the first of three **KeyCachePolicy** elements in this response.

1512 [m05] identifies the **KeyCachePolicyID** (KCPID) assigned to this policy by the SKS server. In this
1513 example, the concatenated values of the Domain ID and Policy ID indicate that the key belongs to the
1514 organization represented by the PEN, 10514; and is the first key-caching policy within the SKMS.

1515 [m06] provides a descriptive name for this key-cache policy through the **PolicyName** element, which is
1516 helpful to human readers when identifying this policy.

1517 [m07] is the start tag of the **Description** element.

1518 [m08] - [m10] provides a human-readable description about this key-cache policy.

1519 [m11] is the closing tag of the **Description** element.

1520 [m12] specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are
1521 subject to this caching policy.

1522 [m13] specifies the date and time that this **KeyCachePolicy** is effective. This is accomplished through a
1523 **StartDate** element. In this example, the policy is effective as of January 01, 2008.

1524 [m14] specifies the date and time that this **KeyCachePolicy** becomes invalid. This is accomplished
1525 through a **EndDate** element. In this example, the use of the UNIX "epoch" date (January 01, 1969)
1526 indicates that this policy never expires.

1527 [m15] specifies the frequency at which this client must check with the SKS server for updates to the key-
1528 caching policy. This is specified in seconds in the *PolicyCheckInterval* element; in this example it is a
1529 monthly interval.

1530 [m16] indicates the **Status** of this *KeyCachePolicy* and whether it is an active policy or not.

1531 [m17] is the closing tag of the first *KeyCachePolicy* element.

1532 [m18] is the start of the second of three *KeyCachePolicy* elements in this response.

1533 [m19] identifies the *KeyCachePolicyID* (KCPID) assigned by the SKS server for the key-caching policy
1534 being returned.

1535 [m20] provides the descriptive name for this key-cache policy through the *PolicyName* element.

1536 [m21] is the start tag of the *Description* element.

1537 [m22] - [m24] provides a human-readable description about this key-cache policy.

1538 [m25] is the closing tag of the *Description* element.

1539 [m26] specifies the *KeyClass* to which this policy applies. Only keys that belong to this key-class are
1540 subject to this caching policy.

1541 [m27] specifies the date and time that this *KeyCachePolicy* is effective. This is accomplished through a
1542 *StartDate* element. In this example, the policy is effective as of January 01, 2008.

1543 [m28] specifies the date and time that this *KeyCachePolicy* becomes invalid. This is accomplished
1544 through a *EndDate* element. In this example, the policy expires on December 31, 2008.

1545 [m29] specifies the frequency at which this client must check with the SKS server for updates to the key-
1546 caching policy. This is specified in seconds in the *PolicyCheckInterval* element; in this example it is a
1547 monthly interval.

1548 [m30] indicates the **Status** of this *KeyCachePolicy* and whether it is an active policy or not.

1549 [m31] is the start of the *NewKeysCacheDetail* element, which provides details about how many new
1550 symmetric keys – that haven't been used for any encryption transactions – may be cached by the client
1551 and for how long.

1552 [m32] indicates the maximum number of new (unused) symmetric keys that may be cached by the client.
1553 This is specified through the *MaximumKeys* element. When the client uses a symmetric key, this
1554 reduces the number of new symmetric keys. In this case, the SKCL connects to the SKS server (if it is on
1555 the network) and requests a new symmetric key to add to its new-key cache.

1556 [m33] indicates the maximum duration that new (unused) symmetric keys may be cached by the client.
1557 This is specified through the *MaximumDuration* element in seconds. If there are any new keys that
1558 exceed this duration limit, the SKCL deletes the specific symmetric key and replaces it with a new
1559 symmetric key from the SKS server.

1560 [m34] is the closing tag of the *NewKeysCacheDetail* element.

1561 [m35] is the start of the *UsedKeysCacheDetail* element, which provides details about how many used
1562 symmetric keys – those that HAVE been used for any encryption transactions – may be cached by the
1563 client and for how long.

1564 [m36] indicates the maximum number of used symmetric keys that may be cached by the client through
1565 the *MaximumKeys* element. If the client already has the maximum number of used-keys in its cache,
1566 using the First-In-First-Out (FIFO) method, it deletes the oldest symmetric key in the cache to replace with
1567 the key that transitioned from the “new” to “used” status.

1568 [m37] indicates the maximum duration that used symmetric keys may be cached by the client through the
1569 **MaximumDuration** element in seconds. If there are any used keys that exceed this duration limit, the
1570 SKCL deletes the specific symmetric key. While this may temporarily reduce the number of used
1571 symmetric keys in the cache, the SKCL takes the most conservative position when making this decision.

1572 [m38] is the closing tag of the **UsedKeysCacheDetail** element.

1573 [m39] is the closing tag of the second **KeyCachePolicy** element.

1574 [m40] is the start of the third **KeyCachePolicy** element in this response.

1575 [m41] identifies the **KeyCachePolicyID** (KCPID) assigned by the SKS server for the key-caching policy
1576 being returned.

1577 [m42] provides the descriptive name for this key-cache policy through the **PolicyName** element.

1578 [m43] is the start tag of the **Description** element.

1579 [m44] - [m46] provides a human-readable description about this key-cache policy.

1580 [m47] is the closing tag of the **Description** element.

1581 [m48] specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are
1582 subject to this caching policy.

1583 [m49] specifies the date and time that this **KeyCachePolicy** is effective.

1584 [m50] specifies the date and time that this **KeyCachePolicy** becomes invalid.

1585 [m51] specifies the frequency at which this client must check with the SKS server for updates to the key-
1586 caching policy.

1587 [m52] indicates the **Status** of this **KeyCachePolicy** and whether it is an active policy or not.

1588 [m53] is the start of the **NewKeysCacheDetail** element, which provides details about how many new
1589 symmetric keys may be cached by the client and for how long.

1590 [m54] indicates the maximum number of new (unused) symmetric keys that may be cached by the client.
1591 This is specified through the **MaximumKeys** element. When the client uses a symmetric key, this
1592 reduces the number of new symmetric keys. In this case, the SKCL connects to the SKS server (if it is on
1593 the network) and requests a new symmetric key to add to its new-key cache.

1594 [m55] indicates the maximum duration that new (unused) symmetric keys may be cached by the client.
1595 This is specified through the **MaximumDuration** element in seconds. If there are any new keys that
1596 exceed this duration limit, the SKCL deletes the specific symmetric key and replaces it with a new
1597 symmetric key from the SKS server.

1598 [m56] is the closing tag of the **NewKeysCacheDetail** element.

1599 [m57] is the start of the **UsedKeysCacheDetail** element, which provides details about how many used
1600 symmetric keys – those that HAVE been used for any encryption transactions – may be cached by the
1601 client and for how long.

1602 [m58] indicates the maximum number of used symmetric keys that may be cached by the client through
1603 the **MaximumKeys** element. If the client already has the maximum number of used-keys in its cache,
1604 using the First-In-First-Out (FIFO) method, it deletes the oldest symmetric key in the cache to replace with
1605 the key that transitioned from the “new” to “used” status.

1606 [m59] indicates the maximum duration that used symmetric keys may be cached by the client through the
1607 **MaximumDuration** element in seconds. If there are any used keys that exceed this duration limit, the
1608 SKCL deletes the specific symmetric key. While this may temporarily reduce the number of used
1609 symmetric keys in the cache, the SKCL takes the most conservative position when making this decision.

- 1610 **[m60]** is the closing tag of the *UsedKeysCacheDetail* element.
- 1611 **[m61]** is the closing tag of the third and final *KeyCachePolicy* element in this response.
- 1612 **[m62]** is the closing tag of the *KeyCachePolicyResponse* element.

1613 4 Specification

1614

1615 4.1 Element <SymkeyRequest>

1616 The <SymkeyRequest> element identifies one or more **GlobalKeyID**'s of symmetric encryption keys
1617 needed by the client application. The request may also specify one or more **KeyClass** elements for the
1618 requested key when the request is for a new symmetric key.

1619 While it is a top-level element within this specification, a <SymkeyRequest> element MAY be enclosed
1620 within a **SOAP Body** element of a **SOAP Envelope** to conform to the security requirements of this
1621 specification. The **SOAP Header** of the **SOAP Envelope** MUST enclose a **Security** element conforming
1622 to [WSS] with a **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
1623 requirements of the specified security profile in [WSS] to form a well-formed, secure message.
1624

1625 Schema Definition:

```
1626 <xsd:element name="SymkeyRequest">  
1627   <xsd:complexType>  
1628     <xsd:sequence>  
1629       <xsd:element  
1630         name="GlobalKeyID"  
1631         type="ekmi:GlobalKeyIDType"  
1632         minOccurs="1"  
1633         maxOccurs="unbounded">  
1634     </xsd:element>  
1635     <xsd:element  
1636       name="KeyClasses"  
1637       type="ekmi:KeyClassesType"  
1638       minOccurs="0">  
1639     </xsd:element>  
1640   </xsd:sequence>  
1641 </xsd:complexType>  
1642 </xsd:element>
```

1643 The <SymkeyRequest> element consists of a sequence of two child elements:

1644 1. <GlobalKeyID> [Required]

1645

1646 This element of type **GlobalKeyIDType**, identifies the unique global key identifier of the
1647 requested symmetric key within the target Symmetric Key Management System (SKMS) the client
1648 is communicating with. There MUST be at least one <GlobalKeyID> element in a
1649 <SymkeyRequest>, but there may be an unbounded (unlimited) number of <GlobalKeyID>
1650 elements specified.
1651

1652 The <GlobalKeyID> element is specified in Section 4.2.

1653 2. <KeyClasses> [Optional]

1654

1655 This element of type **KeyClassesType**, when specified, identifies at least one <KeyClass>
1656 element, but may specify an unbounded (unlimited) number of <KeyClass> elements within the
1657 <KeyClasses> set. Client applications may request one or more symmetric keys conforming to
1658 one or more key classes required by the application. If the client application is authorized to
1659 receive keys conforming to such key classes, the **SKS** server will generate and supply them.
1660

1661 When more than one <GlobalKeyID> for a new symmetric key is specified in the request, there
1662 MAY be only one <KeyClass> element within the <KeyClasses> set.
1663

1664 When the client requires more than one new symmetric key, and each key is required to be of a
1665 different key class, there MUST be only one <GlobalKeyID> element followed by as many
1666 <KeyClass> elements inside the <KeyClasses> set, as needed by the client application.
1667

1668 When a client requires multiple symmetric keys of two or more key classes, the client MUST send
1669 multiple requests to the **SKS** server. See examples 4 and 5 below in this section.
1670

1671 The <KeyClasses> and <KeyClass> elements are specified in Section 4.3.

1672 Some examples of the <SymkeyRequest> element are as follows:

1673 **Example 1 – A single new symmetric key request of a default key class:**

```
1674 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1675 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1676 </ekmi:SymkeyRequest>
```

1677 **Example 2 – A request for three new symmetric keys of a default key class for each symmetric
1678 key:**

```
1679 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1680 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1681 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1682 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1683 </ekmi:SymkeyRequest>
```

1684 **Example 3 – A request for a single new symmetric key of a specific key class:**

```
1685 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1686 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1687 <ekmi:KeyClasses>  
1688 <ekmi:KeyClass>HR-Class</ekmi:KeyClass>  
1689 </ekmi:KeyClasses>  
1690 </ekmi:SymkeyRequest>
```

1691 **Example 4 – A request for a two new symmetric keys with the same key class for each symmetric
1692 key:**

```
1693 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1694 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1695 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1696 <ekmi:KeyClasses>  
1697 <ekmi:KeyClass>FIN-FX</ekmi:KeyClass>  
1698 </ekmi:KeyClasses>  
1699 </ekmi:SymkeyRequest>
```

1700 **Example 5 – A request for a nine new symmetric keys of different key classes:**

```
1701 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1702 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1703 <ekmi:KeyClasses>  
1704 <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>  
1705 <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>  
1706 <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>  
1707 <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>  
1708 <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>  
1709 <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>  
1710 <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
```



```
1711         <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
1712         <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>
1713     </ekmi:KeyClasses>
1714 </ekmi:SymkeyRequest>
```

1715

1716

1717 4.2 Element <GlobalKeyID>

1718 The <GlobalKeyID> element is the unique identifier of a symmetric encryption key within an SKMS.
1719 Every symmetric key generated by the **SKS** server MUST be assigned a unique <GlobalKeyID> as
1720 specified in this section.

1721 Schema Definition:

```
1722     <xsd:simpleType name="GlobalKeyIDType">
1723         <xsd:restriction base="xsd:string">
1724             <xsd:minLength value="5"/>
1725             <xsd:maxLength value="62"/>
1726             <xsd:pattern value="[0-9]{1,20}-[0-9]{1,20}-[0-9]{1,20}"/>
1727             <xsd:whiteSpace value="collapse"/>
1728         </xsd:restriction>
1729     </xsd:simpleType>
```

1730 The <GlobalKeyID> element is of the **GlobalKeyIDType**, and is a string identifier of a symmetric key
1731 consisting of five parts concatenated together:

- 1732 1. A positive integer identifying the **Domain ID**. The **DomainID** identifies the IANA-issued Private
1733 Enterprise Number (PEN) as published at <http://www.iana.org/assignments/enterprise-numbers>
1734 and is used by the **SKS** server to constrain the ownership of objects within the SKMS;
- 1735 2. A literal hyphen ("-") without surrounding spaces;
- 1736 3. A positive integer identifying the Server ID of the server that originally generated the key;
- 1737 4. Another literal hyphen ("-") without surrounding spaces;
- 1738 5. A positive integer identifying the Key ID;

1739 Combined, the five components of this element make up a unique identifier for a symmetric key within the
1740 SKMS. Since all enterprise are expected to use only the PENs assigned to them, technically the
1741 <GlobalKeyID> is unique across the internet.

1742 The **DomainID** part of the <GlobalKeyID> element MUST be a positive integer in the range of 0 (zero) to
1743 18446744073709551615 (20-byte ASCII decimal).

1744 When an SKMS manages the symmetric keys for a single enterprise, the **DomainID** part of the
1745 <GlobalKeyID> element in a <SymkeyRequest> MAY be zero ("0"). When an SKMS manages symmetric
1746 keys for multiple enterprises, the **DomainID** in the <GlobalKeyID> of a <SymkeyRequest> MUST be
1747 positive and non-zero. In such a situation, the client application will request a symmetric key for the
1748 domain in which it is authorized to request and receive keys.

1749 The **DomainID** in the <GlobalKeyID> element of a <SymkeyResponse> MUST always be positive and
1750 non-zero. It will typically contain the PEN of the domain to which the symmetric key belongs.

1751 The **ServerID** part of the <GlobalKeyID> element MUST be a positive integer in the range of 0 (zero) to
1752 18446744073709551615 (20-byte ASCII decimal).

1753 The **ServerID** part of the <GlobalKeyID> element of a <SymkeyRequest> MUST always be zero ("0").

1754 The **ServerID** part of the <GlobalKeyID> element of a <SymkeyResponse> MUST always be positive and
1755 non-zero. It will typically contain the unique server identifier of the **SKS** server where the symmetric key
1756 was generated.

1757 The **KeyD** part of the <GlobalKeyID> element MUST be a positive integer in the range of 0 (zero) to
1758 18446744073709551615 (20-byte ASCII decimal).

1759 The **KeyID** part of the <GlobalKeyID> element of a <SymkeyRequest> MUST always be zero ("0").

1760 The **KeyID** part of the <GlobalKeyID> element of a <SymkeyResponse> MUST always be positive and
1761 non-zero. It will typically contain the unique key identifier of the symmetric key within the **SKS** server
1762 where the key was generated.

1763 **Example 1 – A <GlobalKeyID> value for a new symmetric key from an SKMS that serves a single**
1764 **domain:**

```
1765 <ekmi:GlobalKeyID>0-0-0</ekmi:GlobalKeyID>
```

1766 **Example 2 – A <GlobalKeyID> value for a new symmetric key for the domain with the PEN 10514,**
1767 **from an SKMS that serves multiple domains:**

```
1768 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
```

1769 **Example 3 – A <GlobalKeyID> value for the 16,777,215th symmetric key generated on 2nd SKS**
1770 **server for an enterprise with the PEN 10514, in either a <SymkeyRequest> or a <SymkeyResponse>:**

```
1771 <ekmi:GlobalKeyID>10514-2-16777215</ekmi:GlobalKeyID>
```

1772 **Example 4 – The maximum <GlobalKeyID> value possible (a 62-byte ASCII decimal), in a**
1773 **<SymkeyRequest> or <SymkeyResponse>:**

```
1774 <ekmi:GlobalKeyID>  
1775 18446744073709551615-18446744073709551615-18446744073709551615  
1776 </ekmi:GlobalKeyID>
```

1777 4.3 Element <KeyClasses> and <KeyClass>

1778 The <KeyClasses> element of type **KeyClassesType**, when specified, identifies at least one
1779 <KeyClass> element, but may specify an unbounded (unlimited) number of <KeyClass> elements within
1780 the <KeyClasses> set.

