



[Symmetric Key Services Markup Language (SKSML) Version 1.0 Non-Normative DRAFT 6]

OASIS Technical Committee DRAFT

24 June 2008

Specification URIs:

This Version:

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

<http://docs.oasis-open.org/ekmi/1.0/SKSML-1.0-Specification.odt>

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

Previous Version:

None

Latest Version:

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

<http://docs.oasis-open.org/ekmi/1.0/SKSML-1.0-Specification.odt>

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

Latest Approved Version:

None

Technical Committee:

OASIS Enterprise Key Management Infrastructure (EKMI) TC

Chair(s):

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

Editor(s):

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

Related Work:

This specification replaces or supercedes:

- [specifications replaced by this standard - OASIS as well as other standards organizations]

This specification is related to:

- [specifications related to this standard - OASIS as well as other standards organizations]

Declared XML Namespace(s):

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

33 **Abstract:**

34 This specification defines the first (1.0) version of the Symmetric Key Services Markup Language
35 (SKSML).

36 **Status:**

37 This document was last revised or approved by the EKMI TC on the above date. The level of approval
38 is also listed above. Check the "Latest Version" or "Latest Approved Version" location noted above for
39 possible later revisions of this document.

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

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

47 The non-normative errata page for this specification is located at <http://www.oasis-open.org/committees/>
48 [\[TBD\]](#).

Notices

49

50 Copyright © OASIS® 2008. All Rights Reserved.

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

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

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

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

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

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

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

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

91

92 Table of Contents

93	1 Introduction.....	5
94	1.1 Terminology.....	5
95	1.2 Glossary.....	5
96	1.3 Normative References.....	6
97	1.4 Non-normative References.....	6
98	2 Background (non-normative).....	7
99	2.1 Requirements (non-normative).....	8
100	3 Examples of use of SKSML (non-normative).....	10
101	3.1 Request for a new symmetric key.....	10
102	3.2 Response with a new symmetric key.....	12
103	3.3 Request for an existing symmetric key.....	16
104	3.4 Response with an existing symmetric key.....	17
105	3.5 Request for a new symmetric key of a specific KeyClass.....	17
106	3.6 Response with a new symmetric key of a specific KeyClass.....	17
107	3.7 Request for multiple new symmetric keys.....	18
108	3.8 Response with multiple new symmetric keys.....	19
109	3.9 Response with an SKS error.....	23
110	3.10 Response with symmetric keys and errors.....	24
111	3.11 Request for a symmetric key-caching policy.....	26
112	3.12 Response with a symmetric key-caching policy (1).....	28
113	3.13 Response with a symmetric key-caching policy (2).....	29
114	3.14 Response with multiple symmetric key-caching policies (3).....	31
115		

116 1 Introduction

117 This document presents the specification for the Symmetric Key Services Markup Language (SKSML), a protocol
118 by which applications may request and receive symmetric key-management services, securely, over the network.
119 All text is **non-normative** unless otherwise indicated.

120 1.1 Terminology

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

124 1.2 Glossary

125 **3DES** – Triple Data Encryption Standard

126 **AES** – Advanced Encryption Standard

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

128 **Ciphertext** – Encrypted data

129 **Cryptographic module** – A software library or hardware module dedicated to performing cryptographic
130 operations

131 **DES** – Data Encryption Standard

132 **DID or Domain ID** – Domain Identifier; the unique **PEN** assigned to an implementation of an **SKMS** within an
133 enterprise

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

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

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

140 **KID or Key ID** – Key Identifier; the unique integer assigned to every symmetric encryption key generated within a
141 specific **SKS** server within an **SKMS**

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

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

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

148 **Plaintext** – Unencrypted data

149 **SHA** – Secure Hashing Algorithm

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

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

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

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

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

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

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

158 **SKS** – Symmetric Key Services; a server that provides symmetric key management services over the network

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

161 **SOAP** – Simple Object Access Protocol

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

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

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

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

168 **Symkey** - A symmetric encryption key

169 **XML Encryption** – Encrypted content represented in eXtensible Markup Language and conformant to the World
170 Wide Web Consortium's XML Encryption standard

171 **XML Signature** – A digital signature represented in eXtensible Markup Language and conformant to the World
172 Wide Web Consortium's XML Signature standard

173 **1.3 Normative References**

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

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

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

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

182 **[XML Signature]** XML Signature Syntax and Processing
183 W3C Recommendation 12 Feb 2002. <http://www.w3.org/TR/xmldsig-core/>

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

188 **[RFC 2578]** K. McCloghrie, et al. *Structure of Management Information Version 2 (SMIv2)*.
189 IETF RFC 2578, April 1999. <http://www.ietf.org/html/rfc2578>

190 **1.4 Non-normative References**

191 **[IDtrust-EKMI]** A. Noor, *Securing the core with an EKMI*. NIST-IDtrust 2008, March 2008.
192 <http://middleware.internet2.edu/idtrust/2008/papers/07-noor-ekmi.pdf>
193

2 Background (non-normative)

194

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

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

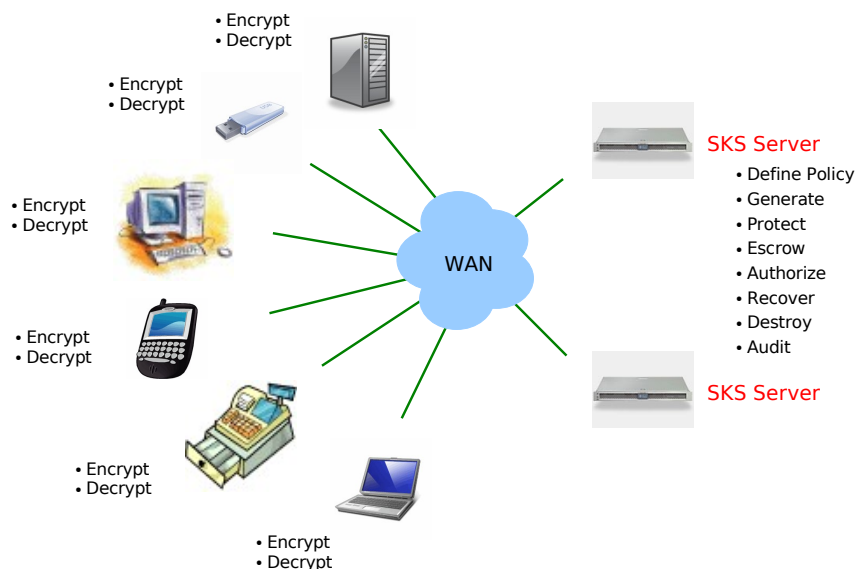
209 In a rush to provide solutions to the market, vendors have created many device-specific, platform-specific,
210 database-specific and application-specific encryption and key-management tools. While these tools may be
211 capable of performing their stated tasks adequately, a typical enterprise would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



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

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

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



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

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

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

233 2.1 Requirements (non-normative)

234 The requirements of the SKSML protocol are that:

- 235 • It must be platform independent;
- 236 • It must support the request of new and previously escrowed symmetric encryption keys;
- 237 • It must support the unique identification of every symmetric encryption key on the internet;
- 238 • It must provide message authenticity, confidentiality and integrity even when used over insecure
 239 networks;
- 240 • It must support the use of encryption/decryption services by a client even when disconnected from the
 241 network;
- 242 • It must provide flexibility in defining key-usage policies;

