



Symmetric Key Services Markup Language (SKSML) Version 1.0

Committee Specification 01

18 January 2009

Specification URIs:

This Version:

<http://docs.oasis-open.org/ekmi/sksml/v1.0/cs01/SKSML-1.0-Specification.html>
<http://docs.oasis-open.org/ekmi/sksml/v1.0/cs01/SKSML-1.0-Specification.odt> (Authoritative)
<http://docs.oasis-open.org/ekmi/sksml/v1.0/cs01/SKSML-1.0-Specification.pdf>

Previous Version:

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

Latest Version:

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

Technical Committee:

OASIS Enterprise Key Management Infrastructure (EKMI) TC

Chair(s):

Arshad Noor

Editor(s):

Allen Schaaf

Related Work:

This specification replaces or supercedes:

- None

This specification is related to:

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

- XML Encryption - W3C Recommendation 10 Dec 2002. <http://www.w3.org/TR/xmlenc-core/>
- XML Signature - W3C Recommendation 12 Feb 2002. <http://www.w3.org/TR/xmldsig-core/>
- Web Services Security - SOAP Message Security 1.0 - OASIS Standard 200401, March 2004 - <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>

Declared XML Namespace(s):

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

Abstract:

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

Status:

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

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

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

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

Notices

Copyright © OASIS® 2008. All Rights Reserved.

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

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

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

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

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

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

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

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

Table of Contents

109		
110	1 Introduction.....	6
111	1.1 Terminology.....	6
112	1.2 Glossary.....	6
113	1.3 Normative References.....	7
114	2 Background (non-normative).....	8
115	2.1 Requirements (non-normative).....	9
116	3 Examples of use of SKSML (non-normative).....	11
117	3.1 Request for a new symmetric key.....	11
118	3.2 Response with a new symmetric key.....	13
119	3.3 Request for an existing symmetric key.....	17
120	3.4 Response with an existing symmetric key.....	18
121	3.5 Request for a new symmetric key of a specific KeyClass.....	18
122	3.6 Response with a new symmetric key of a specific KeyClass.....	18
123	3.7 Request for multiple new symmetric keys.....	19
124	3.8 Response with multiple new symmetric keys.....	20
125	3.9 Request for a symmetric key with an Encryption Certificate.....	25
126	3.10 Response with an SKS error.....	25
127	3.11 Response with symmetric keys and errors.....	26
128	3.12 Response with a pending Request ID.....	29
129	3.13 Request for an update of a pending Request ID.....	29
130	3.14 Request for a symmetric key-caching policy.....	30
131	3.15 Response with a symmetric key-caching policy (1).....	31
132	3.16 Response with a symmetric key-caching policy (2).....	33
133	3.17 Response with multiple symmetric key-caching policies (3).....	34
134	4 Specification.....	39
135	4.1 Element <SymkeyRequest>.....	39
136	4.2 Element <GlobalKeyID>.....	43
137	4.3 Element <KeyClasses> and <KeyClass>.....	44
138	4.4 Element <X509EncryptionCertificate>.....	46
139	4.5 Element <SymkeyRequestID>.....	46
140	4.6 Element <SymkeyResponse>.....	47
141	4.7 Element <Symkey>.....	51
142	4.8 Element <SymkeyWorkInProgress>.....	53
143	4.9 Element <SymkeyError>.....	55
144	4.10 Element <KeyUsePolicy>.....	56
145	4.11 Type TwoPartIDType.....	59
146	4.12 Element <KeyAlgorithm>.....	60
147	4.13 Element <KeySize>.....	61

148	4.14 Element <Status>.....	62
149	4.15 Element <Permissions>.....	63
150	4.16 Element <PermittedApplications> and <PermittedApplication>.....	69
151	4.17 Element <PermittedDates> and <PermittedDate>.....	72
152	4.18 Element <PermittedDays> and <PermittedDay>.....	73
153	4.19 Element <PermittedDuration>.....	75
154	4.20 Element <PermittedLevels> and <PermittedLevel>.....	76
155	4.21 Element <PermittedLocations> and <PermittedLocation>.....	77
156	4.22 Element <PermittedNumberOfTransactions>.....	80
157	4.23 Element <PermittedTimes> and <PermittedTime>.....	81
158	4.24 Element <PermittedUses> and <PermittedUse>.....	83
159	4.25 Element <KeyCachePolicyRequest>.....	84
160	4.26 Element <KeyCachePolicyResponse>.....	84
161	4.27 Element <KeyCachePolicy>.....	85
162	4.28 Type KeyCacheDetailType.....	88
163	4.29 Use of Web Services Security (WSS).....	90
164	4.30 Use of SKMS Error Codes & Messages.....	90
165	5 Conformance.....	91
166		