1781 Schema Definition:

```
1782 <xsd:complexType name="KeyClassesType">  
1783 <xsd:sequence>  
1784 <xsd:element  
1785 name="KeyClass"  
1786 type="tns:KeyClassType"  
1787 minOccurs="1"  
1788 maxOccurs="unbounded"/>  
1789 </xsd:sequence>  
1790 </xsd:complexType>
```

1791

```
1792 <xsd:simpleType name="KeyClassType">  
1793 <xsd:restriction base="xsd:string">  
1794 <xsd:maxLength value="255"/>  
1795 </xsd:restriction>  
1796 </xsd:simpleType>
```

1797 Client applications may request one or more symmetric keys conforming to one or more key classes
1798 required by the application. If the client application is authorized to receive keys conforming to such key
1799 classes, the SKS server will generate and supply them.

1800
1801 The <KeyClasses> element is useful only when requesting new symmetric keys, i.e. symmetric
1802 encryption keys that have previously NOT been used for encrypting data. There is little reason for a client
1803 application to specify the <KeyClasses> element when requesting an existing (escrowed) symmetric key,
1804 since the SKS server will return the requested key to authorized clients with whatever key class is
1805 associated with the key regardless of what key class is specified in the request. The key class will have
1806 been associated with the symmetric key at the time of its generation and cannot be changed once
1807 associated with a key.

1808
1809 When more than one <GlobalKeyID> is specified in the request, there MAY be only one <KeyClass>
1810 element within the <KeyClasses> set. When a key class is not specified in a request, it implies a request
1811 for symmetric key(s) of a default key class configured at the **SKS** server. The default key class for a site
1812 is site-specific.

1813
1814 When the client requires more than one symmetric key, and each key needs to be of a different key class,
1815 there MUST be only one <GlobalKeyID> element followed by as many <KeyClass> elements inside the
1816 <KeyClasses> set as needed by the client application. (Example 5 in this section).

1817
1818 When a client requires many symmetric keys – say five keys – and two or more keys belong to the same
1819 key class, the client MUST send multiple requests to the **SKS** server. One request will contain multiple
1820 <GlobalKeyID> elements with one <KeyClass> element in the <KeyClasses> set, and the other request
1821 will contain one <GlobalKeyID> element and multiple <KeyClass> elements within the <KeyClasses>
1822 set. (Examples 4 and 5 in this section).

1823 **Example 1 – A symmetric key request of a default key class (when no KeyClass is specified):**

```
1824 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1825 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1826 </ekmi:SymkeyRequest>
```

1827 **Example 2 – A request for multiple new symmetric keys, each of a default key class:**

```
1828 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1829 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1830 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1831 </ekmi:SymkeyRequest>
```

1832 **Example 3 – A request for a new symmetric key of a specific key class:**

```
1833 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1834 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1835 <ekmi:KeyClasses>  
1836 <ekmi:KeyClass>256-Bit-Class</ekmi:KeyClass>  
1837 </ekmi:KeyClasses>  
1838 </ekmi:SymkeyRequest>
```

1839 **Example 4 – A request for two new symmetric keys of the same key class for each symmetric key.**
1840 **Note that if the FIN-FX key class was the default key class, a request as shown in Example 2 of**
1841 **this section would result in the same response:**

```
1842 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1843 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1844 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1845 <ekmi:KeyClasses>  
1846 <ekmi:KeyClass>FIN-FX</ekmi:KeyClass>  
1847 </ekmi:KeyClasses>  
1848 </ekmi:SymkeyRequest>
```

1849 **Example 5 – A request for a four new symmetric keys of different key classes:**

```
1850 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1851 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
1852 <ekmi:KeyClasses>
1853 <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
1854 <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
1855 <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
1856 <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
1857 </ekmi:KeyClasses>
1858 </ekmi:SymkeyRequest>
```

1859 **4.4 Element <SymkeyResponse>**

1860 The <SymkeyResponse> element is one of two results returned by an **SKS** server upon being sent a valid
1861 and authorized <SymkeyRequest> by a client application. The other result is a <SymkeyError> which will
1862 be discussed in the next section.

1863 While <SymkeyResponse> is a top-level element within this specification, it MAY be enclosed within a
1864 **SOAP Body** element of a **SOAP Envelope** to conform to the security requirements of this specification.
1865 The **SOAP Header** of the **SOAP Envelope** MUST enclose a **Security** element conforming to [WSS] with
1866 a **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
1867 specified security profile in [WSS] to form a well-formed, secure message.
1868

1869 **Schema Definition:**

```
1870 <xsd:element name="SymkeyResponse">
1871 <xsd:complexType>
1872 <xsd:sequence>
1873 <xsd:element
1874 name="Symkey"
1875 type="tns:SymkeyType"
1876 minOccurs="0"
1877 maxOccurs="unbounded"/>
1878 <xsd:element
1879 name="SymkeyError"
1880 type="tns:SymkeyErrorType"
1881 minOccurs="0"
1882 maxOccurs="unbounded"/>
1883 </xsd:sequence>
1884 </xsd:complexType>
1885 </xsd:element>
```

1886 The <SymkeyResponse> element consists of a sequence of two types of child elements - <Symkey> or
1887 <SymkeyError>. The <SymkeyResponse> element MAY consist of either type of element or both types
1888 of elements. When both elements are contained in a <SymkeyResponse>, all <Symkey> elements
1889 MUST precede the first <SymkeyError> element.

1890 1. <Symkey> [Optional]

1891
1892 This element of type **SymkeyType**, is returned by the **SKS** server in response to a successful
1893 processing of a <SymkeyRequest>. There MAY be more than one <Symkey> element in the
1894 <SymkeyResponse> if the client application made a request for multiple symmetric keys.
1895

1896 The <Symkey> element and the **SymkeyType** are specified in Section 4.5.

1897 2. <SymkeyError> [Optional]

1898

1899 This element of type **SymkeyErrorType**, contains a response to a failed attempt in processing a
1900 request for one or more symmetric keys. There MAY be more than one <SymkeyError> element
1901 in the <SymkeyResponse> if the client application made a request for multiple symmetric keys and
1902 the request resulted in multiple errors.

1903
1904 The <SymkeyError> element is specified in Section 4.6.

1905 Some high-level examples of the <SymkeyResponse> element are as follows:

1906 **Example 1 – A response with a single symmetric key:**

```
1907 <ekmi:SymkeyResponse
1908     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1909     <ekmi:Symkey>.....</ekmi:Symkey>
1910 </ekmi:SymkeyResponse>
```

1911 **Example 2 – A response with three symmetric keys:**

```
1912 <ekmi:SymkeyResponse
1913     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1914     <ekmi:Symkey>.....</ekmi:Symkey>
1915     <ekmi:Symkey>.....</ekmi:Symkey>
1916     <ekmi:Symkey>.....</ekmi:Symkey>
1917 </ekmi:SymkeyResponse>
```

1918 **Example 3 – A response with an error:**

```
1919 <ekmi:SymkeyResponse
1920     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1921     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
1922 </ekmi:SymkeyResponse>
```

1923 **Example 4 – A response with multiple errors:**

```
1924 <ekmi:SymkeyResponse
1925     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1926     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
1927     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
1928 </ekmi:SymkeyResponse>
```

1929 **Example 5 – A response with one symmetric key and one error:**

```
1930 <ekmi:SymkeyResponse
1931     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1932     <ekmi:Symkey>.....</ekmi:Symkey>
1933     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
1934 </ekmi:SymkeyResponse>
```

1935 **Example 6 – A response with multiple symmetric keys and multiple error:**

```
1936 <ekmi:SymkeyResponse
1937     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1938     <ekmi:Symkey>.....</ekmi:Symkey>
1939     <ekmi:Symkey>.....</ekmi:Symkey>
1940     <ekmi:Symkey>.....</ekmi:Symkey>
1941     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
1942     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
1943 </ekmi:SymkeyResponse>
```

1944 4.5 Element <Symkey>

1945 The <Symkey> element is the *raison d'être* of the **SKSML** protocol. The element of type **SymkeyType**,
1946 contains the symmetric key returned by the **SKS** server, in response to a successful processing of a
1947 <SymkeyRequest> from a client application.

1948 Schema Definition:

```
1949 <xsd:complexType name="SymkeyType">  
1950 <xsd:sequence>  
1951 <xsd:element name="GlobalKeyID" type="ekmi:GlobalKeyIDType"/>  
1952 <xsd:element name="KeyUsePolicy" type="ekmi:KeyUsePolicyType"/>  
1953 <xsd:element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>  
1954 <xsd:element ref="xenc:CipherData"/>  
1955 </xsd:sequence>  
1956 </xsd:complexType>
```

1957 When a request for a symmetric key is successful, there MUST be at least one <Symkey> element in a
1958 <SymkeyResponse> element. There MAY be more than one <Symkey> element in the response if the
1959 client application made a request for multiple symmetric keys and the **SKS** server processed the request
1960 successfully.

1961 In the event of an error in processing the request, there SHALL be no <Symkey> element in the response;
1962 there SHALL be a <SymkeyError> element, instead. The <SymkeyError> element is specified in Section
1963 4.6.

1964 The <Symkey> element consists of a sequence of the following child elements:

1965 1. <GlobalKeyID> [Required]

1966 This element of type **GlobalKeyIDType** identifies the unique identifier of the symmetric key within
1967 an SKMS. There SHALL be only one <GlobalKeyID> within a <Symkey> element.

1968 The **GlobalKeyIDType** is specified in Section 4.2.

1971 2. <KeyUsePolicy> [Required]

1972 This element of type **KeyUsePolicyType**, defines how the symmetric key in this <Symkey>
1973 element may be used by applications. There SHALL be only one <KeyUsePolicy> element
1974 within a <Symkey> element.

1975 The <KeyUsePolicy> element is specified in Section 4.7.

1978 3. <EncryptionMethod> [Required]

1979 This element of type **EncryptionMethodType** from **[XMLEncryption]** describes how the
1980 symmetric key in this <Symkey> element is encrypted for transport between the **SKS** Server and
1981 the client application.

1982 The <EncryptionMethod> MUST specify one of the following two transport algorithms in the
1983 **Algorithm** attribute of the element:

- 1984 - http://www.w3.org/2001/04/xmlenc#rsa-1_5
- 1985 - <http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p>

1989 4. <CipherData> [Required]

1990 This element of **CipherDataType** from **[XMLEncryption]** contains the encrypted symmetric-key.
1991

1992 As specified in [XML Encryption], the content of this element is Base-64 encoded and is of the
1993 XML Schema *base64Binary* type.

1994 Some high-level examples of the <Symkey> element are as follows. Details about the <KeyUsePolicy>
1995 element have been elided for brevity:

1996 **Example 1 – A response with a symmetric key:**

```
1997 <ekmi:SymkeyResponse
1998   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1999   <ekmi:Symkey>
2000     <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
2001     <ekmi:KeyUsePolicy>....</ekmi:KeyUsePolicy>
2002     <ekmi:EncryptionMethod
2003       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
2004     <xenc:CipherData>
2005       <xenc:CipherValue>
2006         E9zWB/y93hVSzeTLiDcQoDxmLNxTux+SffMNwCJmt1dIqzQHBnpdQ8
2007         1g6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxj
2008         fg1pU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
2009       </xenc:CipherValue>
2010     </xenc:CipherData>
2011   </ekmi:Symkey>
2012 </ekmi:SymkeyResponse>
```

2013 **Example 2 – A response with multiple symmetric keys:**

```
2014 <ekmi:SymkeyResponse
2015   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2016   <ekmi:Symkey>
2017     <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
2018     <ekmi:KeyUsePolicy>....</ekmi:KeyUsePolicy>
2019     <ekmi:EncryptionMethod
2020       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
2021     <xenc:CipherData>
2022       <xenc:CipherValue>
2023         E9zWB/y93hVSzeTLiDcQoDxmLNxTux+SffMNwCJmt1dIqzQHBnpdQ8
2024         1g6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxj
2025         fg1pU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
2026       </xenc:CipherValue>
2027     </xenc:CipherData>
2028   </ekmi:Symkey>
2029   <ekmi:Symkey>
2030     <ekmi:GlobalKeyID>10514-1-236</ekmi:GlobalKeyID>
2031     <ekmi:KeyUsePolicy>....</ekmi:KeyUsePolicy>
2032     <ekmi:EncryptionMethod
2033       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
2034     <xenc:CipherData>
2035       <xenc:CipherValue>
2036         Qbg65cy93hVSzeTLiDcQoDxmLNxTux+SffMNwCJmt1dIqzQHBnpdQ8
2037         7k6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxj
2038         uyecU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
2039       </xenc:CipherValue>
2040     </xenc:CipherData>
2041   </ekmi:Symkey>
2042 </ekmi:SymkeyResponse>
```

2043 4.6 Element <SymkeyError>

2044 The <SymkeyError> element of type **SymkeyErrorType**, contains the error returned by the **SKS** server,
2045 in response to a failure in processing of a <SymkeyRequest> from a client application.

2046 Schema Definition:

```
2047 <xsd:complexType name="SymkeyErrorType">
2048   <xsd:sequence>
2049     <xsd:element name="RequestedGlobalKeyID" type="ekmi:GlobalKeyIDType"/>
2050     <xsd:element
2051       name="RequestedKeyClass"
2052       type="ekmi:KeyClassType"
2053       minOccurs="0"/>
2054     <xsd:element name="ErrorCode">
2055       <xsd:simpleType>
2056         <xsd:restriction base="xsd:string">
2057           <xsd:maxLength value="255"/>
2058         </xsd:restriction>
2059       </xsd:simpleType>
2060     </xsd:element>
2061     <xsd:element name="ErrorMessage">
2062       <xsd:simpleType>
2063         <xsd:restriction base="xsd:string">
2064           <xsd:maxLength value="1024"/>
2065         </xsd:restriction>
2066       </xsd:simpleType>
2067     </xsd:element>
2068   </xsd:sequence>
2069 </xsd:complexType>
```

2070 When a request for a symmetric key fails despite successfully being processed by the SOAP layer, there
2071 MUST be at least one <SymkeyError> element in a <SymkeyResponse> element. When a
2072 <SymkeyRequest> fails at the SOAP layer, the response SHALL consist of a **SOAPFault**.

2073 There MAY be more than one <SymkeyError> element in the response if the client application made a
2074 request for multiple symmetric keys and the **SKS** server failed in processing the request for more than
2075 one symmetric key.

2076 The <SymkeyError> element consists of a sequence of the following child elements:

2077 1. <RequestedGlobalKeyID> [Required]

2078

2079 This element of type **GlobalKeyIDType** identifies the unique identifier of the symmetric key
2080 requested by the client application. There SHALL be only one <RequestedGlobalKeyID> within
2081 a <SymkeyError> element.

2082

2083 The **GlobalKeyIDType** is specified in Section 4.2.

2084 2. <RequestedKeyClass> [Optional]

2085

2086 This element of type **KeyClassType** identifies the key-class of the symmetric key requested by
2087 the client application. If the <RequestedKeyClass> element is not embedded in the
2088 <SymkeyError> element, this implies that the requested symmetric key was for the default key-
2089 class of the SKMS.

2090

2091 The **KeyClassType** is specified in Section 4.3.

2092 3. <ErrorCode> [Required]

2093

2094 This element of type **String** identifies a mnemonic code identifying the error the **SKS** Server
2095 experienced in processing the client's symmetric key request.

2096
2097 The <ErrorCode> element SHALL return one of the codes identified in Appendix D of this
2098 specification.

2099 4. <ErrorMessage> [Required]

2100 This element of type **String** identifies a localized message describing the error the **SKS** Server
2101 experienced in processing the client's symmetric key request.