243 SKSML meets the above requirements in the following manner:

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

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

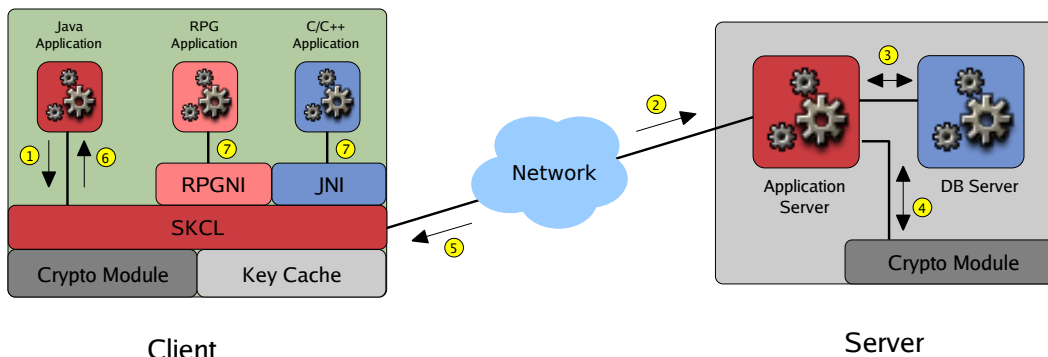
- 254 • Using secure key-caching enabled through centrally-defined policies, SKSML supports the request and
255 receipt of **KeyCachePolicy** elements by clients for the use of symmetric encryption keys even when the
256 client is disconnected from the network and an SKS server;
- 257 • SKSML provides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

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

3 Examples of use of SKSML (non-normative)

264

265 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

268

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

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

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

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

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

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

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

291 For an interpretation of the XML elements shown below, please refer to [WSS].
292 For the sake of brevity, this specification will dispense with showing the SOAP envelope and the WSS elements
293 in all other examples, when discussing SKSML.

```
294 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
295   <SOAP-ENV:Header>
296     <wsse:Security xmlns:wsse="http://docs.oasis-
297 open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" SOAP-
298 ENV:mustUnderstand="1">
299       <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-
300 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
301         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
302 message-security-1.0#Base64Binary"
303         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
304 profile-1.0#X509v3" wsu:Id="XWSSGID-1172790302111-1738806553">
305         MIIDfDCCAmSgAwIBAgIIAe/AvliGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdu3Ryb25nab
306         S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvcjBtdHJvbmdLZXkgREVNTyBVc2Ug1d
307         T25seTEXMBUGA1UEChM0U3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTcxMDMwWhcNMDcwNzIa64dd3k
308         A1UECXMbRm9yIFN0cm9uZ0tleSBERU1PIFVzZSBPbmx5MRcwFQYDVQKkEw5TdHJvbmdBdXR0eIEl2da
309         S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvcjBtdHJvbmdLZXkgREVNTyBVc2Ug1d
310         T25seTEXMBUGA1UEChM0U3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTY0NjEwWhcNMDcwNzI1s34wdd
311         NjEwWjBpMREwDwYKZCIzPyLQBARMBOTEVMBMGA1UEAxMMU0tTIFNlcnZlci0xMSQwIgyYDVQsdw2
312         ExtGb3Igu3Ryb25nS2V5IERFTU8gVXNlIE9ubHkxFzAVBgNVBAoTDlN0cm9uZ0F1dGggS5w5jMIIBd2
313         NBGkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAztpqRoU5A8plxx1Rz1QEUnlAAM1D5g9+isIr3wxa
314         hbwjfFSMYilnY4iV77xU/nsM0nMZ7RxsLYKdCzQ10DVYqQwqmAvaJ5Z6SVy34gZ51YG+rSWE3NjFsd
315         b0XW8RJYA/Tn6Lmht/qngrcaqmtP0cAAiMRZOWtCTmC2K/LEqDabXSyU6Hh8ySNE3njybvMwPresf
316         zsYokTdvWQqT6tKo10wJsdJ1+hxM7DrnMLvMNq5reINfsKhDdX17wzhrBUx+hiYA/qo8tMXkL6wsd
317         4PN5dYugtZpSzIdU05tIg58Avhzw07hy5oofBlKFY22CeljQ36u0bMjuyGj6UYHs3rdfdfsds32rda
318         YzCBnzANBkgqhkiG9w0BAQEFAAOBjQAwYkCgYEAyAmxMZhYA8wHJ4UE4b61s51JVWe4Fyggj4Mcf3a
319         hvcNAQELBQADggEBACK05PtvZD4WPgl0e=
320       </wsse:BinarySecurityToken>
321       <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
322         <ds:SignedInfo>
323           <ds:CanonicalizationMethod
324             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
325             <InclusiveNamespaces
326               xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"
327               PrefixList="wsse SOAP-ENV"/>
328           </ds:CanonicalizationMethod>
329           <ds:SignatureMethod
330             Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
331           <ds:Reference URI="#XWSSGID-1172790300636-653454040">
332             <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
333             <ds:DigestValue>LU4m+rp4oebgl9g+t3nRaZYqULe=</ds:DigestValue>
334           </ds:Reference>
335           <ds:Reference URI="#XWSSGID-1172790300637708871805">
336             <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
337             <ds:DigestValue>WCpOmTCbffcEHXhGf5rLEWLRZg=</ds:DigestValue>
338           </ds:Reference>
339         </ds:SignedInfo>
340         <ds:SignatureValue>
341         svStAvBRRrF+g2biPL7uWHkJTQPIl8t4phMb0ZQsZlQcn36tcMSj/a4+4LPNf0B3Y8y02lr10a1
342         fGqCPAWZnuEH34VQEM196rRwV258mgp8uwpXEYJIgPjqg89w8+/Nda0DccLQ2Bizu7QM/HSM2ab
343         ogNjwqmbSyIaz0sn0cU=
344         </ds:SignatureValue>
345         <ds:KeyInfo>
346           <wsse:SecurityTokenReference xmlns:wssu="http://docs.oasis-
347 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
348             wsu:Id="XWSSGID-1172790300633-442423344">
```

```

349         <wsse:Reference URI="#XWSSGID-1172790302111-1738806553"
350             ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
351 wss-x509-token-profile-1.0#X509v3"/>
352         </wsse:SecurityTokenReference>
353     </ds:KeyInfo>
354 </ds:Signature>
355     <wsu:Timestamp xmlns:wsu="http://docs.oasis-
356 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
357 wsu:Id="XWSSGID-1172790300637708871805">
358         <wsu:Created>2007-03-01T23:05:00Z</wsu:Created>
359         <wsu:Expires>2007-03-01T23:05:05Z</wsu:Expires>
360     </wsu:Timestamp>
361 </wsse:Security>
362 </SOAP-ENV:Header>
363 <SOAP-ENV:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
364 wss-wssecurity-utility-1.0.xsd" wsu:Id="XWSSGID-1172790300636-653454040">
365     <ekmi:SymkeyRequest
366         xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
367         <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
368     </ekmi:SymkeyRequest>
369 </SOAP-ENV:Body>
370 </SOAP-ENV:Envelope>

```

371 3.2 Response with a new symmetric key

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

```

377 [b01] <ekmi:SymkeyResponse
378 [b02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
379 [b03]     xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
380 [b04]     <ekmi:Symkey>
381 [b05]         <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
382 [b06]         <ekmi:KeyUsePolicy>
383 [b07]             <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>
384 [b08]             <ekmi:PolicyName>DES-EDE KeyUsePolicy</ekmi:PolicyName>
385 [b09]             <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
386 [b10]             <ekmi:KeyAlgorithm>
387 [b11]                 http://www.w3.org/2001/04/xmlenc#tripledes-cbc
388 [b12]             </ekmi:KeyAlgorithm>
389 [b13]             <ekmi:KeySize>192</ekmi:KeySize>
390 [b14]             <ekmi:Status>Active</ekmi:Status>
391 [b15]         <ekmi:Permissions>
392 [b16]             <ekmi:PermittedApplications ekmi:any="false">
393 [b17]                 <ekmi:PermittedApplication>
394 [b18]                     <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
395 [b19]                     <ekmi:ApplicationName>
396 [b20]                         Payroll Application
397 [b21]                     </ekmi:ApplicationName>
398 [b22]                     <ekmi:ApplicationVersion>1.0</ekmi:ApplicationVersion>
399 [b23]                     <ekmi:ApplicationDigestAlgorithm>
400 [b24]                         http://www.w3.org/2000/09/xmldsig#sha1
401 [b25]                     </ekmi:ApplicationDigestAlgorithm>
402 [b26]                     <ekmi:ApplicationDigestValue>
403 [b27]                         NIG4bKkt4cziEqFFu0oBTM81efU=
404 [b28]                     </ekmi:ApplicationDigestValue>