1 Introduction

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

1.1 Terminology

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

1.2 Glossary

3DES – Triple Data Encryption Standard

AES – Advanced Encryption Standard

Base64 – An encoding scheme for representing data

Ciphertext – Encrypted data

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

DES – Data Encryption Standard

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

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

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

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

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

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

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

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

Plaintext – Unencrypted data

SHA – Secure Hashing Algorithm

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

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

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

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

207 **SID** or **Server ID** – Server Identifier; the unique integer assigned to every **SKS** server within an

208 enterprise's **SKMS**

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

210 **SKMS** – Symmetric Key Management System; a collection of hardware and software providing

211 symmetric encryption key-management services

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

213 network or other mechanism selected by implementers

214 **SKSML** – Symmetric Key Services Markup Language; an XML-based protocol to request and receive

215 symmetric encryption key-management services

216 **SOAP** – Simple Object Access Protocol

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

218 **SOAP Envelope** – The SOAP message consisting of a SOAP Header and a SOAP Body, conforming to

219 the SOAP protocol standard.

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

221 **SOAP Header** – The header part of a SOAP message containing meta-information about the message,

222 including security-related objects

223 **Symkey** – A symmetric encryption key

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

225 **X509 Digital Certificate** – a digital certificate that conforms to the Internet Engineering Task Force's PKI

226 X509 standard

227 **XMLEncryption** – Encrypted content represented in eXtensible Markup Language that conforms to the

228 World Wide Web Consortium's XML Encryption standard

229 **XMLSignature** – A digital signature represented in eXtensible Markup Language that conforms to the

230 World Wide Web Consortium's XML Signature standard

231 1.3 Normative References

232	[AES]	Advanced Encryption Standard
233		NIST FIPS 197. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
234	[RFC 2119]	S. Bradner. <i>Key words for use in RFCs to Indicate Requirement Levels</i> .
235		IETF RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt
236	[SOAP]	Simple Object Access Protocol 1.1
237		W3C Recommendation 08 May 2000. http://www.w3.org/TR/soap/
238	[XMLEncryption]	XML Encryption Syntax and Processing
239		W3C Recommendation 10 Dec 2002. http://www.w3.org/TR/xmlenc-core/

240 **[XMLSignature]** XML Signature Syntax and Processing
241 W3C Recommendation 12 Feb 2002. <http://www.w3.org/TR/xmlsig-core/>
242 **[WSS]** Web Services Security – SOAP Message Security 1.0
243 OASIS Standard 200401, March 2004
244 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
245 [security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
246 **[RFC 2578]** K. McCloghrie, et al. *Structure of Management Information Version 2* (
247 *SMIv2*). IETF RFC 2578, April 1999. <http://www.ietf.org/html/rfc2578>
248

2 Background (non-normative)

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

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

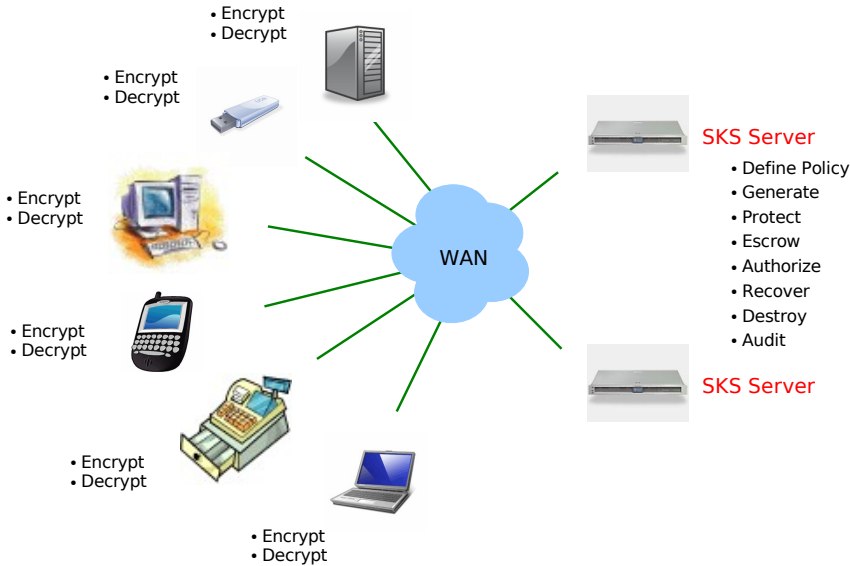
In a rush to provide solutions to the market, vendors have created many device-specific, platform-specific, database-specific and application-specific encryption and key-management tools. While these tools may be capable of performing their stated tasks adequately, a typical enterprise would have to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