2102
2103 The <ErrorMessage> element SHALL return the appropriate localized version of the message
2104 corresponding to the <ErrorCode> element from Appendix D of this specification.
2105

2106 Some high-level examples of the <SymkeyError> element are as follows.

2107 **Example 1 – An error within a response:**

```
2108 <ekmi:SymkeyResponse  
2109     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
2110     <ekmi:SymkeyError>  
2111         <ekmi:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>  
2112         <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>  
2113         <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>  
2114     </ekmi:SymkeyError>  
2115 </ekmi:SymkeyResponse>
```

2116 **Example 2 – Multiple errors within a response:**

```
2117 <ekmi:SymkeyResponse  
2118     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
2119     <ekmi:SymkeyError>  
2120         <ekmi:RequestedGlobalKeyID>10514-1-0</ekmi:RequestedGlobalKeyID>  
2121         <ekmi:ErrorCode>SKS-100001</ekmi:ErrorCode>  
2122         <ekmi:ErrorMessage>Invalid GlobalKeyID</ekmi:ErrorMessage>  
2123     </ekmi:SymkeyError>  
2124     <ekmi:SymkeyError>  
2125         <ekmi:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>  
2126         <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>  
2127         <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>  
2128     </ekmi:SymkeyError>  
2129 </ekmi:SymkeyResponse>
```

2130

2131

2132 4.7 Element <KeyUsePolicy>

2133 The <KeyUsePolicy> element defines rules that conforming implementations of the **SKCL** MUST adhere
2134 to when using the symmetric key sent by the **SKS** Server. It is an integral part of the <Symkey> element .

2135 Schema Definition:

```
2136 <xsd:complexType name="KeyUsePolicyType" mixed="true">  
2137     <xsd:sequence>  
2138         <xsd:element name="KeyUsePolicyID" type="tns:TwoPartIDType"/>  
2139         <xsd:element name="PolicyName">  
2140             <xsd:simpleType>
```

```

2141         <xsd:restriction base="xsd:string">
2142             <xsd:maxLength value="255"/>
2143         </xsd:restriction>
2144     </xsd:simpleType>
2145 </xsd:element>
2146 <xsd:element name="KeyClass" type="tns:KeyClassType"/>
2147 <xsd:element name="KeyAlgorithm" type="tns:EncryptionAlgorithmType"/>
2148 <xsd:element name="KeySize" type="tns:KeySizeType"/>
2149 <xsd:element name="Status" type="tns:StatusType"/>
2150 <xsd:element name="Permissions" type="tns:PermissionsType"/>
2151 </xsd:sequence>
2152 </xsd:complexType>

```

2153 The <KeyUsePolicy> element is of the **KeyUsePolicyType** and consists of the following child elements:

2154 1. <KeyUsePolicyID> [Required]

2155

2156 The <KeyUsePolicyID> element, of type **TwoPartIDType**, identifies the unique policy object
2157 within the SKMS. There SHALL be only one <KeyUsePolicyID> element within a
2158 <KeyUsePolicy> element.

2159

2160 The **TwoPartIDType** is specified in Section 4.8.

2161 2. <PolicyName> [Required]

2162

2163 The <PolicyName> element, of type XSD **String**, with a maximum length of 255
2164 characters, identifies a unique name given to this <KeyUsePolicy>. There SHALL be
2165 only one <PolicyName> element within a <KeyUsePolicy> element.

2166 3. <KeyClass> [Required]

2167

2168 The <KeyClass> element, of type **KeyClassType**, identifies a key-class assigned to this
2169 <KeyUsePolicy>. There SHALL be only one <KeyUsePolicyID> element within a
2170 <KeyUsePolicy> element.

2171

2172 The **KeyClassType** is specified in Section 4.3.

2173 4. <KeyAlgorithm> [Required]

2174

2175 The <KeyAlgorithm> element, of type **EncryptionAlgorithmType**, identifies encryption
2176 algorithm to be used by applications when using this symmetric key. There SHALL be only one
2177 <KeyAlgorithm> element within a <KeyUsePolicy> element.

2178

2179 The <KeyAlgorithm> element is specified in Section 4.9.

2180 5. <KeySize> [Required]

2181

2182 The <KeySize> element, of type **KeySizeType**, defines the size of the symmetric key, in bits
2183 (binary digits). There SHALL be only one <KeySize> element within a <KeyUsePolicy> element.

2184

2185 Note: It is possible to determine the size of a symmetric key in an **SKCL** implementation without
2186 having to send the size in the response. So, why include it? It is our belief that while network
2187 bandwidth and compute performance of devices are increasing steadily, encryption is desired in
2188 many small and portable devices. Consequently, it will speed up applications in cryptographic
2189 processing if they do not have to determine the size of each key they use. While "protocol purity"
2190 demands that implementation issues do not show up in protocol design, we believe it is justified
2191 in this case.

2192

2193 The **KeySizeType** is specified in Section 4.10.

- 2194 6. <Status> [Required]
 2195
 2196 The <Status> element, of type **StatusType**, identifies the current status of the symmetric key.
 2197 There SHALL be only one <Status> element within a <KeyUsePolicy> element.
 2198
 2199 The **StatusType** is specified in Section 4.11.
- 2200 7. <Permissions> [Required]
 2201
 2202 The <Permissions> element, of type **PermissionsType**, defines what is permissible to client
 2203 applications with the symmetric key this element is associated with. It is the responsibility of the
 2204 conforming **SKCL** implementation to enforce these rules.
 2205
 2206 An important distinction of this element – unlike most access control rules – is that the absence of
 2207 sub-elements in the <Permissions> element implies that all permissions are allowed. The
 2208 presence of sub-elements in this element provide rules to the **SKCL** about what actions are
 2209 permitted.
 2210
 2211 There SHALL be only one <Permissions> element in a <KeyUsePolicy> element.
 2212
 2213 The **PermissionsType** is specified in Section 4.12.

2214 Some examples of the <KeyUsePolicy> element are as follows.

2215 **Example 1 – A <KeyUsePolicy> with some permission restrictions:**

```

2216 <ekmi:KeyUsePolicy>
2217   <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>
2218   <ekmi:PolicyName>DES-EDE KeyUsePolicy</ekmi:PolicyName>
2219   <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
2220   <ekmi:KeyAlgorithm>
2221     http://www.w3.org/2001/04/xmlenc#tripledes-cbc
2222   </ekmi:KeyAlgorithm>
2223   <ekmi:KeySize>192</ekmi:KeySize>
2224   <ekmi:Status>Active</ekmi:Status>
2225   <ekmi:Permissions>
2226     <ekmi:PermittedApplications ekmi:any="false">
2227       <ekmi:PermittedApplication>
2228         <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
2229         <ekmi:ApplicationName>Payroll Application</ekmi:ApplicationName>
2230         <ekmi:Version>1.0</ekmi:Version>
2231         <ekmi:DigestAlgorithm>
2232           http://www.w3.org/2000/09/xmlsig#sha1
2233         </ekmi:DigestAlgorithm>
2234         <ekmi:DigestValue>
2235           229ea73a5e76eabd183663d332b283948a9202a1
2236         </ekmi:DigestValue>
2237       </ekmi:PermittedApplication>
2238     </ekmi:PermittedApplications>
2239     <ekmi:PermittedDates ekmi:any="false">
2240       <ekmi:PermittedDate>
2241         <ekmi:StartDate>2008-01-01</ekmi:StartDate>
2242         <ekmi:EndDate>2008-12-31</ekmi:EndDate>
2243       </ekmi:PermittedDate>
2244     </ekmi:PermittedDates>
2245     <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2246     <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2247     <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2248     <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2249     <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
  
```

```

2250         <ekmi:PermittedTimes ekmi:any="false">
2251             <ekmi:PermittedTime>
2252                 <ekmi:StartTime>07:00:00</ekmi:StartTime>
2253                 <ekmi:EndTime>19:00:00</ekmi:EndTime>
2254             </ekmi:PermittedTime>
2255         </ekmi:PermittedTimes>
2256         <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2257     </ekmi:Permissions>
2258 </ekmi:KeyUsePolicy>

```

2259 **Example 2 – A <KeyUsePolicy> with no restrictions on the key:**

```

2260 <ekmi:KeyUsePolicy>
2261     <ekmi:KeyUsePolicyID>10514-2</ekmi:KeyUsePolicyID>
2262     <ekmi:PolicyName>Laptop KeyUsePolicy</ekmi:PolicyName>
2263     <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
2264     <ekmi:KeyAlgorithm>
2265         http://www.w3.org/2001/04/xmlenc#aes256-cbc
2266     </ekmi:KeyAlgorithm>
2267     <ekmi:KeySize>256</ekmi:KeySize>
2268     <ekmi>Status>Active</ekmi>Status>
2269     <ekmi:Permissions>
2270         <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
2271         <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
2272         <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2273         <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2274         <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2275         <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2276         <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
2277         <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
2278         <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2279     </ekmi:Permissions>
2280 </ekmi:KeyUsePolicy>

```

2281

2282 **4.8 Type TwoPartIDType**

2283 The **TwoPartIDType** is used to create identifiers for many elements within the **SKSML**. It is a simple
2284 concatenation of two integers with a hyphen between them ("-") to create an XML Schema **String** type.

2285 The **TwoPartIDType** has a minimum length of three (3) characters, and a maximum length of 41
2286 characters.

2287 **Schema Definition:**

```

2288 <xsd:simpleType name="TwoPartIDType">
2289     <xsd:restriction base="xsd:string">
2290         <xsd:minLength value="3"/>
2291         <xsd:maxLength value="41"/>
2292         <xsd:pattern value="[1-9][0-9]{0,19}-[1-9][0-9]{0,19}"/>
2293         <xsd:whiteSpace value="collapse"/>
2294     </xsd:restriction>
2295 </xsd:simpleType>

```

2296 The **TwoPartIDType** is used in the <ApplicationID>, the <KeyCachePolicyID> and the
2297 <KeyUsePolicyID> elements within the **SKSML**.

2298 Some examples of the <KeyUsePolicy> element are as follows.

2299 **Example 1 – A TwoPartIDType used to identify an ApplicationID:**

2300 <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>

2301 **Example 2 – A *TwoPartIDType* used to identify a *KeyUsePolicyID*:**

2302 <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>

2303 **Example 3 – A minimum-length *TwoPartIDType* :**

2304 <ekmi:KeyCachePolicyID>5-4</ekmi:KeyCachePolicyID>

2305 **Example 4 – A maximum-length *TwoPartIDType* :**

2306 <ekmi:ApplicationID>
2307 18446744073709551615-18446744073709551615
2308 </ekmi:ApplicationID>

2309

2310

2311 4.9 Element <KeyAlgorithm>

2312

2313 The element <KeyAlgorithm> , of type **EncryptionAlgorithmType**, is used to identify the cryptographic
2314 algorithm to be used with the symmetric keys in the <SymkeyResponse>.

2315 Schema Definition:

```
2316 <xsd:simpleType name="EncryptionAlgorithmType">  
2317   <xsd:restriction base="xsd:anyURI">  
2318     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#tripleDES-cbc"/>  
2319     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>  
2320     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes192-cbc"/>  
2321     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>  
2322   </xsd:restriction>  
2323 </xsd:simpleType>
```

2324 The algorithms currently supported by this specification are the algorithms defined in **[XMLEncryption]**.
2325 As new algorithms are added to **[XMLEncryption]**, they will be added to the enumerated list in this
2326 element. Currently, the following four algorithms are supported:

- 2327 1. Triple Data Encryption Standard (3DES)

2328

2329 Within the context of this specification, and as specified in **[XMLEncryption]**, the form of 3DES
2330 supported within **SKSML** is a 192-bit key with a 64-bit Initialization Vector. Of the key bits, the
2331 first 64 are used in the first DES operation, the second 64 bits in the second (middle) DES
2332 operation, and the third 64 bits in the third (last) DES operation. Each of these 64 bits of key
2333 contain 56 effective bits and 8 parity bits.

2334

2335 The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#tripleDES-cbc>.

- 2336 2. Advanced Encryption Standard (AES) – 128-bit

2337

2338 Within the context of this specification, and as specified in **[AES]**, this is a 128-bit symmetric key
2339 used in the Cipher Block Chaining (CBC) mode.

2340

2341 The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#aes128-cbc>.

- 2342 3. Advanced Encryption Standard (AES) – 192-bit

2343

2344 Within the context of this specification, and as specified in [AES], this is a 192-bit symmetric key
2345 used in the Cipher Block Chaining (CBC) mode.

2346 The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#aes192-cbc>.

2348 4. Advanced Encryption Standard (AES) – 256-bit

2349 Within the context of this specification, and as specified in [AES], this is a 256-bit symmetric key
2350 used in the Cipher Block Chaining (CBC) mode.

2351 The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#aes256-cbc>.

2352 There SHALL be only one <KeyAlgorithm> element within a <KeyUsePolicy> element.

2353 Some examples of the <KeyAlgorithm> element are as follows; other elements of the <KeyUsePolicy>
2354 element are not displayed for brevity:

2355 Example 1 – An example using the Triple-DES key algorithm:

```
2356 <ekmi:KeyUsePolicy>  
2357 ...  
2358 <ekmi:KeyAlgorithm>  
2359 http://www.w3.org/2001/04/xmlenc#tripledes-cbc  
2360 </ekmi:KeyAlgorithm>  
2361 ...  
2362 </ekmi:KeyUsePolicy>
```

2363 Example 2 – An example using the AES-128 key algorithm:

```
2364 <ekmi:KeyUsePolicy>  
2365 ...  
2366 <ekmi:KeyAlgorithm>  
2367 http://www.w3.org/2001/04/xmlenc#aes128-cbc  
2368 </ekmi:KeyAlgorithm>  
2369 ...  
2370 </ekmi:KeyUsePolicy>
```

2371

2372

2373 4.10 Element <KeySize>

2374

2375 The element <KeySize> , of type **KeySizeType**, is used to identify the size of the symmetric key, in
2376 binary digits (bits) in the <SymkeyResponse>.

2377 Schema Definition:

```
2378 <xsd:simpleType name="KeySizeType">  
2379 <xsd:restriction base="xsd:unsignedShort">  
2380 <xsd:totalDigits value="3"/>  
2381 <xsd:fractionDigits value="0"/>  
2382 <xsd:enumeration value="128"/>  
2383 <xsd:enumeration value="192"/>  
2384 <xsd:enumeration value="256"/>  
2385 </xsd:restriction>  
2386 </xsd:simpleType>
```

2387 There SHALL be only one <KeySize> element within a <KeyUsePolicy> element.

2390 Currently, the following three key-sizes are supported:

- 2391 1. 128-bits when used with the **AES-192** algorithm
- 2392 2. 192-bits when used with the **AES-192** or the **3DES** algorithms
- 2393 3. 256-bits when used with the **AES-256** algorithm

2394 Some examples of the <KeySize> element are as follows; other elements of the <KeyUsePolicy>
2395 element are not displayed for brevity:

2396 **Example 1 – An example using a 128-bit key size:**

```
2397 <ekmi:KeyUsePolicy>
2398   ...
2399   <ekmi:KeySize>128</ekmi:KeySize>
2400   ...
2401 </ekmi:KeyUsePolicy>
```

2402 **Example 2 – An example using a 192-bit key size:**

```
2403 <ekmi:KeyUsePolicy>
2404   ...
2405   <ekmi:KeySize>192</ekmi:KeySize>
2406   ...
2407 </ekmi:KeyUsePolicy>
```

2408 **Example 3 – An example using a 256-bit key size:**

```
2409 <ekmi:KeyUsePolicy>
2410   ...
2411   <ekmi:KeySize>256</ekmi:KeySize>
2412   ...
2413 </ekmi:KeyUsePolicy>
```

2414

2415

2416 **4.11 Element <Status>**

2417

2418 The element <Status>, of type **StatusType**, is used to identify the current status of an object . It is
2419 used in almost every element within the SKMS.

2420 **Schema Definition:**

```
2421 <xsd:simpleType name="StatusType">
2422   <xsd:restriction base="xsd:string">
2423     <xsd:enumeration value="Active"/>
2424     <xsd:enumeration value="Default"/>
2425     <xsd:enumeration value="Inactive"/>
2426     <xsd:enumeration value="Other"/>
2427   </xsd:restriction>
2428 </xsd:simpleType>
```

2429 Where it exists within an element, there SHALL be only one <Status> element within the enclosing
2430 element.

2431 The <Status> element can contain one of four **String** type values:

- 2432 1. The **Active** value indicates that the element that makes up the document-root is currently active in
2433 the SKMS and conforming **SKCL** implementations may use it within applications.
- 2434 2. The **Default** value indicates that the element that makes up the document root is the default
2435 element in the SKMS, is also active, and conforming **SKCL** implementations may use it within
2436 applications.
- 2437 3. The **Inactive** value indicates that the element that makes up the document root is not active in the
2438 SKMS, and conforming **SKCL** implementations may NOT use it within applications.
- 2439 4. The **Other** value indicates that the element that makes up the document root has a meaning that
2440 is application-specific. However, conforming **SKCL** implementations may NOT use it within
2441 applications.