```

```

405 [b29]         </ekmi:PermittedApplication>
406 [b30]     </ekmi:PermittedApplications>
407 [b31]     <ekmi:PermittedDates ekmi:any="false">
408 [b32]         <ekmi:PermittedDate>
409 [b33]             <ekmi:StartDate>2008-01-01</ekmi:StartDate>
410 [b34]             <ekmi:EndDate>2008-12-31</ekmi:EndDate>
411 [b35]         </ekmi:PermittedDate>
412 [b36]     </ekmi:PermittedDates>
413 [b37]     <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
414 [b38]     <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
415 [b39]     <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
416 [b40]     <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
417 [b41]     <ekmi:PermittedNumberOfTransactions ekmi:any="true"
418 xsi:nil="true"/>
419 [b42]         <ekmi:PermittedTimes ekmi:any="false">
420 [b43]             <ekmi:PermittedTime>
421 [b44]                 <ekmi:StartTime>07:00:00</ekmi:StartTime>
422 [b45]                 <ekmi:EndTime>19:00:00</ekmi:EndTime>
423 [b46]             </ekmi:PermittedTime>
424 [b47]         </ekmi:PermittedTimes>
425 [b48]         <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
426 [b49]     </ekmi:Permissions>
427 [b50] </ekmi:KeyUsePolicy>
428 [b51]     <ekmi:EncryptionMethod
429 [b52]         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
430 [b53]     <xenc:CipherData>
431 [b54]         <xenc:CipherValue>
432 [b55]             E9zWB/y93hVSzeTLiDcQoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJM
433 [b56]             hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/2fw
434 [b57]             UJow/qimxi8+huUYJMtaGHtXuLLWtx27STRcRpIsY=
435 [b58]         </xenc:CipherValue>
436 [b59]     </xenc:CipherData>
437 [b60] </ekmi:Symkey>
438 [b61] </ekmi:SymkeyResponse>

```

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

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

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

443 [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.

447 [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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

572 3.3 Request for an existing symmetric key

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

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

```
584 [c01] <ekmi:SymkeyRequest
585 [c02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
586 [c03]     <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
587 [c04] </ekmi:SymkeyRequest>
```

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

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

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

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

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

597 3.4 Response with an existing symmetric key

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

603 3.5 Request for a new symmetric key of a specific **KeyClass**

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

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

```
612 [e01] <ekmi:SymkeyRequest  
613 [e02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
614 [e03]     <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
615 [e04]     <ekmi:KeyClasses>  
616 [e05]         <ekmi:KeyClass>HR-Class</ekmi:KeyClass>  
617 [e06]     </ekmi:KeyClasses>  
618 [e07] </ekmi:SymkeyRequest>
```

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

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

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

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

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

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

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

628 3.6 Response with a new symmetric key of a specific **KeyClass**

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

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

635 3.7 Request for multiple new symmetric keys

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

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

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

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

```
654 [g01] <ekmi:SymkeyRequest  
655 [g02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
656 [g03]     <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
657 [g04]     <ekmi:KeyClasses>  
658 [g05]         <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>  
659 [g06]         <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>  
660 [g07]         <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>  
661 [g08]         <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>  
662 [g09]         <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>  
663 [g10]        <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>  
664 [g11]        <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>  
665 [g12]        <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>  
666 [g13]        <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>  
667 [g14]     </ekmi:KeyClasses>  
668 [g15] </ekmi:SymkeyRequest>
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

700 3.8 Response with multiple new symmetric keys

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

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

```

710 [h01] <ekmi:SymkeyResponse
711 [h02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
712 [h03]     xmlns:xenc='http://www.w3.org/2001/04/xmenc#'>
713 [h04]     <ekmi:Symkey>
714 [h05]         <ekmi:GlobalKeyID>10514-4-3792</ekmi:GlobalKeyID>
715 [h06]         <ekmi:KeyUsePolicy>
716 [h07]             <ekmi:KeyUsePolicyID>10514-9</ekmi:KeyUsePolicyID>
717 [h08]             <ekmi:PolicyName>DES-EDE KeyUsePolicy for EHR-CDC</ekmi:PolicyName>
718 [h09]             <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
719 [h10]             <ekmi:KeyAlgorithm>
720 [h11]                 http://www.w3.org/2001/04/xmenc#tripleDES-cbc
721 [h12]             </ekmi:KeyAlgorithm>
722 [h13]             <ekmi:KeySize>192</ekmi:KeySize>
723 [h14]             <ekmi:Status>Active</ekmi:Status>
724 [h15]             <ekmi:Permissions>
725 [h16]                 <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
726 [h17]                 <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
727 [h18]                 <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
728 [h19]                 <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
729 [h20]                 <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>

```

```

730 [h21]         <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
731 [h22]         <ekmi:PermittedNumberOfTransactions
732 [h23]             ekmi:any="true" xsi:nil="true"/>
733 [h24]         <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
734 [h25]         <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
735 [h26]     </ekmi:Permissions>
736 [h27] </ekmi:KeyUsePolicy>
737 [h28] <ekmi:EncryptionMethod
738 [h29]     Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
739 [h30] <xenc:CipherData>
740 [h31]     <xenc:CipherValue>
741 [h32]         E9zWB/y93hVSzeTLiDcQoDxmlNXTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJMa1w
742 [h33]         hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfg1pU8tLWtx27STRcR/2fw2ava
743 [h34]         ULWtx27STRcRJMtaGHtXuLLWtx27STRcRpIsY=
744 [h35]     </xenc:CipherValue>
745 [h36] </xenc:CipherData>
746 [h37] </ekmi:Symkey>
747 [h38] <ekmi:Symkey>
748 [h39]     <ekmi:GlobalKeyID>10514-4-3793</ekmi:GlobalKeyID>
749 [h40]     <ekmi:KeyUsePolicy>
750 [h41]         <ekmi:KeyUsePolicyID>10514-12</ekmi:KeyUsePolicyID>
751 [h42]         <ekmi:PolicyName>DES-EDE KeyUsePolicy for EHR-CR0</ekmi:PolicyName>
752 [h43]         <ekmi:KeyClass>EHR-CR0</ekmi:KeyClass>
753 [h44]         <ekmi:KeyAlgorithm>
754 [h45]             http://www.w3.org/2001/04/xmlenc#tripledes-cbc
755 [h46]         </ekmi:KeyAlgorithm>
756 [h47]         <ekmi:KeySize>192</ekmi:KeySize>
757 [h48]         <ekmi:Status>Active</ekmi:Status>
758 [h49]         <ekmi:Permissions>
759 [h50]             <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
760 [h51]             <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
761 [h52]                 <ekmi:PermittedDate>
762 [h53]                     <ekmi:StartDate>2008-01-01</ekmi:StartDate>
763 [h54]                     <ekmi:EndDate>2009-12-31</ekmi:EndDate>
764 [h55]                 </ekmi:PermittedDate>
765 [h56]             </ekmi:PermittedDates>
766 [h57]             <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
767 [h58]             <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
768 [h59]             <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
769 [h60]             <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
770 [h61]             <ekmi:PermittedNumberOfTransactions
771 [h62]                 ekmi:any="true" xsi:nil="true"/>
772 [h63]             <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
773 [h64]             <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
774 [h65]         </ekmi:Permissions>
775 [h66]     </ekmi:KeyUsePolicy>
776 [h67]     <ekmi:EncryptionMethod
777 [h68]         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
778 [h69]     <xenc:CipherData>
779 [h70]         <xenc:CipherValue>
780 [h71]             qUiQXG0ca8EU871zBoXBjDoDxmlNXTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJM1
781 [h72]             hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/2fw1
782 [h73]             UJow/qimxi8+ huUYJMtaGHtXuLLWtx27STRcRpIsY=
783 [h74]         </xenc:CipherValue>
784 [h75]     </xenc:CipherData>
785 [h76] </ekmi:Symkey>
786 [h77] <ekmi:Symkey>
787 [h78]     <ekmi:GlobalKeyID>10514-4-3795</ekmi:GlobalKeyID>
788     ...
789 [h79]     <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>

```