2442 Some examples of the <Status> element are shown below; other parts of their enclosing elements are
2443 not shown for brevity:

2444 **Example 1 – An example with an Active status within a <KeyUsePolicy> element:**

```
2445 <ekmi:KeyUsePolicy>
2446   ...
2447   <ekmi:Status>Active</ekmi:Status>
2448   ...
2449 </ekmi:KeyUsePolicy>
```

2450 **Example 2 – An example with an Inactive status within a <KeyUsePolicy> element:**

```
2451 <ekmi:KeyUsePolicy>
2452   ...
2453   <ekmi:Status>Inactive</ekmi:Status>
2454   ...
2455 </ekmi:KeyUsePolicy>
```

2456 **Example 3 – An example with a Default status within a <KeyUsePolicy> element:**

```
2457 <ekmi:KeyUsePolicy>
2458   ...
2459   <ekmi:Status>Default</ekmi:Status>
2460   ...
2461 </ekmi:KeyUsePolicy>
```

2462

2463

2464 **4.12 Element <Permissions>**

2465 The <Permissions> element, of the type *PermissionsType* is at the heart of the <KeyUsePolicy>
2466 element. It provides guidance to conforming **SKCL** implementations on who may use the symmetric key,
2467 when they may use it, for what purposes, for how long and in which locations. For applications that
2468 conform to the Multi-Level Security (MLS) model, there is a provision for specifying which levels are
2469 permitted use of the key. There is also an element that allows for extending the <Permissions> element
2470 to accommodate rules that have not been envisioned in the current specification.

2471 There SHALL be only one <Permissions> element within a <KeyUsePolicy> element.

2472 **Schema Definition:**

```
2473 <xsd:complexType name="PermissionsType">
2474 <xsd:sequence>
2475 <xsd:element
```



```

2476         name="PermittedApplications"
2477         type="tns:PermittedApplicationsType"
2478         minOccurs="1"
2479         nillable="true"/>
2480     <xsd:element
2481         name="PermittedDates"
2482         type="tns:PermittedDatesType"
2483         minOccurs="1"
2484         nillable="true"/>
2485     <xsd:element
2486         name="PermittedDays"
2487         type="tns:PermittedDaysType"
2488         minOccurs="1"
2489         nillable="true"/>
2490     <xsd:element
2491         name="PermittedDuration"
2492         type="tns:PermittedDurationType"
2493         minOccurs="1"
2494         nillable="true"/>
2495     <xsd:element
2496         name="PermittedLevels"
2497         type="tns:PermittedLevelsType"
2498         minOccurs="1"
2499         nillable="true"/>
2500     <xsd:element
2501         name="PermittedLocations"
2502         type="tns:PermittedLocationsType"
2503         minOccurs="1"
2504         nillable="true"/>
2505     <xsd:element
2506         name="PermittedNumberOfTransactions"
2507         type="tns:PermittedNumberOfTransactionsType"
2508         minOccurs="1"
2509         nillable="true"/>
2510     <xsd:element
2511         name="PermittedTimes"
2512         type="tns:PermittedTimesType"
2513         minOccurs="1"
2514         nillable="true"/>
2515     <xsd:element
2516         name="PermittedUses"
2517         type="tns:PermittedUsesType"
2518         minOccurs="1"
2519         nillable="true"/>
2520     <xsd:element
2521         name="Other"
2522         type="xsd:anyType"
2523         minOccurs="0"/>
2524 </xsd:sequence>
2525 </xsd:complexType>

```

2526
2527 The <Permissions> element consists of the following sub-elements:

- 2528 1. A required <PermittedApplications> element which identifies applications that are permitted
2529 use of the symmetric key in question. While the <PermittedApplications> element is required,
2530 it may be empty (NULL). The absence of sub-elements in the <PermittedApplications>
2531 element implies that all applications are permitted to use the key. Identifying a specific application
2532 restricts the use of the key to only the identified applications.

2533
2534

The <PermittedApplications> element is specified in Section 4.13.

2535
2536
2537
2538
2539
2540
2541

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

The <PermittedDates> element is specified in Section 4.12.

2542
2543
2544
2545
2546
2547
2548

3. A required <PermittedDays> element which identifies the days of week during which applications are permitted use of the symmetric key in question. While the <PermittedDays> element is required, it may be empty (NULL). The absence of sub-elements in the <PermittedDays> element implies that applications are permitted to use the key on any day of the week. Identifying specific days restricts the use of the key to only the identified days.

The <PermittedDays> element is specified in Section 4.15.

2549
2550
2551
2552
2553
2554
2555
2556

4. A required <PermittedDuration> element which identifies the duration (in seconds) in which applications are permitted use of the symmetric key in question *once the SKCL starts using the symmetric key*. While the <PermittedDuration> element is required, it may be empty (NULL). The absence of any content – the duration time - in the <PermittedDuration> element implies that applications are permitted to use the key for any duration after it has been used. Identifying a non-zero, positive duration value restricts the use of the key to only the period after the start of the use of the key.

A distinction between <PermittedDates> and <PermittedDuration> is that the former has fixed start and end-dates for the use of the key, whereas the latter has a fixed end-date-and-time after the key has begun to be used without a fixed start-date-and-time. Thus, an application with a <PermittedDuration> can begin the use of a symmetric key at any time, but must stop its use at the end of the <PermittedDuration> once it has begun. With <PermittedDates>, an application can continue using the symmetric key until the fixed date-and-time have been reached.

2564
2565

The <PermittedDuration> element is specified in Section 4.16

2566
2567
2568
2569
2570
2571
2572
2573

5. Within a Multi-Level Security (MLS) system, the required <PermittedLevels> element identifies the security levels at which applications are permitted use of the symmetric key in question. While the <PermittedLevels> element is required, it may be empty (NULL). The absence of sub-elements in the <PermittedLevels> element implies that applications are permitted to use the key at any level of security. Identifying specific MLS level(s) restricts the use of the key to only the identified security level(s).

The <PermittedLevels> element is specified in Section 4.17

2574
2575
2576
2577
2578
2579
2580
2581

6. A required <PermittedLocations> element which identifies physical geographic locations where applications are permitted use of the symmetric key in question. While the <PermittedLocations> element is required, it may be empty (NULL). The absence of sub-elements in the <PermittedLocations> element implies that applications are permitted to use the key at any physical location. Identifying specific locations restricts the use of the key to only the identified locations.

The <PermittedLocations> element is specified in Section 4.18.

2582
2583
2584

7. A required <PermittedNumberOfTransactions> element which identifies the number of encryption transactions that applications are permitted, with the use of the symmetric key in question. While the <PermittedNumberOfTransactions> element is required, it may be empty

2585 (NULL). The absence of content – the number of transactions – in the
2586 <PermittedNumberOfTransactions> element implies that applications are permitted to use the
2587 key for as many encryption transactions as necessary. Identifying a specific, non-zero, positive
2588 number of transactions in this element restricts the use of the key to only the limit identified in the
2589 element.

2590 The <PermittedNumberOfTransactions> element is specified in Section 4.19.

2592 8. A required <PermittedTimes> element which identifies the times of day during which applications
2593 are permitted the use of the symmetric key in question. While the <PermittedTimes> element is
2594 required, it may be empty (NULL). The absence of sub-elements in the <PermittedTimes>
2595 element implies that applications are permitted to use the key at any time of day. Identifying
2596 specific times restricts the use of the key to only the duration of the identified times.

2597 The <PermittedTimes> element is specified in Section 4.20.

2599 9. A required <PermittedUses> element which identifies application-uses that applications are
2600 permitted with the symmetric key in question. While the <PermittedUses> element is required, it
2601 may be empty (NULL). The absence of sub-elements in the <PermittedUses> element implies
2602 that applications are permitted to use the key for any purpose. Identifying specific uses restricts
2603 the use of the key to only the identified uses.

2604 The <PermittedUses> element is specified in Section 4.21.

2606 10. The optional <Other> element allows implementers to specify permissions that cannot be
2607 addressed with the above-mentioned categories, for restricting the use of the symmetric key in
2608 question.

2609 While the <Other> element provides flexibility for implementations, the disadvantage of the
2610 element is that it may render a specific implementation incompatible with other **SKMS**
2611 implementations that use the **SKSML** standard.

2612 **It is strongly recommended that implementers avoid the use of the <Other> element**
2613 **unless they definitely do not expect to inter-operate with other SKCL implementations. If**
2614 **there is a strong need for capability that does not exist within the current specification of**
2615 **the <Permissions> element, implementers are encouraged to contact the OASIS EKMI TC**
2616 **with their requirements.**

2617 When all sub-elements of the <Permissions> element are empty, there are no restrictions on the use of
2618 the symmetric key other than that the application calling on the **SKCL** be authorized to access the key in
2619 question. However, when there are elements defined within the sub-elements of the <Permissions>
2620 element, conforming **SKCL** implementations must comply with all the permission elements, evaluating the
2621 most restrictive permissions first and in decreasing order of restriction, before allowing the use of the key.
2622

2623 For example, if a <Permissions> element specifies that a key may be used on Weekdays, between the
2624 hours of 0900 and 1700 Hours, then a request for a symmetric key on a Saturday at 1105 would deny use
2625 of the key in question, since it violates the more restrictive permission of being allowed for use only on
2626 weekdays.
2627

2628 **It should be noted that it is the primary responsibility of a conforming SKCL to enforce the**
2629 **<Permission> elements' rules. The SKS server will generate the key – or return an existing key -**
2630 **when an authorized client with appropriate access requests it. However, it is up to the SKCL**
2631 **implementation to comply with the rules in the <Permissions> element.**

2632 In another example, if a <Permissions> element specifies a <PermittedDuration> of 600 seconds from
2633 the start of use of the key, and there is also present a <PermittedNumberOfTransactions> element with
2634 a value of 10 (encryption transactions), conforming **SKCL** implementations must evaluate both
2635 permissions before each transaction and determine if they are both within the specified thresholds before
2636 using the key. If the 600 seconds expire before the 10 encryption transactions have been completed, or if

2637 the 10 encryption transactions are completed before 600 seconds have expired, conforming **SKCL**
2638 implementations **MUST** not use the key in question anymore.

2639 Some examples of the <Permissions> element are as shown below; the enclosing <KeyUsePolicy>
2640 element, <Symkey> element and <SymkeyResponse> elements are not displayed for brevity:

2641 **Example 1 – A <Permissions> element that permits a single application the use of the symmetric**
2642 **key in question, between January 01, 2008 and December 31, 2008 and between the hours of 0700**
2643 **and 1900. There are, however, no restrictions on what days of the week the key may be used, the**
2644 **locations where it may be used, at which MLS level it may be used (if it applies), the number of**
2645 **data files/transactions that may be encrypted with the key or the uses of the key within that**
2646 **application:**

```
2647 <ekmi:Permissions>
2648   <ekmi:PermittedApplications ekmi:any="false">
2649     <ekmi:PermittedApplication>
2650       <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
2651       <ekmi:ApplicationName>Payroll Application</ekmi:ApplicationName>
2652       <ekmi:Version>1.0</ekmi:Version>
2653       <ekmi:DigestAlgorithm>
2654         http://www.w3.org/2000/09/xmldsig#sha1
2655       </ekmi:DigestAlgorithm>
2656       <ekmi:DigestValue>
2657         229ea73a5e76eabd183663d332b283948a9202a1
2658       </ekmi:DigestValue>
2659     </ekmi:PermittedApplication>
2660   </ekmi:PermittedApplications>
2661   <ekmi:PermittedDates ekmi:any="false">
2662     <ekmi:PermittedDate>
2663       <ekmi:StartDate>2008-01-01</ekmi:StartDate>
2664       <ekmi:EndDate>2008-12-31</ekmi:EndDate>
2665     </ekmi:PermittedDate>
2666   </ekmi:PermittedDates>
2667   <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2668   <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2669   <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2670   <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2671   <ekmi:PermittedNumberOfWorkTransactions ekmi:any="true" xsi:nil="true"/>
2672   <ekmi:PermittedTimes ekmi:any="false">
2673     <ekmi:PermittedTime>
2674       <ekmi:StartTime>07:00:00</ekmi:StartTime>
2675       <ekmi:EndTime>19:00:00</ekmi:EndTime>
2676     </ekmi:PermittedTime>
2677   </ekmi:PermittedTimes>
2678   <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2679 </ekmi:Permissions>
```

2680 **Example 2 – A <Permissions> element that permits two specific applications the use of the**
2681 **symmetric key in question, between January 01, 2009 and January 31, 2009.**

```
2682 <ekmi:Permissions>
2683   <ekmi:PermittedApplications ekmi:any="false">
2684     <ekmi:PermittedApplication>
2685       <ekmi:ApplicationID>10514-24</ekmi:ApplicationID>
2686       <ekmi:ApplicationName>
2687         Employee Tax Reporting Application
2688       </ekmi:ApplicationName>
2689       <ekmi:Version>3.3</ekmi:Version>
2690       <ekmi:DigestAlgorithm>
2691         http://www.w3.org/2000/09/xmldsig#sha1
2692       </ekmi:DigestAlgorithm>
```

```

2693         <ekmi:DigestValue>
2694             af96d65a7a2415239c8eb8be1347f704322957a4
2695         </ekmi:DigestValue>
2696     </ekmi:PermittedApplication>
2697     <ekmi:PermittedApplication>
2698         <ekmi:ApplicationID>10514-25</ekmi:ApplicationID>
2699         <ekmi:ApplicationName>
2700             IRS Tax Reporting Application
2701         </ekmi:ApplicationName>
2702         <ekmi:Version>2.1</ekmi:Version>
2703         <ekmi:DigestAlgorithm>
2704             http://www.w3.org/2000/09/xmlsig#sha1
2705         </ekmi:DigestAlgorithm>
2706         <ekmi:DigestValue>
2707             a4f5925185ffe12c1a91ea3de90fc086b34b34b2
2708         </ekmi:DigestValue>
2709     </ekmi:PermittedApplication>
2710 </ekmi:PermittedApplications>
2711 <ekmi:PermittedDates ekmi:any="false">
2712     <ekmi:PermittedDate>
2713         <ekmi:StartDate>2009-01-01</ekmi:StartDate>
2714         <ekmi:EndDate>2009-12-31</ekmi:EndDate>
2715     </ekmi:PermittedDate>
2716 </ekmi:PermittedDates>
2717 <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2718 <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2719 <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2720 <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2721 <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
2722 <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
2723 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2724 </ekmi:Permissions>

```

2725 **Example 3 – A <Permissions> element that permits all applications the use of the symmetric key**
2726 **in question, for 100 transactions for encrypting credit card numbers.**

```

2727 <ekmi:Permissions>
2728     <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
2729     <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
2730     <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2731     <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2732     <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2733     <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2734     <ekmi:PermittedNumberOfTransactions ekmi:any="false">
2735         100
2736     </ekmi:PermittedNumberOfTransactions>
2737     <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
2738     <ekmi:PermittedUses ekmi:any="false">
2739         <ekmi:PermittedUse>CCN</ekmi:PermittedUse>
2740     </ekmi:PermittedUses>
2741 </ekmi:Permissions>

```

2742 **Example 4 – A <Permissions> element that permits all applications the use of the symmetric key**
2743 **in question, for 600 seconds once the SKCL starts using the key.**

```

2744 <ekmi:Permissions>
2745     <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
2746     <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
2747     <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2748     <ekmi:PermittedDuration ekmi:any="false">600</ekmi:PermittedDuration>
2749     <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>

```

```

2750     <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2751     <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
2752     <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
2753     <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2754 </ekmi:Permissions>

```

2755 **Example 5 – A <Permissions> element that permits a specific application the use of the**
2756 **symmetric key in question, at specific geographic locations only on weekdays between the hours**
2757 **of 0800 and 1700, and only when the application is operating at the Secret security level within an**
2758 **MLS system.**

```

2759 <ekmi:Permissions>
2760   <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
2761   <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
2762   <ekmi:PermittedDays ekmi:any="false">
2763     <ekmi:PermittedDay>Weekday</ekmi:PermittedDay>
2764   </ekmi:PermittedDays>
2765   <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2766   <ekmi:PermittedLevels ekmi:any="false">
2767     <ekmi:PermittedLevel>Secret</ekmi:PermittedLevel>
2768   </ekmi:PermittedLevels>
2769   <ekmi:PermittedLocations ekmi:any="false">
2770     <ekmi:PermittedLocation>
2771       <ekmi:LocationName>Facility A51</ekmi:LocationName>
2772       <ekmi:Latitude>37.385562</ekmi:Latitude>
2773       <ekmi:Longitude>-121.993387</ekmi:Longitude>
2774     </ekmi:PermittedLocation>
2775     <ekmi:PermittedLocation>
2776       <ekmi:LocationName>Facility DC-VA01</ekmi:LocationName>
2777       <ekmi:Latitude>88.485362</ekmi:Latitude>
2778       <ekmi:Longitude>-21.453648</ekmi:Longitude>
2779     </ekmi:PermittedLocation>
2780   </ekmi:PermittedLocations>
2781   <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
2782   <ekmi:PermittedTimes ekmi:any="false">
2783     <ekmi:PermittedTime>
2784       <ekmi:StartTime>08:00:00</ekmi:StartTime>
2785       <ekmi:EndTime>17:00:00</ekmi:EndTime>
2786     </ekmi:PermittedTime>
2787   </ekmi:PermittedTimes>
2788   <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2789 </ekmi:Permissions>

```

2790

2791

2792 4.13 Element <PermittedApplications> and <PermittedApplication>

2793

2794 The element <PermittedApplications>, of type **PermittedApplicationsType** and its only child-
2795 element <PermittedApplication> of type **ApplicationsType** are used to define the list of applications
2796 that are permitted to use a symmetric key within a specific <Symkey> element.

2797 Schema Definition:

```

2798   <xsd:complexType name="PermittedApplicationsType">
2799     <xsd:sequence>
2800       <xsd:element
2801         name="PermittedApplication"

```

```

2802         type="tns:ApplicationsType"
2803         minOccurs="0"
2804         maxOccurs="unbounded"/>
2805     </xsd:sequence>
2806     <xsd:attribute ref="tns:any" use="required"/>
2807 </xsd:complexType>

```

2808 **Schema Definition:**

```

2809
2810 <xsd:complexType name="ApplicationsType">
2811     <xsd:sequence>
2812         <xsd:element name="ApplicationID" type="tns:TwoPartIDType"/>
2813         <xsd:element name="ApplicationName">
2814             <xsd:simpleType>
2815                 <xsd:restriction base="xsd:string">
2816                     <xsd:maxLength value="256"/>
2817                     <xsd:whiteSpace value="preserve"/>
2818                 </xsd:restriction>
2819             </xsd:simpleType>
2820         </xsd:element>
2821         <xsd:element name="Version" minOccurs="0">
2822             <xsd:simpleType>
2823                 <xsd:restriction base="xsd:string">
2824                     <xsd:maxLength value="32"/>
2825                     <xsd:whiteSpace value="preserve"/>
2826                 </xsd:restriction>
2827             </xsd:simpleType>
2828         </xsd:element>
2829         <xsd:group ref="tns:MessageDigestGroup" minOccurs="0"/>
2830         <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
2831     </xsd:sequence>
2832 </xsd:complexType>

```

2833 **Schema Definition:**

```

2834
2835 <xsd:group name="MessageDigestGroup">
2836     <xsd:sequence>
2837         <xsd:element name="DigestAlgorithm">
2838             <xsd:simpleType>
2839                 <xsd:restriction base="xsd:anyURI">
2840                     <xsd:enumeration value="http://www.w3.org/2000/09/xmlsig#sha1"/>
2841                     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#sha256"/>
2842                     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#sha512"/>
2843                 </xsd:restriction>
2844             </xsd:simpleType>
2845         </xsd:element>
2846         <xsd:element name="DigestValue">
2847             <xsd:simpleType>
2848                 <xsd:restriction base="xsd:base64Binary">
2849                     <xsd:maxLength value="1024"/>
2850                 </xsd:restriction>
2851             </xsd:simpleType>
2852         </xsd:element>
2853     </xsd:sequence>
2854 </xsd:group>

```

2855 **Schema Definition:**

```

2856     <xsd:attribute name="any">
2857         <xsd:simpleType>
2858             <xsd:restriction base="xsd:string">

```

```
2859         <xsd:enumeration value="false"/>
2860         <xsd:enumeration value="true"/>
2861     </xsd:restriction>
2862 </xsd:simpleType>
2863 </xsd:attribute>
```

2864 There SHALL be only one <PermittedApplications> element within the <Permissions> element.
2865 However, there MAY be an unbounded (unlimited) number of <PermittedApplication> elements within
2866 a <PermittedApplications> element.

2867 The <PermittedApplications> element SHALL have one attribute named "any", that will have a
2868 "false" or "true" value, based on the following:

- 2869 • When the <PermittedApplications> element is null (i.e. it does not have a single
2870 <PermittedApplication> sub-element in it), the value of the "any" attribute SHALL be set to
2871 "true" AND the XML Schema Instance (XSI) "nil" attribute SHALL be set to "true".
- 2872 • When the <PermittedApplications> element is not-null (i.e. it has at least one
2873 <PermittedApplication> sub-element in it), the value of the "any" attribute SHALL be set to
2874 "false" AND the XML Schema Instance (XSI) "nil" attribute SHALL NOT be present.

2875 A null <PermittedApplications> element specifies that ALL applications are permitted use of the
2876 symmetric key, subject to complying with all other permission clauses in the <KeyUsePolicy> element.

2877 The <PermittedApplication> sub-element of type **ApplicationsType**, provides details of the
2878 application which is permitted use of the symmetric key in question. The <PermittedApplication>
2879 element consists of the following sub-elements:

- 2880 1. The <ApplicationID> element identifies the unique identifier assigned to this application within
2881 the SKMS. It is a **TwoPartIDType** as specified in Section 4.8. There SHALL be only one
2882 <ApplicationID> element within a <PermittedApplication> element.
- 2883 2. The <ApplicationName> element identifies the name assigned to this application within the
2884 SKMS. It is an XSD **String** with a maximum length of 256 characters. There SHALL be only one
2885 <ApplicationName> element within a <PermittedApplication> element.
- 2886 3. An optional <Version> element identifying the version number of this application within the
2887 **SKMS**. It is an XSD **String** with a maximum length of 32 characters. There MAY be only one
2888 <Version> element within a <PermittedApplication> element.
- 2889 4. The <MessageDigestGroup> group which identifies the message digest value of the application's
2890 binary image, along with the message digest algorithm used to calculate the digest value. The
2891 <MessageDigestGroup> consists of the following elements:
 - 2892 a) The <DigestAlgorithm> element of the XSD type **anyURI**, which supports one of the
2893 following three digest algorithms:
 - 2894 i. <http://www.w3.org/2000/09/xmlsig#sha1>
 - 2895 ii. <http://www.w3.org/2001/04/xmlenc#sha256>
 - 2896 iii. <http://www.w3.org/2001/04/xmlenc#sha512>
 - 2897 b) The <DigestValue> element of the XSD type **base64Binary**.
- 2898 There SHALL be only one <MessageDigestGroup> group within a <PermittedApplication>
2899 element.
- 2900 5. An optional <Other> element that provides implementers the ability to carry other information
2901 about the application that may be relevant to their **SKMS**. Implementers are cautioned that the
2902 use of the <Other> element may not be supported by other **SKCL** implementations, and may
2903 break interoperability between two **SKMS** implementations. Should there be a strong need for

2904 additional features in the <PermittedApplication> element, implementers are encouraged to
2905 contact the OASIS EKMI TC with their requirements.

2906 NOTE: The **SKSML** specification does not specify how an **SKCL** implementation will determine the
2907 message digest of an application that needs to use the symmetric key in question. It is left to the
2908 implementers of the **SKCL** to determine the message digest of the calling application using the specified
2909 algorithm, and verify that the digest values match before the **SKCL** uses the symmetric key on behalf of
2910 the application.

2911 Some examples of the <PermittedApplications> element are shown below; other parts of their
2912 enclosing elements are not shown for brevity:

2913 **Example 1 – An example of a <PermittedApplications> element with two child**
2914 **<PermittedApplication> elements with specific version numbers and message digest values:**

```
2915 <ekmi:PermittedApplications ekmi:any="false">
2916   <ekmi:PermittedApplication>
2917     <ekmi:ApplicationID>10514-24</ekmi:ApplicationID>
2918     <ekmi:ApplicationName>Employee Tax
2919 Reporting</ekmi:ApplicationName>
2920     <ekmi:Version>3.3</ekmi:Version>
2921     <ekmi:DigestAlgorithm>
2922       http://www.w3.org/2000/09/xmlsig#sha1
2923     </ekmi:DigestAlgorithm>
2924     <ekmi:DigestValue>G4bsdfKkt4cziEqFFu0oBTM81efU=</ekmi:DigestValue>
2925   </ekmi:PermittedApplication>
2926   <ekmi:PermittedApplication>
2927     <ekmi:ApplicationID>10514-25</ekmi:ApplicationID>
2928     <ekmi:ApplicationName>IRS Tax Reporting
2929 Application</ekmi:ApplicationName>
2930     <ekmi:Version>2.1</ekmi:Version>
2931     <ekmi:DigestAlgorithm>
2932       http://www.w3.org/2001/04/xmlenc#sha256
2933     </ekmi:DigestAlgorithm>
2934     <ekmi:DigestValue>
2935       ab7b85c9410d48c54fc7939c391be4028e7305085191c56e7b3740f2cbdbbc79
2936     </ekmi:DigestValue>
2937   </ekmi:PermittedApplication>
2938 </ekmi:PermittedApplications>
```

2940 **Example 2 – An example of a <PermittedApplications> element with one child**
2941 **<PermittedApplication> element that applies to all versions of the application; the message**
2942 **digest value and algorithm are not used in this example:**

```
2943 <ekmi:PermittedApplications ekmi:any="false">
2944   <ekmi:PermittedApplication>
2945     <ekmi:ApplicationID>10514-14</ekmi:ApplicationID>
2946     <ekmi:ApplicationName>E-Commerce Payment</ekmi:ApplicationName>
2947   </ekmi:PermittedApplication>
2948 </ekmi:PermittedApplications>
```

2949 **Example 3 – An example of a null <PermittedApplications> element specifying that ALL**
2950 **applications are permitted the use of the symmetric key:**

```
2951 <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
```

2952

2953 4.14 Element <PermittedDates> and <PermittedDate>

2954 The element <PermittedDates>, of type *PermittedDatesType* and its only child-element
2955 <PermittedDate>, which is an anonymous XSD *ComplexType*, are used to define ranges of dates
2956 between which applications are permitted to use the symmetric key within a specific <Symkey> element.

2957 Schema Definition:

```
2958 <xsd:complexType name="PermittedDatesType">
2959   <xsd:sequence>
2960     <xsd:element name="PermittedDate" minOccurs="0" maxOccurs="unbounded">
2961       <xsd:complexType>
2962         <xsd:sequence>
2963           <xsd:element name="StartDate">
2964             <xsd:simpleType>
2965               <xsd:restriction base="xsd:date">
2966                 <xsd:pattern value="\p{Nd}{4}-\p{Nd}{2}-\p{Nd}{2}"/>
2967               </xsd:restriction>
2968             </xsd:simpleType>
2969           </xsd:element>
2970           <xsd:element name="EndDate">
2971             <xsd:simpleType>
2972               <xsd:restriction base="xsd:date">
2973                 <xsd:pattern value="\p{Nd}{4}-\p{Nd}{2}-\p{Nd}{2}"/>
2974               </xsd:restriction>
2975             </xsd:simpleType>
2976           </xsd:element>
2977         </xsd:sequence>
2978       </xsd:complexType>
2979     </xsd:element>
2980   </xsd:sequence>
2981   <xsd:attribute ref="tns:any" use="required"/>
2982 </xsd:complexType>
```

2983 There SHALL be only one <PermittedDates> element within the <Permissions> element. However,
2984 there MAY be an unbounded number of <PermittedDate> elements within a <PermittedDates>
2985 element.

2986 The <PermittedDates> element SHALL have one attribute named “any”, that will have a “false” or “true”
2987 value, based on the following:

- 2988 • When the <PermittedDates> element is null (i.e. it does not have a single <PermittedDate>
2989 sub-element in it), the value of the “any” attribute SHALL be set to “true” AND the XML Schema
2990 Instance (XSI) “nil” attribute SHALL be set to “true”.
- 2991 • When the <PermittedDates> element is not-null (i.e. it has at least one <PermittedDate> sub-
2992 element in it), the value of the “any” attribute SHALL be set to “false” AND the XML Schema
2993 Instance (XSI) “nil” attribute SHALL NOT be present.

2994 A null <PermittedDates> element specifies that applications are permitted use of the symmetric key on
2995 any calendar date of the year, subject to complying with all other permission clauses in the
2996 <Permissions> element.

2997 The <PermittedDate> sub-element identifies an individual set of dates between which application are
2998 permitted use of the symmetric key in question. The <PermittedDate> element consists of the following
2999 sub-elements:

- 3000 1. The <StartDate> element identifies the date from which applications MAY start using the
3001 symmetric key in question. It is an XSD *Date* type that MUST be specified in a specific pattern
3002 (see examples) where the first four digits specify the year, the second two digits specify the

3003 calendar month in the year, and the last two digits specify the calendar date of the month.

3004
3005 There SHALL be only one <StartDate> element within a <PermittedDate> element.

3006
3007 Conforming **SKCL** implementations SHALL NOT start using the symmetric before the onset of
3008 the <StartDate> on the client machine. Unless further constrained by the <PermittedTimes>
3009 element, the onset of the <StartDate> is specified to be the first second of the day – 00:00:01
3010 Hours – on the client machine.

3011 2. The <EndDate> element identifies the date until which applications may use the symmetric key in
3012 question. It is an XSD **Date** type that MUST be specified in a specific pattern (see examples)
3013 where the first four digits specify the year, the second two digits specify the calendar month in the
3014 year, and the last two digits specify the calendar date of the month.

3015
3016 There SHALL be only one <EndDate> element within a <PermittedDate> element.

3017
3018 Conforming **SKCL** implementations SHALL NOT use the symmetric after the end of the
3019 <EndDate> on the client machine. Unless further constrained by the <PermittedTimes>
3020 element, the end of the <EndDate> is specified to be the last second of the day – 23:59:59 Hours
3021 – on the client machine.

3022 Examples of the <PermittedDates> element are shown below; other required parts of their enclosing
3023 elements are not shown for brevity:

3024 **Example 1 – An example of a <PermittedDates> element with a single<PermittedDate> element.**
3025 **The <StartDate> specifies January 01, 2009 while the <EndDate> specifies January 31, 2009:**

```
3026 <ekmi:PermittedDates ekmi:any="false">  
3027 <ekmi:PermittedDate>  
3028 <ekmi:StartDate>2009-01-01</ekmi:StartDate>  
3029 <ekmi:EndDate>2009-01-31</ekmi:EndDate>  
3030 </ekmi:PermittedDate>  
3031 </ekmi:PermittedDates>
```

3032 **Example 2 – An example of a <PermittedDates> element with two <PermittedDate> elements. For**
3033 **the first <PermittedDate> element, the <StartDate> element specifies July 01, 2008 while the**
3034 **<EndDate> element specifies July 03, 2008. For the second <PermittedDate> element, the**
3035 **<StartDate> element specifies July 07, 2008 while the <EndDate> element specifies July 12, 2008.**
3036 **This policy would restrict a symmetric key with this <PermittedDates> element so it cannot be**
3037 **used on the July 4th weekend of 2008:**

```
3038 <ekmi:PermittedDates ekmi:any="false">  
3039 <ekmi:PermittedDate>  
3040 <ekmi:StartDate>2008-07-01</ekmi:StartDate>  
3041 <ekmi:EndDate>2008-07-03</ekmi:EndDate>  
3042 </ekmi:PermittedDate>  
3043 <ekmi:PermittedDate>  
3044 <ekmi:StartDate>2008-07-07</ekmi:StartDate>  
3045 <ekmi:EndDate>2009-07-12</ekmi:EndDate>  
3046 </ekmi:PermittedDate>  
3047 </ekmi:PermittedDates>
```

3048 **Example 3 – An example of a null <PermittedDates> element, specifying that applications are**
3049 **permitted use of the symmetric key on any date:**

```
3050 <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
```

3051

3052

3053 4.15 Element <PermittedDays> and <PermittedDay>

3054 The element <PermittedDays> , of the type **PermittedDaysType** and its only child-element
3055 <PermittedDay> of the **PermittedDayType**, are used to define days of the week on which applications
3056 are permitted to use a symmetric key within a specific <Symkey> element.