```

790      ...
791 [h80]      </ekmi:Symkey>
792 [h81]      <ekmi:Symkey>
793 [h82]      <ekmi:GlobalKeyID>10514-4-3797</ekmi:GlobalKeyID>
794      ...
795 [h83]      <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
796      ...
797 [h84]      </ekmi:Symkey>
798 [h85]      <ekmi:Symkey>
799 [h86]      <ekmi:GlobalKeyID>10514-4-3798</ekmi:GlobalKeyID>
800      ...
801 [h87]      <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
802      ...
803 [h88]      </ekmi:Symkey>
804 [h89]      <ekmi:Symkey>
805 [h90]      <ekmi:GlobalKeyID>10514-4-3799</ekmi:GlobalKeyID>
806      ...
807 [h91]      <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
808      ...
809 [h92]      </ekmi:Symkey>
810 [h93]      <ekmi:Symkey>
811 [h94]      <ekmi:GlobalKeyID>10514-4-3801</ekmi:GlobalKeyID>
812      ...
813 [h95]      <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
814      ...
815 [h96]      </ekmi:Symkey>
816 [h97]      <ekmi:Symkey>
817 [h98]      <ekmi:GlobalKeyID>10514-4-3803</ekmi:GlobalKeyID>
818      ...
819 [h99]      <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
820      ...
821 [h100]     </ekmi:Symkey>
822 [h101]     <ekmi:Symkey>
823 [h102]     <ekmi:GlobalKeyID>10514-4-3805</ekmi:GlobalKeyID>
824      ...
825 [h103]     <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>
826      ...
827 [h104]     </ekmi:Symkey>
828 [h105]     </ekmi:SymkeyResponse>

```

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

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

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

834 [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.

838 [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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- 901 • [h39] identifies a different **GlobalKeyID** (10514-4-3793) for this symmetric key;
- 902 • [h41] identifies a different **KeyUsePolicy** (10514-12) for this symmetric key;
- 903 • [h43] identifies a different **KeyClass** (EHR-CRO) for this symmetric key;
- 904 • [h49] - [h65] defines a different **Permissions** element for this symmetric key;
- 905 • [h71] - [h73] contains a different **CipherValue** for this symmetric key;

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

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

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

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

917 3.9 Response with an SKS error

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

```
922 [i01] <ekmi:SymkeyResponse  
923 [i02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'  
924 [i03]     xmlns:xenc='http://www.w3.org/2001/04/xmenc#'>  
925 [i04]     <ekmi:SymkeyError>  
926 [i05]         <ekmi:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>  
927 [i06]         <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>  
928 [i07]         <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>  
929 [i08]     </ekmi:SymkeyError>  
930 [i09] </ekmi:SymkeyResponse>
```

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

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

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

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

938 [i06] is an *ErrorCode* returned by the SKS server. The code may be one of the standard error codes defined in
 939 this specification, or may be a vendor-specific error code.

940 [i07] is the text of the *ErrorMessage*, localized to the region and language of the requesting client application.

941 [i08] is the closing tag of the *SymkeyError* tag.

942 [i09] is the closing tag of the *SymkeyResponse* tag.

943 3.10 Response with symmetric keys and errors

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

- 946 i. Return all symmetric keys as requested;
- 947 ii. Return no symmetric keys – i.e. it returns all errors;
- 948 iii. Return some symmetric keys and some errors.

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

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

```

955 [j01] <ekmi:SymkeyResponse
956 [j02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
957 [j03]     xmlns:xenc='http://www.w3.org/2001/04/xmenc#'>
958 [j04]   <ekmi:Symkey>
959 [j05]     <ekmi:GlobalKeyID>10514-4-3792</ekmi:GlobalKeyID>
960 [j06]     <ekmi:KeyUsePolicy>
961 [j07]       <ekmi:KeyUsePolicyID>10514-9</ekmi:KeyUsePolicyID>
962 [j08]       <ekmi:PolicyName>DES-EDE Policy for EHR-CDC</ekmi:PolicyName>
963 [j09]       <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
964 [j10]       <ekmi:KeyAlgorithm>
965 [j11]         http://www.w3.org/2001/04/xmenc#tripleDES-cbc
966 [j12]       </ekmi:KeyAlgorithm>
967 [j13]     <ekmi:KeySize>192</ekmi:KeySize>
968 [j14]     <ekmi:Status>Active</ekmi:Status>
969 [j15]     <ekmi:Permissions>
970 [j16]       <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
971 [j17]       <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
972 [j18]       <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
973 [j19]       <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
974 [j20]       <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
975 [j21]       <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
976 [j22]       <ekmi:PermittedNumberOfTransactions
977 [j23]         ekmi:any="true" xsi:nil="true"/>
978 [j24]       <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
979 [j25]       <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
  
```



```

980 [j26]         </ekmi:Permissions>
981 [j27]     </ekmi:KeyUsePolicy>
982 [j28]     <ekmi:EncryptionMethod
983 [j29]         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
984 [j30]     <xenc:CipherData>
985 [j31]         <xenc:CipherValue>
986 [j32]             E9zWB/y93hVSzeTLiDcQoDxmLNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJ
987 [j33]             hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfglpU8tLWtx27STRcR/2fw
988 [j34]             ULWtx27STRcRJMtaGHtXuLlWtx27STRcRpIsY=
989 [j35]         </xenc:CipherValue>
990 [j36]     </xenc:CipherData>
991 [j37] </ekmi:Symkey>
992 [j38] <ekmi:Symkey>
993 [j39]     <ekmi:GlobalKeyID>10514-4-3793</ekmi:GlobalKeyID>
994 [j40]     <ekmi:KeyUsePolicy>
995 [j41]         <ekmi:KeyUsePolicyID>10514-12</ekmi:KeyUsePolicyID>
996 [j42]         <ekmi:PolicyName>DES-EDE Policy for EHR-CRO</ekmi:PolicyName>
997 [j43]         <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
998 [j44]         <ekmi:KeyAlgorithm>
999 [j45]             http://www.w3.org/2001/04/xmlenc#tripledes-cbc
1000 [j46]         </ekmi:KeyAlgorithm>
1001 [j47]     <ekmi:KeySize>192</ekmi:KeySize>
1002 [j48]     <ekmi:Status>Active</ekmi:Status>
1003 [j49]     <ekmi:Permissions>
1004 [j50]         <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
1005 [j51]         <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
1006 [j52]             <ekmi:PermittedDate>
1007 [j53]                 <ekmi:StartDate>2008-01-01</ekmi:StartDate>
1008 [j54]                 <ekmi:EndDate>2009-12-31</ekmi:EndDate>
1009 [j55]             </ekmi:PermittedDate>
1010 [j56]         </ekmi:PermittedDates>
1011 [j57]         <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
1012 [j58]         <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
1013 [j59]         <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
1014 [j60]         <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
1015 [j61]         <ekmi:PermittedNumberOfTransactions
1016 [j62]             ekmi:any="true" xsi:nil="true"/>
1017 [j63]         <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
1018 [j64]         <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
1019 [j65]     </ekmi:Permissions>
1020 [j66] </ekmi:KeyUsePolicy>
1021 [j67] <ekmi:EncryptionMethod
1022 [j68]     Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
1023 [j69] <xenc:CipherData>
1024 [j70]     <xenc:CipherValue>
1025 [j71]         qUiQXG0ca8EU871zBoXBjDoDxmLNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCF
1026 [j72]         hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfglpU8tGFbpWZcd/ATpJD/
1027 [j73]         UJow/qimxi8+ huUYJMtaGHtXuLlWtx27STRcRpIsY=
1028 [j74]     </xenc:CipherValue>
1029 [j75] </xenc:CipherData>
1030 [j76] </ekmi:Symkey>
1031 [j77] <ekmi:Symkey>
1032 [j78]     <ekmi:GlobalKeyID>10514-4-3795</ekmi:GlobalKeyID>
1033     ...
1034 [j79]     <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
1035     ...
1036 [j80] </ekmi:Symkey>
1037 [j81] <ekmi:Symkey>
1038 [j82]     <ekmi:GlobalKeyID>10514-4-3797</ekmi:GlobalKeyID>
1039     ...

```

```

1040 [j83]          <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
1041          ...
1042 [j84]        </ekmi:Symkey>
1043 [j85]        <ekmi:Symkey>
1044 [j86]          <ekmi:GlobalKeyID>10514-4-3798</ekmi:GlobalKeyID>
1045          ...
1046 [j87]          <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
1047          ...
1048 [j88]        </ekmi:Symkey>
1049 [j89]        <ekmi:Symkey>
1050 [j90]          <ekmi:GlobalKeyID>10514-4-3799</ekmi:GlobalKeyID>
1051          ...
1052 [j91]          <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
1053          ...
1054 [j92]        </ekmi:Symkey>
1055 [j93]        <ekmi:Symkey>
1056 [j94]          <ekmi:GlobalKeyID>10514-4-3801</ekmi:GlobalKeyID>
1057          ...
1058 [j95]          <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
1059          ...
1060 [j96]        </ekmi:Symkey>
1061 [j97]        <ekmi:SymkeyError>
1062 [j98]          <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
1063 [j99]          <ekmi:RequestedKeyClass>EHR-PAT</ekmi:RequestedKeyClass>
1064 [j100]         <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
1065 [j101]         <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
1066 [j102]        </ekmi:SymkeyError>
1067 [j103]        <ekmi:SymkeyError>
1068 [j104]         <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
1069 [j105]         <ekmi:RequestedKeyClass>EHR-PHY</ekmi:RequestedKeyClass>
1070 [j106]         <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
1071 [j107]         <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
1072 [j108]        </ekmi:SymkeyError>
1073 [j109] </ekmi:SymkeyResponse>

```

1074 [j01] – [j96] is no different from the response shown in *Section 3.8 Response with multiple new symmetric*
1075 *keys*.

1076 [j97] - [j102] identifies the first *SymkeyError* returned by the SKS server. It is not unlike the error described in
1077 *Section 3.9 Response with an SKS error*. However, there is one difference in this element on line [j79]. This
1078 element includes the *RequestedKeyClass* of the request.

1079 [j103] - [j108] identifies the second *SymkeyError* returned by the SKS server. It is similar to the error described
1080 in lines [j77] through [j82] including the *RequestedKeyClass* of the request.

1081 [j109] is the closing tag of the *SymkeyResponse* tag.

1082 3.11 Request for a symmetric key-caching policy

1083 When a client application (that has been linked to the SKCL) needs to encrypt sensitive data, it will call an API
1084 method within the SKCL for a new symmetric key. After the SKCL has ensured that the application is authorized
1085 to make such a request (by verifying that the configured/passed-in credentials can access the cryptographic
1086 key-store module on the client containing the PrivateKey used for signing SKSML requests), the SKCL
1087 assembles the following SKSML request:

```

1088 [k01] <ekmi:KeyCachePolicyRequest
1089 [k02]   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01"/>

```

1090 [k01] is the start of the *KeyCachePolicyRequest* element.

1091 [k02] identifies the namespace to which this XML conforms, and the location of its XML Schema Definition
1092 (XSD).

1093 The **KeyCachePolicyRequest** is an “empty” request. It does not need to specify any requesting parameter or
1094 element, since the SKS server only needs to know who requested the policy. The server derives this information
1095 from the SOAP Header and consequently has everything it needs to know – the digital signature to establish the
1096 identity of the requester, as well as to ensure message integrity in the request.

1097 While the **KeyCachePolicyRequest** element is very simple, the Web Service Security (WSS) envelope – which
1098 provides security for all SKSML messages – expands the size of the message. The same request shown above,
1099 is displayed below in its entirety, with its WSS envelope. Please note that some content – such as Base64-
1100 encoded binary content - has been reformatted for aesthetics and clarity of the XML elements. The actual
1101 elements and data-types have been preserved from actual SKSML messages.

1102 For an interpretation of the XML elements shown below, please refer to [WSS].

1103 For the sake of brevity, this specification will dispense with showing the SOAP envelope and the WSS elements
1104 in all other examples, when discussing SKSML.

```
1105 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
1106   <SOAP-ENV:Header>
1107     <wsse:Security xmlns:wsse="http://docs.oasis-
1108 open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" SOAP-
1109 ENV:mustUnderstand="1">
1110       <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-
1111 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
1112 EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
1113 message-security-1.0#Base64Binary"
1114 Value="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
1115 profile-1.0#X509v3" wsu:Id="XWSSGID-1172790302111-1738806553">
1116     MIIIDfDCCAmSgAwIBAgIAe/AvliGc3AwDQYJKoZIhvcNAQELBQAwZzEmMQGA1UEAxMdU3Ryb25nbn
1117 S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvcjBTdHJvbmdLZXkgREVNTyBVc2Ug1d
1118 T25seTEXMBUGA1UEChMOU3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTcxMDMwHhcNMDcwNzIa64dd3k
1119 A1UECXMbRm9yIFN0cm9uZ0tleSBERU1PIFVzZSBPbmx5MRcwFQYDVQKEw5TdHJvbmdBdXR0IEl2da
1120 S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvcjBTdHJvbmdLZXkgREVNTyBVc2Ug1d
1121 T25seTEXMBUGA1UEChMOU3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTY0NjEwWWhcNMDcwNzI1s34wdd
1122 NjEwWjBpMREwDwYKZCIzImZPYLQGBARMB0TEVMBMGA1UEAxMMU0tTIFNlcnZlcj0xMSQwIyYDVQsdcw2
1123 ExtG3IgU3Ryb25nS2V5IERFTU8gVXNlIE9ubHkxZzAVBgNVBAoTDlN0cm9uZ0F1dGggSW5jMIIBd2
1124 NBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAztpqRoU5A8plxx1Rz1QEUnLAAM1D5g9+isIr3wxa
1125 hbwjtFSMYilnY4iV77xU/nsM0nMz7RxsLYKdCzQ10DVYqQwqmAvaJ5Z6SVy34gZ51YG+rSWE3NjFsd
1126 b0XW8RjYA/Tn6Lmht/qngrcaqmtP0cAAiMRZOWtCTmC2K/LEqDabXSyU6Hh8ySNE3njbvbmWpresf
1127 zsYokTdvWQqT6tKol0wJsdJl+hxM7DrnMLvMNq5reINfsKhDdX17wzhrBUx+hiYA/qo8tMXkL6wsd
1128 4PN5dYugtZpSzIdU05tIg58Avhzw07hy5ooFBlKFY22CeljQ36u0bmjuyGj6UYHs3rdfdfsds32rda
1129 YzCBnzANBgkqhkiG9w0BAQEFAAOBjQAwYkCgYEAyAmxMZhYA8wHJ4UE4b61s51JVWe4Fygj4Mcf3a
1130 hvcNAQELBQADggEBACK05PtvZD4WPgl0e=
1131   </wsse:BinarySecurityToken>
1132   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
1133     <ds:SignedInfo>
1134       <ds:CanonicalizationMethod
1135         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1136       <InclusiveNamespaces
1137         xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"
1138         PrefixList="wsse SOAP-ENV"/>
1139       </ds:CanonicalizationMethod>
1140     <ds:SignatureMethod
1141       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
1142     <ds:Reference URI="#XWSSGID-1172790300636-653454040">
1143       <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1144       <ds:DigestValue>lu4m+rp4oebgl9g+t3nRaZYqULe=</ds:DigestValue>
1145     </ds:Reference>
1146     <ds:Reference URI="#XWSSGID-1172790300637708871805">
```