3057 Schema Definition:

```
3058 <xsd:complexType name="PermittedDaysType">
3059   <xsd:sequence>
3060     <xsd:element
3061       name="PermittedDay"
3062       type="tns:PermittedDayType"
3063       minOccurs="0"
3064       maxOccurs="unbounded">
3065     </xsd:element>
3066   </xsd:sequence>
3067   <xsd:attribute ref="tns:any" use="required"/>
3068 </xsd:complexType>
```

3069 Schema Definition:

```
3070 <xsd:simpleType name="PermittedDayType">
3071   <xsd:restriction base="xsd:string">
3072     <xsd:enumeration value="Sunday"/>
3073     <xsd:enumeration value="Monday"/>
3074     <xsd:enumeration value="Tuesday"/>
3075     <xsd:enumeration value="Wednesday"/>
3076     <xsd:enumeration value="Thursday"/>
3077     <xsd:enumeration value="Friday"/>
3078     <xsd:enumeration value="Saturday"/>
3079     <xsd:enumeration value="Weekday"/>
3080     <xsd:enumeration value="Weekend"/>
3081   </xsd:restriction>
3082 </xsd:simpleType>
```

3083 There SHALL be only one <PermittedDays> element within the <Permissions> element. However,
3084 there MAY be an unbounded (unlimited) number of <PermittedDay> elements within a
3085 <PermittedDays> element.

3086 The <PermittedDays> element SHALL have one attribute named "any", that will have a "false" or "true"
3087 value, based on the following:

- 3088 • When the <PermittedDays> element is null (i.e. it does not have a single <PermittedDay> sub-
3089 element in it), the value of the "any" attribute SHALL be set to "true" AND the XML Schema
3090 Instance (XSI) "nil" attribute SHALL be set to "true".
- 3091 • When the <PermittedDays> element is not-null (i.e. it has at least one <PermittedDay> sub-
3092 element in it), the value of the "any" attribute SHALL be set to "false" AND the XML Schema
3093 Instance (XSI) "nil" attribute SHALL NOT be present.

3094 A null <PermittedDays> element specifies that applications are permitted use of the symmetric key on all
3095 days of the week, subject to complying with all other permission clauses in the <Permissions> element.

3096 The <PermittedDay> element, of the XSD **String** type, identifies individual days of the week from an
3097 enumerated list on which application are permitted to use the symmetric key in question.

3098 Examples of the <PermittedDays> element are shown below; other parts of their enclosing elements are
3099 not shown for brevity:

3100 **Example 1 – An example of a <PermittedDays> element with a single<PermittedDay> child**
3101 **element, specifying that the symmetric key may be used only on weekdays:**

```
3102     <ekmi:PermittedDays ekmi:any="false">  
3103         <ekmi:PermittedDay>Weekday </ekmi:PermittedDay>  
3104     </ekmi:PermittedDays>
```

3105 **Example 2 – An example of a <PermittedDays> element with three <PermittedDay> child elements,**
3106 **specifying that the symmetric key may be used only on Mondays, Wednesdays and Fridays:**

```
3107     <ekmi:PermittedDays ekmi:any="false">  
3108         <ekmi:PermittedDay>Monday</ekmi:PermittedDay>  
3109         <ekmi:PermittedDay>Wednesday</ekmi:PermittedDay>  
3110         <ekmi:PermittedDay>Friday</ekmi:PermittedDay>  
3111     </ekmi:PermittedDays>
```

3112 **Example 3 – An example of a null <PermittedDays> element, specifying that the symmetric key**
3113 **may be used on any day of the week:**

```
3114     <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
```

3115

3116 4.16 Element <PermittedDuration>

3117

3118 The element <PermittedDuration>, of the type *PermittedDurationType* is used to define the number
3119 of seconds, applications are permitted to use a symmetric key, once the **SKCL** has started using the
3120 symmetric key in question.

3121 Schema Definition:

```
3122     <xsd:complexType name="PermittedDurationType">  
3123         <xsd:simpleContent>  
3124             <xsd:extension base="tns:DurationType">  
3125                 <xsd:attribute ref="tns:any" use="required"/>  
3126             </xsd:extension>  
3127         </xsd:simpleContent>  
3128     </xsd:complexType>
```

3129 Schema Definition:

```
3130     <xsd:simpleType name="DurationType">  
3131         <xsd:restriction base="xsd:positiveInteger">  
3132             <xsd:minInclusive value="1"/>  
3133             <xsd:maxInclusive value="18446744073709551615"/>  
3134         </xsd:restriction>  
3135     </xsd:simpleType>
```

3136 There SHALL be only one <PermittedDuration> element within the <Permissions> element.

3137 The <PermittedDuration> element SHALL have one attribute, named “any” that will have a “false” or
3138 “true” value, based on the following:

- 3139 • When the <PermittedDuration> element is null (i.e. it does not have any content in it), the value
3140 of the “any” attribute SHALL be set to “true” AND the XML Schema Instance (XSI) “nil” attribute
3141 SHALL be set to “true”.

- 3142 • When the <PermittedDuration> element is not-null (i.e. it has content in it), the value of the
3143 “any” attribute SHALL be set to “false” AND the XML Schema Instance (XSI) “nil” attribute SHALL
3144 NOT be present.

3145 A null <PermittedDuration> element specifies that applications are permitted use of the symmetric key
3146 indefinitely, subject to complying with all other permission clauses in the <Permissions> element.

3147 The <PermittedDuration> element, of the XSD *positiveInteger* type, identifies the number of seconds
3148 for which the symmetric key in question may be used, ONCE the key has been used by conforming **SKCL**
3149 implementations for the first time. The values for <PermittedDuration> may range between 1 and
3150 18446744073709551615.

3151 As long as the symmetric has not been used by an **SKCL** on a client device (it might be cached for many
3152 days/weeks/months depending on the <KeyCachePolicy> in effect within the SKMS for that device) the
3153 effective lifetime of the symmetric key may be well past the number of seconds specified in
3154 <PermittedDuration> when calculated from the time of the key's generation time on the **SKS** server. It
3155 is the responsibility of the **SKCL** implementation, when presented with a <PermittedDuration> element
3156 in a <KeyUsePolicy> of a symmetric key, to keep track of the date/time when the symmetric key in
3157 question was first used on the client device, and how long the key will last after that.

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

3160 **Example 1 – An example of a <PermittedDuration> element specifying that the symmetric key**
3161 **may be used only for a single 24-hour period from the moment it is first used by an SKCL:**

3162 `<ekmi:PermittedDuration ekmi:any="false">86400</ekmi:PermittedDuration>`

3163 **Example 2 – An example of a <PermittedDuration> element specifying that the symmetric key**
3164 **may be used only for week from the moment it is first used by an SKCL:**

3165 `<ekmi:PermittedDuration ekmi:any="false">604800</ekmi:PermittedDuration>`

3166 **Example 3 – An example of a <PermittedDuration> element specifying that the symmetric key**
3167 **may be used only 5 minutes from the moment it is first used by an SKCL:**

3168 `<ekmi:PermittedDuration ekmi:any="false">300</ekmi:PermittedDuration>`

3169 **Example 4 – An example of a null <PermittedDuration> element specifying that the symmetric key**
3170 **may be used indefinitely by an SKCL:**

3171 `<ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>`

3172

3173

3174 **4.17 Element <PermittedLevels> and <PermittedLevel>**

3175

3176 The element <PermittedLevels>, of the type *LevelClassificationType*, is used to define the security
3177 level at which applications are permitted use of a symmetric key. This element is useful only to
3178 applications and systems that conform to the multi-level security (MLS) system as defined in the Bell-
3179 LaPadula model.

3180 **Schema Definition:**

```
3181 <xsd:complexType name="PermittedLevelsType">  
3182 <xsd:sequence>  
3183 <xsd:element
```

```

3184         name="PermittedLevel"
3185         type="tns:LevelClassificationType"
3186         minOccurs="0"
3187         maxOccurs="unbounded">
3188     </xsd:element>
3189     <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
3190 </xsd:sequence>
3191 <xsd:attribute ref="tns:any" use="required"/>
3192 </xsd:complexType>

```

3193 **Schema Definition:**

```

3194 <xsd:simpleType name="LevelClassificationType">
3195     <xsd:restriction base="xsd:string">
3196         <xsd:enumeration value="Unclassified"/>
3197         <xsd:enumeration value="Confidential"/>
3198         <xsd:enumeration value="Secret"/>
3199         <xsd:enumeration value="Top-Secret"/>
3200     </xsd:restriction>
3201 </xsd:simpleType>

```

3202 There SHALL be only one <PermittedLevels> element within the <Permissions> element. However,
3203 there MAY be an unbounded (unlimited) number of <PermittedLevel> elements within the
3204 <PermittedLevels> element. (Practically, it makes no sense to have more than the known levels;
3205 however, this specification leaves itself open to the possibility that other levels may be defined).

3206 The <PermittedLevels> element SHALL have one attribute named "any", that will have a "false" or "true"
3207 value, based on the following:

- 3208 • When the <PermittedLevels> element is null (i.e. it does not have a single <PermittedLevel>
3209 sub-element in it), the value of the "any" attribute SHALL be set to "true" AND the XML Schema
3210 Instance (XSI) "nil" attribute SHALL be set to "true".
- 3211 • When the <PermittedLevels> element is not-null (i.e. it has at least one <PermittedLevel>
3212 sub-element in it), the value of the "any" attribute SHALL be set to "false" AND the XML Schema
3213 Instance (XSI) "nil" attribute SHALL NOT be present.

3214 A null <PermittedLevels> element specifies that applications at ANY level are permitted use of the
3215 symmetric key, subject to complying with all other permission clauses in the <Permissions> element.

3216 The <PermittedLevel> sub-element, of the **LevelClassificationType**, identifies the precise MLS level
3217 at which the symmetric key in question may be used. The <PermittedLevel> SHALL contain one of the
3218 following four (4) enumerated values:

- 3219 1. Unclassified
- 3220 2. Confidential
- 3221 3. Secret
- 3222 4. Top-Secret

3223 Examples of the <PermittedLevels> element are shown below; other parts of their enclosing elements
3224 are not shown for brevity:

3225 **Example 1 – An example of a <PermittedLevels> element specifying that the symmetric key may
3226 be used only by applications at the Confidential level:**

```

3227 <ekmi:PermittedLevels ekmi:any="false">
3228     <ekmi:PermittedLevel>Confidential</ekmi:PermittedLevel>
3229 </ekmi:PermittedLevels>

```

3230 **Example 2 – An example of a <PermittedLevels> element specifying that the symmetric key may**
3231 **be used only by applications at the Secret or Top-Secret level:**

```
3232     <ekmi:PermittedLevels ekmi:any="false">  
3233         <ekmi:PermittedLevel>Secret</ekmi:PermittedLevel>  
3234         <ekmi:PermittedLevel>Top-Secret</ekmi:PermittedLevel>  
3235     </ekmi:PermittedLevels>
```

3236 **Example 3 – An example of a null <PermittedLevels> element specifying that the symmetric key**
3237 **may be used at any level:**

```
3238     <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
```

3239

3240

3241 **4.18 Element <PermittedLocations> and <PermittedLocation>**

3242

3243 The element <PermittedLocations>, of the type *PermittedLocationsType*, is used to define the
3244 geographically physical locations where applications are permitted use of a symmetric key. This element
3245 is useful only to applications that have the ability to determine the Global Positioning System (GPS)
3246 location of the client device intending to use the symmetric key.

3247 **Schema Definition:**

```
3248     <xsd:complexType name="PermittedLocationsType">  
3249         <xsd:sequence>  
3250             <xsd:element name="PermittedLocation" minOccurs="1" maxOccurs="unbounded">  
3251                 <xsd:complexType>  
3252                     <xsd:sequence>  
3253                         <xsd:element name="LocationName">  
3254                             <xsd:simpleType>  
3255                                 <xsd:restriction base="xsd:string">  
3256                                     <xsd:maxLength value="256"/>  
3257                                     <xsd:whiteSpace value="preserve"/>  
3258                                 </xsd:restriction>  
3259                             </xsd:simpleType>  
3260                         </xsd:element>  
3261                         <xsd:group  
3262                             ref="tns:LocationCoordinateGroup"  
3263                             minOccurs="0"  
3264                             maxOccurs="unbounded"/>  
3265                         <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>  
3266                     </xsd:sequence>  
3267                 </xsd:complexType>  
3268             </xsd:element>  
3269         </xsd:sequence>  
3270     <xsd:attribute ref="tns:any" use="required"/>  
3271 </xsd:complexType>
```

3272 **Schema Definition:**

```
3273     <xsd:group name="LocationCoordinateGroup">  
3274         <xsd:sequence>  
3275             <xsd:element name="Latitude">  
3276                 <xsd:simpleType>  
3277                     <xsd:restriction base="xsd:decimal">
```



```

3278         <xsd:totalDigits value="10"/>
3279         <xsd:fractionDigits value="7"/>
3280     </xsd:restriction>
3281 </xsd:simpleType>
3282 </xsd:element>
3283 <xsd:element name="Longitude">
3284     <xsd:simpleType>
3285         <xsd:restriction base="xsd:decimal">
3286             <xsd:totalDigits value="10"/>
3287             <xsd:fractionDigits value="7"/>
3288         </xsd:restriction>
3289     </xsd:simpleType>
3290 </xsd:element>
3291 </xsd:sequence>
3292 </xsd:group>

```

3293 There SHALL be only one <PermittedLocations> element within the <Permissions> element.
3294 However, there MAY be an unbounded (unlimited) number of <PermittedLocation> sub-elements within
3295 the <PermittedLocations> element.

3296 The <PermittedLocations> element SHALL have one attribute named "any", that will have a "false" or
3297 "true" value, based on the following:

- 3298 • When the <PermittedLocations> element is null (i.e. it does not have a single
3299 <PermittedLocation> sub-element in it), the value of the "any" attribute SHALL be set to "true"
3300 AND the XML Schema Instance (XSI) "nil" attribute SHALL be set to "true".
- 3301 • When the <PermittedLocations> element is not-null (i.e. it has at least one
3302 <PermittedLocation> sub-element in it), the value of the "any" attribute SHALL be set to "false"
3303 AND the XML Schema Instance (XSI) "nil" attribute SHALL NOT be present.

3304 A null <PermittedLocations> element specifies that applications are permitted use of the symmetric key
3305 at ANY physical location, subject to complying with all other permission clauses in the <Permissions>
3306 element.

3307 The <PermittedLocation> element, of the **PermittedLocationType**, identifies the precise geographical
3308 location where the symmetric key in question may be used. The <PermittedLocation> SHALL contain
3309 the following elements:

- 3310 1. The <LocationName> element identifies a human-readable name of the physical location. It is
3311 an XSD **String** type element, with a maximum length of 256 characters.

3312 There SHALL be only one <LocationName> element within a <PermittedLocation> element.
3313

- 3314 2. An optional **LocationCoordinateGroup** which, when present, SHALL contain the following two
3315 elements:

- 3316 a) The <Latitude> element of XSD **Decimal** type, that identifies the horizontal coordinate
3317 location of the client device on the Earth, measured in *degrees* and expressed as a
3318 decimal with the *minutes* and *seconds* part of the measurement expressed as a single
3319 fraction.

3320 When used, there SHALL be only one <Latitude> element within the
3321 <PermittedLocation> element.
3322

- 3323 b) The <Longitude> element of XSD **Decimal** type, that identifies the vertical coordinate
3324 location of the client device on the Earth, measured in *degrees* and expressed as a
3325 decimal with the *minutes* and *seconds* part of the measurement expressed as a single
3326 fraction.
3327

3328 When used, there SHALL be only one <Longitude> element within the
3329 <PermittedLocation> element.

3330 Some examples of the <PermittedLocations> element are shown below; other parts of their enclosing
3331 elements are not shown for brevity:

3332 **Example 1 – An example of a <PermittedLocations> element specifying that the symmetric key**
3333 **may be used only by applications at a single named location:**

```
3334       <ekmi:PermittedLocations ekmi:any="false">  
3335           <ekmi:PermittedLocation>  
3336               <ekmi:LocationName>StrongAuth Server Room</ekmi:LocationName>  
3337           </ekmi:PermittedLocation>  
3338       </ekmi:PermittedLocations>
```

3339 **Example 2 – An example of a <PermittedLocations> element specifying that the symmetric key**
3340 **may be used only by applications at a single location at the given GPS coordinates:**

```
3341       <ekmi:PermittedLocations ekmi:any="false">  
3342           <ekmi:PermittedLocation>  
3343               <ekmi:LocationName>StrongAuth Server Room</ekmi:LocationName>  
3344               <ekmi:Latitude>37.385653 </ekmi:Latitude>  
3345               <ekmi:Longitude>-121.993192 </ekmi:Longitude>  
3346           </ekmi:PermittedLocation>  
3347       </ekmi:PermittedLocations>
```

3348 **Example 3 – An example of a <PermittedLocations> element specifying that the symmetric key**
3349 **may be used only by applications at multiple locations:**

```
3350       <ekmi:PermittedLocations ekmi:any="false">  
3351           <ekmi:PermittedLocation>  
3352               <ekmi:LocationName>Humongous Headquarters</ekmi:LocationName>  
3353           </ekmi:PermittedLocation>  
3354           <ekmi:PermittedLocation>  
3355               <ekmi:LocationName> Humongous Primary Data Center</ekmi:LocationName>  
3356               <ekmi:Latitude>37.385653 </ekmi:Latitude>  
3357               <ekmi:Longitude>-121.993192 </ekmi:Longitude>  
3358           </ekmi:PermittedLocation>  
3359           <ekmi:PermittedLocation>  
3360               <ekmi:LocationName>Humongous DR Data Center</ekmi:LocationName>  
3361               <ekmi:Latitude>68.845901 </ekmi:Latitude>  
3362               <ekmi:Longitude>11.393385 </ekmi:Longitude>  
3363           </ekmi:PermittedLocation>  
3364       </ekmi:PermittedLocations>
```

3365 **Example 4 – An example of a null <PermittedLocations> element specifying that the symmetric**
3366 **key may be used at any location on the planet:**

```
3367       <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
```

3368 **4.19 Element <PermittedNumberOfTransactions>**

3369 The element <PermittedNumberOfTransactions>, of type *PermittedNumberOfTransactionsType* is
3370 used to define the number of *encryption* transactions that applications are permitted with a symmetric
3371 key within a specific <Symkey> element, once the **SKCL** has started using the symmetric key in question.
3372 It does not limit the number of *decryption* transactions with the same symmetric key.

3373 **Schema Definition:**

```
3374 <xsd:complexType name="PermittedNumberOfTransactionsType">
3375   <xsd:simpleContent>
3376     <xsd:extension base="tns:NumberOfTransactionsType">
3377       <xsd:attribute ref="tns:any" use="required"/>
3378     </xsd:extension>
3379   </xsd:simpleContent>
3380 </xsd:complexType>
```

3381 **Schema Definition:**

```
3382 <xsd:simpleType name="NumberOfTransactionsType">
3383   <xsd:restriction base="xsd:positiveInteger">
3384     <xsd:minInclusive value="1"/>
3385     <xsd:maxInclusive value="18446744073709551615"/>
3386   </xsd:restriction>
3387 </xsd:simpleType>
```

3388 There SHALL be only one <PermittedNumberOfTransactions> element within the <Permissions>
3389 element.

3390 The <PermittedNumberOfTransactions> element SHALL have one attribute named “any”, that will have
3391 a “false” or “true” value, based on the following:

- 3392 • When the <PermittedNumberOfTransactions> element is null (i.e. it does not have any content
3393 in it), the value of the “any” attribute SHALL be set to “true” AND the XML Schema Instance (XSI)
3394 “nil” attribute SHALL be set to “true”.
- 3395 • When the <PermittedNumberOfTransactions> element is not-null (i.e. it has a positive integer
3396 content in it), the value of the “any” attribute SHALL be set to “false” AND the XML Schema
3397 Instance (XSI) “nil” attribute SHALL NOT be present.

3398 A null <PermittedNumberOfTransactions> element specifies that applications are permitted use of the
3399 symmetric key for an unlimited number of encryption transactions, subject to complying with all other
3400 permission clauses in the <Permissions> element.

3401 The value of <PermittedNumberOfTransactions> element, of the XSD *positiveInteger* type, MAY
3402 range between 1 and 18446744073709551615.

3403 Some examples of the <PermittedNumberOfTransactions> element are shown below; other parts of
3404 their enclosing elements are not shown for brevity:

3405 **Example 1 – An example of a <PermittedNumberOfTransactions> element specifying that the**
3406 **symmetric key may be used only for a single encryption transaction by an SKCL:**

```
3407 <ekmi:PermittedNumberOfTransactions ekmi:any="false">
3408   1
3409 </ekmi:PermittedNumberOfTransactions>
```

3410 **Example 2 – An example of a <PermittedNumberOfTransactions> element specifying that the**
3411 **symmetric key may be used only for 100 transactions by an SKCL:**

```
3412 <ekmi:PermittedNumberOfTransactions ekmi:any="false">
3413   100
3414 </ekmi:PermittedNumberOfTransactions>
```

3415 **Example 3 – An example of a null <PermittedNumberOfTransactions> element specifying that the**
3416 **symmetric key may be used for an unlimited number of encryption transactions by an SKCL:**

```
3417 <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
```

3418

3419

3420 4.20 Element <PermittedTimes> and <PermittedTime>

3421

3422 The element <PermittedTimes>, of the type *PermittedTimesType* and its only child-element
3423 <PermittedTime>, which is an anonymous XSD *ComplexType*, are used to define sets of times during
3424 the day between which applications are permitted to use a symmetric key within a specific <Symkey>
3425 element.

3426 Schema Definition:

```
3427 <xsd:complexType name="PermittedTimesType">
3428   <xsd:sequence>
3429     <xsd:element name="PermittedTime" minOccurs="0" maxOccurs="unbounded">
3430       <xsd:complexType>
3431         <xsd:sequence>
3432           <xsd:element name="StartTime">
3433             <xsd:simpleType>
3434               <xsd:restriction base="xsd:time">
3435                 <xsd:pattern value="\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}"/>
3436               </xsd:restriction>
3437             </xsd:simpleType>
3438           </xsd:element>
3439           <xsd:element name="EndTime">
3440             <xsd:simpleType>
3441               <xsd:restriction base="xsd:time">
3442                 <xsd:pattern value="\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}"/>
3443               </xsd:restriction>
3444             </xsd:simpleType>
3445           </xsd:element>
3446         </xsd:sequence>
3447       </xsd:complexType>
3448     </xsd:element>
3449   </xsd:sequence>
3450   <xsd:attribute ref="tns:any" use="required"/>
3451 </xsd:complexType>
```

3452 There SHALL be only one <PermittedTimes> element within the <Permissions> element. However,
3453 there MAY be an unbounded (unlimited) number of <PermittedTime> sub-elements within a
3454 <PermittedTimes> element.

3455 The <PermittedTimes> element SHALL have one attribute named "any", that will have a "false" or "true"
3456 value, based on the following:

- 3457 • When the <PermittedTimes> element is null (i.e. it does not have a single <PermittedTime>
3458 sub-element in it), the value of the "any" attribute SHALL be set to "true" AND the XML Schema
3459 Instance (XSI) "nil" attribute SHALL be set to "true".
- 3460 • When the <PermittedTimes> element is not-null (i.e. it has at least one <PermittedTime> sub-
3461 element in it), the value of the "any" attribute SHALL be set to "false" AND the XML Schema
3462 Instance (XSI) "nil" attribute SHALL NOT be present.

3463 A null <PermittedTimes> element specifies that applications are permitted use of the symmetric key at
3464 ANY time of the day or night, subject to complying with all other permission clauses in the
3465 <Permissions> element.

3466 The <PermittedTime> sub-element identifies an individual set of times between which application are
3467 permitted to use the symmetric key in question. The <PermittedTime> element consists of the following
3468 sub-elements:

- 3469 1. The <StartTime> element identifies the date from which applications may start using the
3470 symmetric key in question. It is an XSD *Time* type that MUST be specified in a specific pattern
3471 (see examples) where the first two digits specify the hour, the second two digits specify the
3472 minutes and the last two digits specify the seconds in a 24 hour format.

3473 There SHALL be only one <StartTime> element within a <PermittedTime> element.

3474 Conforming **SKCL** implementations SHALL NOT start using the symmetric before the onset of
3475 the <StartTime> on the client machine.

- 3476 2. The <EndTime> element identifies the time until which applications may use the symmetric key in
3477 question. It is an XSD *Time* type that MUST be specified in a specific pattern (see examples)
3478 where the first two digits specify the hour, the second two digits specify the minutes and the last
3479 two digits specify the seconds in a 24 hour format.

3480 There SHALL be only one <EndTime> element within a <PermittedTime> element.

3481 Conforming **SKCL** implementations SHALL NOT use the symmetric after the end of the
3482 <EndTime> on the client machine.

3483 Some examples of the <PermittedTimes> element are shown below; other parts of their enclosing
3484 elements are not shown for brevity:

3485 **Example 1 – An example of a <PermittedTimes> element with a single<PermittedTime> element.**
3486 **The <StartTime> specifies 9:00AM on the client machine while the <EndTime> specifies 5:00PM:**

```
3491 <ekmi:PermittedTimes ekmi:any="false">  
3492 <ekmi:PermittedTime>  
3493 <ekmi:StartTime>09:00:00</ekmi:StartTime>  
3494 <ekmi:EndTime>17:00:00</ekmi:EndTime>  
3495 </ekmi:PermittedTime>  
3496 </ekmi:PermittedTimes>
```

3497 **Example 2 – An example of a <PermittedTimes> element with two <PermittedTime> elements. For**
3498 **the first <PermittedTime> element , the <StartTime> element specifies 6:00AM while the**
3499 **<EndTime> element specifies 12:00 Noon. For the second <PermittedTime> element, the**
3500 **<StartTime> element specifies 3:00 PM in the afternoon, while the <EndTime> element specifies**
3501 **7:00PM in the evening. This policy might imply that a symmetric key with this <PermittedTimes>**
3502 **element cannot be used during a lunch break of 12:00 Noon to 3:00PM:**

```
3503 <ekmi:PermittedTimes ekmi:any="false">  
3504 <ekmi:PermittedTime>  
3505 <ekmi:StartTime>06:00:00</ekmi:StartTime>  
3506 <ekmi:EndTime>12:00:00</ekmi:EndTime>  
3507 </ekmi:PermittedTime>  
3508 <ekmi:PermittedTime>  
3509 <ekmi:StartTime>15:00:00</ekmi:StartTime>  
3510 <ekmi:EndTime>19:00:00</ekmi:EndTime>  
3511 </ekmi:PermittedTime>  
3512 </ekmi:PermittedTimes>
```

3513 **Example 3 – An example of a null <PermittedTimes> element, specifying that the key may be used**
3514 **at any time:**

```
3515 <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
```

3516

3517 4.21 Element <PermittedUses> and <PermittedUse>

3518

3519 The element <PermittedUses>, of the type *PermittedUsesType*, is used to define the specific ways in
3520 which applications are permitted to use a symmetric key within a specific <Symkey> element.

3521 Schema Definition:

```
3522 <xsd:complexType name="PermittedUsesType" mixed="true">
3523   <xsd:sequence>
3524     <xsd:element name="PermittedUse" minOccurs="0" maxOccurs="unbounded">
3525       <xsd:simpleType>
3526         <xsd:restriction base="xsd:string">
3527           <xsd:maxLength value="256"/>
3528           <xsd:whiteSpace value="preserve"/>
3529         </xsd:restriction>
3530       </xsd:simpleType>
3531     </xsd:element>
3532     <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
3533   </xsd:sequence>
3534   <xsd:attribute ref="tns:any" use="required"/>
3535 </xsd:complexType>
```

3536 There SHALL be only one <PermittedUses> element within the <Permissions> element. However,
3537 there MAY be an unbounded (unlimited) number of <PermittedUse> sub-elements within the
3538 <PermittedUses> element.

3539 The <PermittedUses> element SHALL have one attribute named "any", that will have a "false" or "true"
3540 value, based on the following:

- 3541 • When the <PermittedUses> element is null (i.e. it does not have a single <PermittedUse> sub-
3542 element in it), the value of the "any" attribute SHALL be set to "true" AND the XML Schema
3543 Instance (XSI) "nil" attribute SHALL be set to "true".
- 3544 • When the <PermittedUses> element is not-null (i.e. it has at least one <PermittedUse> sub-
3545 element in it), the value of the "any" attribute SHALL be set to "false" AND the XML Schema
3546 Instance (XSI) "nil" attribute SHALL NOT be present.

3547 A null <PermittedUses> element specifies that applications are permitted use of the symmetric key for
3548 ANY purpose, subject to complying with all other permission clauses in the <Permissions> element.

3549 Examples of the <PermittedUses> element are shown below; other parts of their enclosing elements are
3550 not shown for brevity:

3551 **Example 1 – An example of a <PermittedUses> element specifying that the symmetric key may be
3552 used only by VPN applications for session encryption keys:**

```
3553 <ekmi:PermittedUses ekmi:any="false">
3554   <ekmi:PermittedUse>VPN</ekmi:PermittedUse>
3555 </ekmi:PermittedUses>
```

3556 **Example 2 – An example of a <PermittedUses> element specifying that the symmetric key may be
3557 used only by applications on laptops and Personal Digital Assistants (PDA):**

```
3558 <ekmi:PermittedUses ekmi:any="false">
3559   <ekmi:PermittedUse>Laptop</ekmi:PermittedUse>
```

3560 <ekmi:PermittedUse>PDA</ekmi:PermittedUse>
3561 </ekmi:PermittedUses>

3562 **Example 3 – An example of a null <PermittedUses> element specifying that the symmetric key**
3563 **may be used for any purpose:**

3564 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>

3565 **4.22 Element <KeyCachePolicyRequest>**

3566 The <KeyCachePolicyRequest> element is used to request a key-cache policy from the **SKS** server , so
3567 the client may know if and how to cache symmetric keys locally.

3568 While it is a top-level element within this specification, a <SymkeyRequest> element MUST be enclosed
3569 within a **SOAP Body** element of a **SOAP Envelope** to conform to the security requirements of this
3570 specification. The **SOAP Header** of the **SOAP Envelope** MUST enclose a **Security** element conforming
3571 to **[WSS]** with a **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
3572 requirements of the specified security profile in **[WSS]** to form a well-formed, secure message.
3573

3574 **Schema Definition:**

```
3575       <xsd:element name="KeyCachePolicyRequest">  
3576           <xsd:complexType>  
3577            <xsd:annotation>  
3578             <xsd:documentation>  
3579              No elements/attributes are defined for KeyCachePolicyRequest.  
3580             </xsd:documentation>  
3581            </xsd:annotation>  
3582           </xsd:complexType>  
3583       </xsd:element>
```

3584 The <KeyCachePolicyRequest> has no child elements. The SOAP Header of the signed request
3585 provides the **SKS** server with all the information it needs to process the request: the identity of the
3586 requester, strong authentication and message integrity of the request.

3587 Some examples of the use of the <SymkeyRequest> element are as follows:

3588 **Example 1 – An example of a <KeyCachePolicyRequest>; the surrounding SOAP envelope is not**
3589 **displayed here for brevity:**

3590 <ekmi:KeyCachePolicyRequest
3591 xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01"/>

3592

3593 **4.23 Element <KeyCachePolicyResponse>**

3594 The <KeyCachePolicyResponse> element is the response sent by an **SKS** Server to a client that
3595 requested a key-cache policy through a <KeyCachePolicyRequest>. The <KeyCachePolicyResponse>
3596 contains policy elements, which define rules that conforming implementations of the **SKCL** MUST
3597 adhere to when caching symmetric keys sent by the **SKS** Server.

3598 **Schema Definition:**

```
3599       <xsd:element name="KeyCachePolicyResponse">  
3600           <xsd:complexType>  
3601            <xsd:sequence>  
3602             <xsd:element
```

```

3603         name="KeyCachePolicy"
3604         type="ekmi:KeyCachePolicyType"
3605         minOccurs="1" maxOccurs="unbounded"/>
3606     </xsd:sequence>
3607 </xsd:complexType>
3608 </xsd:element>

```

3609 The <KeyCachePolicyResponse> element consists of a minimum of one, but an unbounded (unlimited)
3610 number of <KeyCachePolicy> children elements.