```

1147         <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1148         <ds:DigestValue>WCP0mTCbffcEHXhGf5rLEYWLRzg=</ds:DigestValue>
1149     </ds:Reference>
1150 </ds:SignedInfo>
1151 <ds:SignatureValue>
1152 svStAvBRRrF+g2biPL7uWHkJTQPIl8t4phMb0ZQsZlQcn36tcMSj/a4+4LPnf0B3Y8y02lr10a1
1153 fGqCPAWZNuEH34VQEM196rRwV258mgp8uwpXEYJIgPJqg89w8+/Nda0DccLQ2Bizu7QM/HSM2ab
1154 ogNjwqmbSyIaz0sn0cU=
1155 </ds:SignatureValue>
1156 <ds:KeyInfo>
1157     <wsse:SecurityTokenReference xmlns:wsu="http://docs.oasis-
1158 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
1159     wsu:Id="XWSSGID-1172790300633-442423344">
1160     <wsse:Reference URI="#XWSSGID-1172790302111-1738806553"
1161     ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1162 wss-x509-token-profile-1.0#X509v3"/>
1163     </wsse:SecurityTokenReference>
1164 </ds:KeyInfo>
1165 </ds:Signature>
1166 <wsu:Timestamp xmlns:wsu="http://docs.oasis-
1167 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
1168 wsu:Id="XWSSGID-1172790300637708871805">
1169     <wsu:Created>2007-03-01T23:05:00Z</wsu:Created>
1170     <wsu:Expires>2007-03-01T23:05:05Z</wsu:Expires>
1171 </wsu:Timestamp>
1172 </wsse:Security>
1173 </SOAP-ENV:Header>
1174 <SOAP-ENV:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1175 wss-wssecurity-utility-1.0.xsd" wsu:Id="XWSSGID-1172790300636-653454040">
1176     <ekmi:KeyCachePolicyRequest xmlns:ekmi="http://docs.oasis-
1177 open.org/ekmi/2008/01"/>
1178 </SOAP-ENV:Body>
1179 </SOAP-ENV:Envelope>

```

1180 3.12 Response with a symmetric key-caching policy (1)

1181 After an SKS server has performed its operations of authenticating a *KeyCachePolicyRequest*, identifying the
1182 requester, determining policies that apply to the requester, it assembles the following response and returns it to
1183 the client. (The SOAP message, as indicated earlier, is secured using WSS, but only the actual SKSML content
1184 is displayed and discussed here).

```

1185 [m01] <ekmi:KeyCachePolicyResponse
1186 [m02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1187 [m03]     xmlns:xenc='http://www.w3.org/2001/04/xmldsig#>
1188 [m04]     <ekmi:KeyCachePolicy>
1189 [m05]         <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
1190 [m06]         <ekmi:PolicyName>No Caching Policy</ekmi:PolicyName>
1191 [m07]         <ekmi:Description>
1192 [m08]             This policy is for high-risk, always-connected machines on the
1193 [m09]             network, which will never cache symmetric keys locally. This
1194 [m10]             policy never expires (but checks monthly for any updates).
1195 [m11]         </ekmi:Description>
1196 [m12]         <ekmi:KeyClass>NoCachingClass</ekmi:KeyClass>
1197 [m13]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1198 [m14]         <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
1199 [m15]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1200 [m16]         <ekmi:Status>Active</ekmi:Status>
1201 [m17]     </ekmi:KeyCachePolicy>
1202 [m18] </ekmi:KeyCachePolicyResponse>

```

1203 [m01] is the start of the *KeyCachePolicyResponse* element.

1204 [m02] and [m03] identify the namespaces to which this XML conforms, and the location of their XML Schema Definitions (XSD).

1206 [m04] is the start of the first – and only - *KeyCachePolicy* element.

1207 [m05] identifies the *KeyCachePolicyID* (KCPID) assigned to this policy by the SKS server. In this example, the concatenated values of the Domain ID and Policy ID indicate that the key belongs to the organization represented by the PEN, 10514; and is the first key-caching policy within the SKMS.

1210 [m06] provides a descriptive name for this key-cache policy through the *PolicyName* element, which is helpful to human readers when identifying this policy.

1212 [m07] is the start tag of the *Description* element.

1213 [m08] - [m10] provides a human-readable description about this key-cache policy.

1214 [m11] is the closing tag of the *Description* element.

1215 [m12] specifies the *KeyClass* to which this policy applies. Only keys that belong to this key-class are subject to this caching policy.

1217 [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.

1219 [m14] specifies the date and time that this *KeyCachePolicy* becomes invalid. This is accomplished through a *EndDate* element. In this example, the use of the UNIX “epoch” date (January 01, 1969) indicates that this policy never expires.

1222 [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.

1224 [m16] indicates the *Status* of this *KeyCachePolicy* and whether it is an active policy or not.

1225 [m17] is the closing tag of the *KeyCachePolicy* element.

1226 [m18] is the closing tag of the *KeyCachePolicyResponse* element.

1227 3.13 Response with a symmetric key-caching policy (2)

1228 This is a second example of a key-caching policy response that has additional elements in the policy permitting
 1229 caching and specify the number of unused and used (for encryption) symmetric keys that may be cached by the
 1230 client machine.

```

1231 [m01] <ekmi:KeyCachePolicyResponse
1232 [m02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1233 [m03]     xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
1234 [m04]     <ekmi:KeyCachePolicy>
1235 [m05]         <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
1236 [m06]         <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
1237 [m07]         <ekmi:Description>
1238 [m08]             This policy defines how company-issued laptops will manage symmetric
1239 [m09]             keys used for file/disk encryption in their local cache. This
1240 [m10]             policy must be used by all laptops that use the company EKMI.
1241 [m11]         </ekmi:Description>
1242 [m12]         <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
1243 [m13]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1244 [m14]         <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
1245 [m15]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1246 [m16]         <ekmi:Status>Active</ekmi:Status>
1247 [m17]     </ekmi:KeyCachePolicy>
  
```

```

1248 [m18]         <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1249 [m19]         <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1250 [m20]         </ekmi:NewKeysCacheDetail>
1251 [m21]         <ekmi:UsedKeysCacheDetail>
1252 [m22]         <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1253 [m23]         <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1254 [m24]         </ekmi:UsedKeysCacheDetail>
1255 [m25]         </ekmi:KeyCachePolicy>
1256 [m26] </ekmi:KeyCachePolicyResponse>

```

1257 [m01] is the start of the *KeyCachePolicyResponse* element.

1258 [m02] and [m03] identify the namespaces to which this XML conforms, and the location of their XML Schema
1259 Definitions (XSD).

1260 [m04] is the start of the first – and only - *KeyCachePolicy* element.

1261 [m05] identifies the *KeyCachePolicyID* (KCPID) assigned by the SKS server for the key-caching policy being
1262 returned.

1263 [m06] provides a descriptive name for this key-cache policy through the *PolicyName* element, which is helpful to
1264 human readers when identifying this policy.

1265 [m07] is the start tag of the *Description* element.

1266 [m08] - [m10] provides a human-readable description about this key-cache policy.

1267 [m11] is the closing tag of the *Description* element.

1268 [m12] specifies the *KeyClass* to which this policy applies. Only keys that belong to this key-class are subject to
1269 this caching policy.

1270 [m13] specifies the date and time that this *KeyCachePolicy* is effective. This is accomplished through a
1271 *StartDate* element. In this example, the policy is effective as of January 01, 2008.

1272 [m14] specifies the date and time that this *KeyCachePolicy* becomes invalid. This is accomplished through a
1273 *EndDate* element. In this example, the policy expires on December 31, 2008.

1274 [m15] specifies the frequency at which this client must check with the SKS server for updates to the key-caching
1275 policy. This is specified in seconds in the *PolicyCheckInterval* element; in this example it is a monthly interval.

1276 [m16] indicates the *Status* of this *KeyCachePolicy* and whether it is an active policy or not.

1277 [m17] is the start of the *NewKeysCacheDetail* element, which provides details about how many new symmetric
1278 keys – that haven't been used for any encryption transactions – may be cached by the client and for how long.

1279 [m18] indicates the maximum number of new (unused) symmetric keys that may be cached by the client. This is
1280 specified through the *MaximumKeys* element. When the client uses a symmetric key, this reduces the number
1281 of new symmetric keys. In this case, the SKCL connects to the SKS server (if it is on the network) and requests
1282 a new symmetric key to add to its new-key cache.

1283 [m19] indicates the maximum duration that new (unused) symmetric keys may be cached by the client. This is
1284 specified through the *MaximumDuration* element in seconds. If there are any new keys that exceed this
1285 duration limit, the SKCL deletes the specific symmetric key and replaces it with a new symmetric key from the
1286 SKS server.

1287 [m20] is the closing tag of the *NewKeysCacheDetail* element.

1288 [m21] is the start of the *UsedKeysCacheDetail* element, which provides details about how many used
1289 symmetric keys – those that HAVE been used for any encryption transactions – may be cached by the client and
1290 for how long.

1291 [m22] indicates the maximum number of used symmetric keys that may be cached by the client through the
1292 *MaximumKeys* element. If the client already has the maximum number of used-keys in its cache, using the

1293 First-In-First-Out (FIFO) method, it deletes the oldest symmetric key in the cache to replace with the key that
 1294 transitioned from the “new” to “used” status.

1295 [m23] indicates the maximum duration that used symmetric keys may be cached by the clien through the
 1296 **MaximumDuration** element in seconds. If there are any used keys that exceed this duration limit, the SKCL
 1297 deletes the specific symmetric key. While this may temporarily reduce the number of used symmetric keys in the
 1298 cache, the SKCL takes the most conservative position when making this decision.

1299 [m24] is the closing tag of the **UsedKeysCacheDetail** element.

1300 [m25] is the closing tag of the **KeyCachePolicy** element.

1301 [m26] is the closing tag of the **KeyCachePolicyResponse** element.

1302 3.14 Response with multiple symmetric key-caching policies (3)

1303 This is a third example of a key-caching policy response that has multiple key-cache policies that apply to
 1304 different classes of symmetric keys.

```

1305 [m01] <ekmi:KeyCachePolicyResponse
1306 [m02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1307 [m03]     xmlns:xenc='http://www.w3.org/2001/04/xmllenc#'>
1308 [m04]     <ekmi:KeyCachePolicy>
1309 [m05]         <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
1310 [m06]         <ekmi:PolicyName>No Caching Policy</ekmi:PolicyName>
1311 [m07]         <ekmi:Description>
1312 [m08]             This policy is for high-risk, always-connected machines on the
1313 [m09]             network, which will never cache symmetric keys locally. This
1314 [m10]             policy never expires (but checks monthly for any updates).
1315 [m11]         </ekmi:Description>
1316 [m12]         <ekmi:KeyClass>NoCachingClass</ekmi:KeyClass>
1317 [m13]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1318 [m14]         <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
1319 [m15]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1320 [m16]         <ekmi>Status>Active</ekmi>Status>
1321 [m17]     </ekmi:KeyCachePolicy>
1322 [m18]     <ekmi:KeyCachePolicy>
1323 [m19]         <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
1324 [m20]         <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
1325 [m21]         <ekmi:Description>
1326 [m22]             This policy defines how company-issued laptops will manage symmetric
1327 [m23]             keys used for file/disk encryption in their local cache. This
1328 [m24]             policy must be used by all laptops that use the company EKMI.
1329 [m25]         </ekmi:Description>
1330 [m26]         <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
1331 [m27]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1332 [m28]         <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
1333 [m29]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1334 [m30]         <ekmi>Status>Active</ekmi>Status>
1335 [m31]         <ekmi:NewKeysCacheDetail>
1336 [m32]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1337 [m33]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1338 [m34]         </ekmi:NewKeysCacheDetail>
1339 [m35]         <ekmi:UsedKeysCacheDetail>
1340 [m36]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1341 [m37]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1342 [m38]         </ekmi:UsedKeysCacheDetail>
1343 [m39]     </ekmi:KeyCachePolicy>
1344 [m40] </ekmi:KeyCachePolicy>
1345 [m41]     <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
  
```