3611 4.24 Element <KeyCachePolicy>

3612

3613 The <KeyCachePolicy> element contains policy elements, which define rules that conforming
3614 implementations of the **SKCL** MUST adhere to when caching symmetric keys sent by the **SKS** Server.

3615 Schema Definition:

```

3616 <xsd:element name="KeyCachePolicyResponse">
3617   <xsd:complexType>
3618     <xsd:sequence>
3619       <xsd:element
3620         name="KeyCachePolicy"
3621         type="ekmi:KeyCachePolicyType"
3622         minOccurs="1" maxOccurs="unbounded"/>
3623     </xsd:sequence>
3624   </xsd:complexType>
3625 </xsd:element>

3626 <xsd:complexType name="KeyCachePolicyType" mixed="true">
3627   <xsd:sequence>
3628     <xsd:element name="KeyCachePolicyID" type="tns:TwoPartIDType"/>
3629     <xsd:element name="PolicyName">
3630       <xsd:simpleType>
3631         <xsd:restriction base="xsd:string">
3632           <xsd:maxLength value="255"/>
3633           <xsd:whiteSpace value="preserve"/>
3634         </xsd:restriction>
3635       </xsd:simpleType>
3636     </xsd:element>
3637     <xsd:element name="Description" nillable="true">
3638       <xsd:simpleType>
3639         <xsd:restriction base="xsd:string">
3640           <xsd:maxLength value="2048"/>
3641           <xsd:whiteSpace value="preserve"/>
3642         </xsd:restriction>
3643       </xsd:simpleType>
3644     </xsd:element>
3645     <xsd:element name="KeyClass" type="tns:KeyClassType"/>
3646     <xsd:element name="StartDate" type="xsd:dateTime"/>
3647     <xsd:element name="EndDate" type="xsd:dateTime" nillable="true"/>
3648     <xsd:element name="PolicyCheckInterval">
3649       <xsd:simpleType>
3650         <xsd:restriction base="xsd:nonNegativeInteger">
3651           <xsd:minInclusive value="0"/>
3652           <xsd:maxInclusive value="2592000"/>
3653         </xsd:restriction>
3654       </xsd:simpleType>
3655     </xsd:element>

```



```

3656         <xsd:element name="Status" type="tns:StatusType"/>
3657     <xsd:element
3658         name="NewKeysCacheDetail"
3659         type="tns:KeyCacheDetailType"
3660         minOccurs="0"/>
3661     <xsd:element
3662         name="UsedKeysCacheDetail"
3663         type="tns:KeyCacheDetailType"
3664         minOccurs="0"/>
3665 </xsd:sequence>
3666 </xsd:complexType>

```

3667 The <KeyCachePolicy> element is of the **KeyCachePolicyType** and consists of the following child
3668 elements:

3669 1. <KeyCachePolicyID> [Required]

3670

3671 The <KeyCachePolicyID> element, of type **TwoPartIDType**, identifies the unique policy object
3672 within the **SKMS**. There SHALL be only one <KeyCachePolicyID> element within a
3673 <KeyCachePolicy> element.

3674

3675 The **TwoPartIDType** is specified in Section 4.8.

3676 2. <PolicyName> [Required]

3677

3678 The <PolicyName> element, of type XSD **String**, with a maximum length of 255 characters,
3679 identifies a unique name given to this <KeyCachePolicy>. There SHALL be only one
3680 <PolicyName> element within a <KeyCachePolicy> element.

3681 3. <Description> [Required]

3682

3683 The <Description> element, of type XSD **String**, with a maximum length of 2048 characters,
3684 provides a human-readable description of this policy. There SHALL be only one <Description>
3685 element within a <KeyCachePolicy> element.

3686

3687 The <Description> MAY be an empty element, but MUST exist within the
3688 <KeyCachePolicy> element.

3689 4. <KeyClass> [Required]

3690

3691 This element of type **KeyClassType** identifies the key-class of the symmetric key to which this
3692 policy applies.

3693 5. <StartDate> [Required]

3694

3695 The <StartDate> element, of type XSD **dateTime**, specifies the date and time at which this
3696 policy becomes effective. There SHALL be only one <StartDate> element within a
3697 <KeyCachePolicy> element.

3698 6. <EndDate> [Required]

3699

3700 The <EndDate> element, of type XSD **dateTime**, specifies the date and time at which this policy
3701 expires. There SHALL be only one <EndDate> element within a <KeyCachePolicy> element.

3702

3703 The <EndDate> MAY be an empty element, but MUST exist within the
3704 <KeyCachePolicy> element.

3705 7. <PolicyCheckInterval> [Required]

3706

3707 The <PolicyCheckInterval> element , of type XSD *nonNegativeInteger*, specifies
3708 the frequency at which the client SHALL check the **SKS** server for updates to this
3709 policy. This frequency is specified in seconds and SHALL NOT exceed 2592000
3710 seconds (30 calendar days). There SHALL be only one <PolicyCheckInterval>
3711 element within a <KeyCachePolicy> element.

3712 8. <Status> [Required]

3713
3714
3715
3716
3717

The <Status> element, of type *StatusType*, identifies the current status of this policy within the SKMS. There SHALL be only one <Status> element within a <KeyCachePolicy> element.

The *StatusType* is specified in Section 4.11.

3718 9. <NewKeysCacheDetail> [Required]

3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731

The <NewKeysCacheDetail> element, of type *KeyCacheDetailType*, defines how many new (as yet unused for any encryption transaction) symmetric keys a client may cache, and for how long. It is the responsibility of the conforming **SKCL** implementation to enforce these rules.

The absence of the <NewKeysCacheDetail> element implies that new symmetric keys SHALL NEVER be cached on the client. New keys may be cached only when this element exists, and SHALL conform to the rules specified in this element.

When it exists, there SHALL be only one <NewKeysCacheDetail> element in a <KeyCachePolicy> element.

The *KeyCacheDetailType* is specified in Section 4.22.

3732 10. <UsedKeysCacheDetail> [Required]

3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745

The <UsedKeysCacheDetail> element, of type *KeyCacheDetailType*, defines how many used symmetric keys a client may cache, and for how long. It is the responsibility of the conforming **SKCL** implementation to enforce these rules.

The absence of the <UsedKeysCacheDetail> element implies that used symmetric keys SHALL NEVER be cached on the client. Used keys may be cached only when this element exists, and SHALL conform to the rules specified in this element.

When it exists, there SHALL be only one <UsedKeysCacheDetail> element in a <KeyCachePolicy> element.

The *KeyCacheDetailType* is specified in Section 4.22.

3746 Some examples of the <KeyUsePolicy> element are as follows.

3747 **Example 1 – A <KeyCachePolicy> that is valid between January 01, 2008 and December 31, 2008. It**
3748 **requires the client to check for policy updates every day and allows 3 new and 3 used keys to be**
3749 **cached for up to 90 days:**

```
3750 <ekmi:KeyCachePolicy>
3751   <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
3752   <ekmi:PolicyName>
3753     Corporate Laptop Symmetric Key Caching Policy
3754   </ekmi:PolicyName>
3755   <ekmi:Description>
3756     This policy defines how company-issued laptops will manage
3757     symmetric keys used for file/disk encryption in each laptop's
3758     local cache. This policy must be used by all laptops that
3759     use the company EKMI.
```

```

3760     </ekmi:Description>
3761     <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
3762     <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
3763     <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
3764     <ekmi:Status>Active</ekmi:Status>
3765     <ekmi:NewKeysCacheDetail>
3766         <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
3767         <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
3768     </ekmi:NewKeysCacheDetail>
3769     <ekmi:UsedKeysCacheDetail>
3770         <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
3771         <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
3772     </ekmi:UsedKeysCacheDetail>
3773 </ekmi:KeyCachePolicy>

```

3774 **Example 2 – A <KeyCachePolicy> that is effective starting January 01, 2008 and never expires. It**
3775 **does NOT permit any caching of symmetric keys through the absence of the detail elements on**
3776 **caching:**

```

3777     <ekmi:KeyCachePolicy>
3778         <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
3779         <ekmi:PolicyName>
3780             No Caching Policy
3781         </ekmi:PolicyName>
3782         <ekmi:Description>
3783             This policy is for high-risk, always-connected machines on the
3784             network, which will never cache symmetric keys locally. This
3785             policy never expires (but checks monthly for any updates).
3786         </ekmi:Description>
3787         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
3788         <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
3789         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
3790         <ekmi:Status>Active</ekmi:Status>
3791     </ekmi:KeyCachePolicy>

```

3792

3793 4.25 Type *KeyCacheDetailType*

3794 The *KeyCacheDetailType* type allows **SKS** servers to specify precisely how many symmetric keys
3795 MAY be cached on the client machine, and for how long.

3796 Schema Definition:

```

3797     <xsd:complexType name="KeyCacheDetailType">
3798         <xsd:sequence>
3799             <xsd:element name="MaximumKeys" minOccurs="1">
3800                 <xsd:simpleType>
3801                     <xsd:restriction base="xsd:integer">
3802                         <xsd:minInclusive value="0"/>
3803                         <xsd:maxInclusive value="18446744073709551615"/>
3804                     </xsd:restriction>
3805                 </xsd:simpleType>
3806             </xsd:element>
3807             <xsd:element name="MaximumDuration" minOccurs="1">
3808                 <xsd:simpleType>
3809                     <xsd:restriction base="xsd:integer">
3810                         <xsd:minInclusive value="0"/>
3811                         <xsd:maxInclusive value="18446744073709551615"/>
3812                     </xsd:restriction>

```

```

3813         </xsd:simpleType>
3814     </xsd:element>
3815 </xsd:sequence>
3816 </xsd:complexType>

```

3817 The **KeyCacheDetailType** consists of the following child elements:

3818 1. <MaximumKeys> [Required]

3819

3820 The <MaximumKeys> element, of type XSD **Integer**, specifies the maximum number of symmetric
3821 keys that MAY be cached on a client machine. It SHALL be a positive number between the
3822 values 0 and 18446744073709551615. There SHALL be only one <MaximumKeys> element
3823 within an element that uses the **KeyCacheDetailType**.

3824 2. <MaximumDuration> [Required]

3825

3826 The <MaximumDuration> element, of type XSD **Integer**, specifies the maximum number of
3827 seconds that symmetric keys MAY be cached on a client machine. It SHALL be a positive
3828 number between the values 0 and 18446744073709551615. There SHALL be only one
3829 <MaximumDuration> element within an element that uses the **KeyCacheDetailType**.

3830 Examples of the **KeyCacheDetailType** when used in the <KeyCachePolicy> element are as follows.

3831 **Example 1 – A <KeyCachePolicy> that is valid between January 01, 2008 and December 31, 2008. It**
3832 **requires the client to check for policy updates every day and allows 3 new and 3 used keys to be**
3833 **cached for up to 90 days:**

```

3834 <ekmi:KeyCachePolicy>
3835   <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
3836   <ekmi:PolicyName>
3837     Corporate Laptop Symmetric Key Caching Policy
3838   </ekmi:PolicyName>
3839   <ekmi:Description>
3840     This policy defines how company-issued laptops will manage
3841     symmetric keys used for file/disk encryption in their local
3842     cache. This policy must be used by all laptops that use
3843     the company EKMI.
3844   </ekmi:Description>
3845   <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
3846   <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
3847   <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
3848   <ekmi:Status>Active</ekmi:Status>
3849   <ekmi:NewKeysCacheDetail>
3850     <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
3851     <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
3852   </ekmi:NewKeysCacheDetail>
3853   <ekmi:UsedKeysCacheDetail>
3854     <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
3855     <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
3856   </ekmi:UsedKeysCacheDetail>
3857 </ekmi:KeyCachePolicy>

```

3858 **Example 1 – A <KeyCachePolicy> that is valid between January 01, 2008 and December 31, 2008. It**
3859 **requires the client to check for policy updates every day and allows 1 new and 0 used keys to be**
3860 **cached for upto 15 days:**

```

3861 <ekmi:KeyCachePolicy>
3862   <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
3863   <ekmi:PolicyName>
3864     Corporate Laptop Symmetric Key Caching Policy
3865   </ekmi:PolicyName>

```

3866 <ekmi:Description>
3867 *This policy defines how company-issued laptops will manage*
3868 *symmetric keys used for file/disk encryption in each laptop's*
3869 *local cache. This policy must be used by all laptops that*
3870 *use the company EKML.*
3871 </ekmi:Description>
3872 <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
3873 <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
3874 <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
3875 <ekmi:Status>Active</ekmi:Status>
3876 <ekmi:NewKeysCacheDetail>
3877 <ekmi:MaximumKeys>1</ekmi:MaximumKeys>
3878 <ekmi:MaximumDuration>1296000</ekmi:MaximumDuration>
3879 </ekmi:NewKeysCacheDetail>
3880 <ekmi:UsedKeysCacheDetail>
3881 <ekmi:MaximumKeys>0</ekmi:MaximumKeys>
3882 <ekmi:MaximumDuration>1296000</ekmi:MaximumDuration>
3883 </ekmi:UsedKeysCacheDetail>
3884 </ekmi:KeyCachePolicy>

3885

3886

3887

3888 **5 Bindings**

3889 An SKSML implementation can provide symmetric key services over a wide variety of transport
3890 mechanisms. This is referred to as “Bindings” of this specification. To maintain compliance with the
3891 specification, implementations must support the bindings that are REQUIRED.

3892 **5.1 W3C Security Binding**

3893 An SKSML implementation is REQUIRED to provide a binding that transports SKSML messages with
3894 support for XML Signatures and XML Encryption.

3895 **5.2 Mutually Authenticated TLS Binding**

3896 An SKSML implementation is REQUIRED to provide a binding that transports SKSML messages over a
3897 TLS connection that is mutually authenticated.

3898 **5.3 SOAP-WSS Binding**

3899 An SKSML implementation MAY provide SOAP and Web Services Security [WSS] support for transporting
3900 SKSML messages.

3901

6 Conformance

3902 An implementation conforms to this specification if it satisfies all of the MUST or REQUIRED level requirements
3903 defined within this specification. An SKSML Node MUST NOT use the XML namespace identifier for this
3904 specification (listed in the Title section under Declared Namespace(s)) within SOAP Envelopes unless it is compliant
3905 with this specification.

3906 This specification references a number of other specifications (see the table above). In order to comply with this
3907 specification, an implementation MUST implement the portions of referenced specifications necessary to comply
3908 with the required provisions of this specification. Additionally, the implementation of the portions of the referenced
3909 specifications that are specifically cited in this specification MUST comply with the rules for those portions as
3910 established in the referenced specification.

3911 Additionally normative text within this specification takes precedence over normative outlines, which in turn take
3912 precedence over the XML Schema [XML Schema Part 1, Part 2] descriptions. That is, the normative text in this
3913 specification further constrains the schema part of this specification; and this specification contains further
3914 constraints on the elements defined in referenced schemas.

3915 If an OPTIONAL message is not supported, then the implementation SHOULD Fault just as it would for
3916 any other unrecognized/unsupported message. If an OPTIONAL message is supported, then the
3917 implementation MUST satisfy all of the MUST and REQUIRED sections of the message.

3918

Appendix A. Acknowledgments

3919 The following individuals have participated in the creation of this specification and are gratefully
3920 acknowledged

3921 **Participants:**

- 3922 • Arshad Noor, StrongAuth (Former Chair of EKMI TC)
- 3923 • Anil Saldhana, Red Hat
- 3924 • Tomas Gustavsson, PrimeKey
- 3925 • Tim Bruce, CA
- 3926 • Eric Lengvenis, Wells Fargo
- 3927 • Upendra Mardikar, Paypal Inc.
- 3928 • Ken Adler, Individual
- 3929 • Davi Ottenheimer, Individual
- 3930 • Benjamin Tomhave, Individual
- 3931 • Shahid Sharif, Individual
- 3932 • Marc Massar, Individual
- 3933 • Shaheen N Abdul Jabbar - Individual

3934

3935

3936

3937

3938

Appendix B. Non-Normative Text

3939

3940

Appendix C. SKSML Error Codes and Error Messages

3941

3942

Appendix D. Revision History

3943

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.
PR2	10/19/09	Anil Saldhana	Updated the document to the August 2008 template. Added a Profile section

3944