```

1346 [m42] <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
1347 [m43] <ekmi:Description>
1348 [m44]     This policy defines how company-issued laptops will manage
1349 [m45]     symmetric keys used for file/disk encryption in their local
1350 [m46]     cache. This policy must be used by all laptops.
1351 [m47] </ekmi:Description>
1352 [m48] <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
1353 [m49] <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1354 [m50] <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
1355 [m51] <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1356 [m52] <ekmi:Status>Active</ekmi:Status>
1357 [m53] <ekmi:NewKeysCacheDetail>
1358 [m54]     <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1359 [m55]     <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1360 [m56] </ekmi:NewKeysCacheDetail>
1361 [m57] <ekmi:UsedKeysCacheDetail>
1362 [m58]     <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1363 [m59]     <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1364 [m60] </ekmi:UsedKeysCacheDetail>
1365 [m61] <ekmi:KeyCachePolicy>
1366 [m62] </ekmi:KeyCachePolicyResponse>

```

1367

1368 [m01] is the start of the *KeyCachePolicyResponse* element.

1369 [m02] and [m03] identify the namespaces to which this XML conforms, and the location of their XML Schema
1370 Definitions (XSD).

1371 [m04] is the start of the first of three *KeyCachePolicy* elements in this response.

1372 [m05] identifies the *KeyCachePolicyID* (KCPID) assigned to this policy by the SKS server. In this example, the
1373 concatenated values of the Domain ID and Policy ID indicate that the key belongs to the organization
1374 represented by the PEN, 10514; and is the first key-caching policy within the SKMS.

1375 [m06] provides a descriptive name for this key-cache policy through the *PolicyName* element, which is helpful to
1376 human readers when identifying this policy.

1377 [m07] is the start tag of the *Description* element.

1378 [m08] - [m10] provides a human-readable description about this key-cache policy.

1379 [m11] is the closing tag of the *Description* element.

1380 [m12] specifies the *KeyClass* to which this policy applies. Only keys that belong to this key-class are subject to
1381 this caching policy.

1382 [m13] specifies the date and time that this *KeyCachePolicy* is effective. This is accomplished through a
1383 *StartDate* element. In this example, the policy is effective as of January 01, 2008.

1384 [m14] specifies the date and time that this *KeyCachePolicy* becomes invalid. This is accomplished through a
1385 *EndDate* element. In this example, the use of the UNIX "epoch" date (January 01, 1969) indicates that this
1386 policy never expires.

1387 [m15] specifies the frequency at which this client must check with the SKS server for updates to the key-caching
1388 policy. This is specified in seconds in the *PolicyCheckInterval* element; in this example it is a monthly interval.

1389 [m16] indicates the *Status* of this *KeyCachePolicy* and whether it is an active policy or not.

1390 [m17] is the closing tag of the first *KeyCachePolicy* element.

1391 [m18] is the start of the second of three *KeyCachePolicy* elements in this response.

1392 [m19] identifies the **KeyCachePolicyID** (KCPID) assigned by the SKS server for the key-caching policy being
1393 returned.

1394 [m20] provides the descriptive name for this key-cache policy through the **PolicyName** element.

1395 [m21] is the start tag of the **Description** element.

1396 [m22] - [m24] provides a human-readable description about this key-cache policy.

1397 [m25] is the closing tag of the **Description** element.

1398 [m26] specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are subject to
1399 this caching policy.

1400 [m27] specifies the date and time that this **KeyCachePolicy** is effective. This is accomplished through a
1401 **StartDate** element. In this example, the policy is effective as of January 01, 2008.

1402 [m28] specifies the date and time that this **KeyCachePolicy** becomes invalid. This is accomplished through a
1403 **EndDate** element. In this example, the policy expires on December 31, 2008.

1404 [m29] specifies the frequency at which this client must check with the SKS server for updates to the key-caching
1405 policy. This is specified in seconds in the **PolicyCheckInterval** element; in this example it is a monthly interval.

1406 [m30] indicates the **Status** of this **KeyCachePolicy** and whether it is an active policy or not.

1407 [m31] is the start of the **NewKeysCacheDetail** element, which provides details about how many new symmetric
1408 keys – that haven't been used for any encryption transactions – may be cached by the client and for how long.

1409 [m32] indicates the maximum number of new (unused) symmetric keys that may be cached by the client. This is
1410 specified through the **MaximumKeys** element. When the client uses a symmetric key, this reduces the number
1411 of new symmetric keys. In this case, the SKCL connects to the SKS server (if it is on the network) and requests
1412 a new symmetric key to add to its new-key cache.

1413 [m33] indicates the maximum duration that new (unused) symmetric keys may be cached by the client. This is
1414 specified through the **MaximumDuration** element in seconds. If there are any new keys that exceed this
1415 duration limit, the SKCL deletes the specific symmetric key and replaces it with a new symmetric key from the
1416 SKS server.

1417 [m34] is the closing tag of the **NewKeysCacheDetail** element.

1418 [m35] is the start of the **UsedKeysCacheDetail** element, which provides details about how many used
1419 symmetric keys – those that HAVE been used for any encryption transactions – may be cached by the client and
1420 for how long.

1421 [m36] indicates the maximum number of used symmetric keys that may be cached by the client through the
1422 **MaximumKeys** element. If the client already has the maximum number of used-keys in its cache, using the
1423 First-In-First-Out (FIFO) method, it deletes the oldest symmetric key in the cache to replace with the key that
1424 transitioned from the “new” to “used” status.

1425 [m37] indicates the maximum duration that used symmetric keys may be cached by the client through the
1426 **MaximumDuration** element in seconds. If there are any used keys that exceed this duration limit, the SKCL
1427 deletes the specific symmetric key. While this may temporarily reduce the number of used symmetric keys in the
1428 cache, the SKCL takes the most conservative position when making this decision.

1429 [m38] is the closing tag of the **UsedKeysCacheDetail** element.

1430 [m39] is the closing tag of the second **KeyCachePolicy** element.

1431 [m40] is the start of the third **KeyCachePolicy** element in this response.

1432 [m41] identifies the **KeyCachePolicyID** (KCPID) assigned by the SKS server for the key-caching policy being
1433 returned.

1434 [m42] provides the descriptive name for this key-cache policy through the **PolicyName** element.

1435 [m43] is the start tag of the *Description* element.

1436 [m44] - [m46] provides a human-readable description about this key-cache policy.

1437 [m47] is the closing tag of the *Description* element.

1438 [m48] specifies the *KeyClass* to which this policy applies. Only keys that belong to this key-class are subject to
1439 this caching policy.

1440 [m49] specifies the date and time that this *KeyCachePolicy* is effective.

1441 [m50] specifies the date and time that this *KeyCachePolicy* becomes invalid.

1442 [m51] specifies the frequency at which this client must check with the SKS server for updates to the key-caching
1443 policy.

1444 [m52] indicates the *Status* of this *KeyCachePolicy* and whether it is an active policy or not.

1445 [m53] is the start of the *NewKeysCacheDetail* element, which provides details about how many new symmetric
1446 keys may be cached by the client and for how long.

1447 [m54] indicates the maximum number of new (unused) symmetric keys that may be cached by the client. This is
1448 specified through the *MaximumKeys* element. When the client uses a symmetric key, this reduces the number
1449 of new symmetric keys. In this case, the SKCL connects to the SKS server (if it is on the network) and requests
1450 a new symmetric key to add to its new-key cache.

1451 [m55] indicates the maximum duration that new (unused) symmetric keys may be cached by the client. This is
1452 specified through the *MaximumDuration* element in seconds. If there are any new keys that exceed this
1453 duration limit, the SKCL deletes the specific symmetric key and replaces it with a new symmetric key from the
1454 SKS server.

1455 [m56] is the closing tag of the *NewKeysCacheDetail* element.

1456 [m57] is the start of the *UsedKeysCacheDetail* element, which provides details about how many used
1457 symmetric keys – those that HAVE been used for any encryption transactions – may be cached by the client and
1458 for how long.

1459 [m58] indicates the maximum number of used symmetric keys that may be cached by the client through the
1460 *MaximumKeys* element. If the client already has the maximum number of used-keys in its cache, using the
1461 First-In-First-Out (FIFO) method, it deletes the oldest symmetric key in the cache to replace with the key that
1462 transitioned from the “new” to “used” status.

1463 [m59] indicates the maximum duration that used symmetric keys may be cached by the client through the
1464 *MaximumDuration* element in seconds. If there are any used keys that exceed this duration limit, the SKCL
1465 deletes the specific symmetric key. While this may temporarily reduce the number of used symmetric keys in the
1466 cache, the SKCL takes the most conservative position when making this decision.

1467 [m60] is the closing tag of the *UsedKeysCacheDetail* element.

1468 [m61] is the closing tag of the third and final *KeyCachePolicy* element in this response.

1469 [m62] is the closing tag of the *KeyCachePolicyResponse* element.

1470

1471 **Appendix A. Acknowledgments**

1472 The following individuals have participated in the creation of this specification and are gratefully
1473 acknowledged

1474 **Participants:**

1475 •

1476

1477

1478

Appendix B. Revision History

1479

Version	Date	Author	Notes
DRAFT 5	June 17, 2008	Arshad Noor	Initial version

1480 **Appendix C. Non-Normative Text**

1481