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

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

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



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

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

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

2.1 Requirements (non-normative)

The requirements of the SKSML protocol are that:

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

SKSML meets the above requirements in the following manner:

301 • SKSML uses SOAP and XML for encapsulating its requests and responses and can thus, be
302 used on any platform that supports these two underlying protocols;

303 • Using a scheme that concatenates unique Domain identifiers (Private Enterprise Numbers
304 issued by the IANA), unique SKS Server identifiers within a domain and unique Key identifiers
305 within an SKS server, SKSML creates Global Key Identifiers (GKID) that can uniquely identify
306 symmetric keys across the internet;

307 • SKSML relies on the Web Services Security (WSS) standard 1.0, which in turn supports the use
308 of XML Signature and XML Encryption within the SOAP Header. Relying only the on the WSS
309 profile that uses RSA cryptographic key-pairs and digital certificates, SKSML uses the digital
310 signatures for authenticity and message-integrity, while using RSA-encryption for confidentiality;

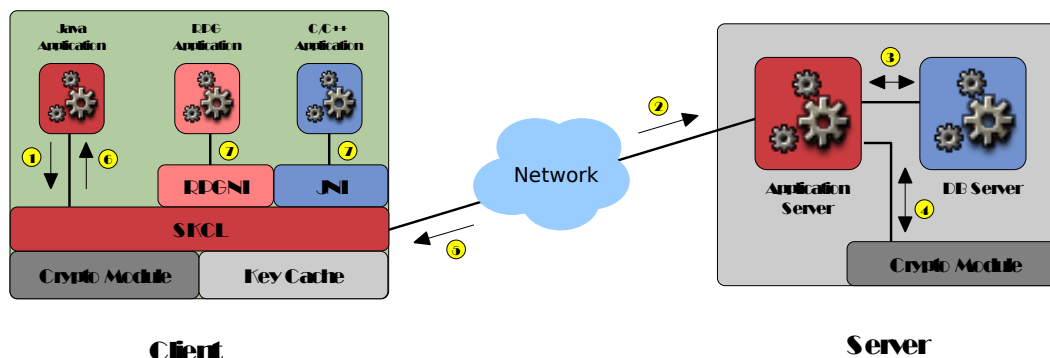
311 • Using secure key-caching enabled through centrally-defined policies, SKSML supports the
312 request and receipt of **KeyCachePolicy** elements by clients for the use of symmetric encryption
313 keys even when the client is disconnected from the network and an SKS server;

314 • SKSML provides significant flexibility for defining policies on how symmetric encryption keys
315 may be used by client applications. The **KeyUsePolicy** element allows Security Officers to
316 define which applications may use a specific key, days and times of use, location of use,
317 purpose of use, key-sizes, encryption algorithms, etc.

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

3 Examples of use of SKSML (non-normative)

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



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

3.1 Request for a new symmetric key

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

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

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

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

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

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

While the **SymkeyRequest** element is very simple, the Web Service Security (WSS) envelope – which provides security for all SKSML messages – expands the size of the message. The same request shown above, is displayed below in its entirety, with its WSS envelope. Please note that some content –


```

403 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
404     wsu:Id="XWSSGID-1172790300633-442423344">
405     <wsse:Reference URI="#XWSSGID-1172790302111-1738806553"
406         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
407 wss-x509-token-profile-1.0#X509v3"/>
408     </wsse:SecurityTokenReference>
409 </ds:KeyInfo>
410 </ds:Signature>
411 <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
412 wss-wssecurity-utility-1.0.xsd" wsu:Id="XWSSGID-1172790300637708871805">
413     <wsu:Created>2007-03-01T23:05:00Z</wsu:Created>
414     <wsu:Expires>2007-03-01T23:05:05Z</wsu:Expires>
415 </wsu:Timestamp>
416 </wsse:Security>
417 </SOAP-ENV:Header>
418 <SOAP-ENV:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
419 wssecurity-utility-1.0.xsd" wsu:Id="XWSSGID-1172790300636-653454040">
420     <ekmi:SymkeyRequest
421         xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
422         <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
423     </ekmi:SymkeyRequest>
424 </SOAP-ENV:Body>
425 </SOAP-ENV:Envelope>

```

3.2 Response with a new symmetric key

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

```

432 [b01] <ekmi:SymkeyResponse
433 [b02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
434 [b03]     xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
435 [b04]     <ekmi:Symkey>
436 [b05]         <ekmi:SymkeyRequestID>10514-1-7476</ekmi:SymkeyRequestID>
437 [b06]         <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
438 [b07]         <ekmi:KeyUsePolicy>
439 [b08]             <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>
440 [b09]             <ekmi:PolicyName>DES-EDE KeyUsePolicy</ekmi:PolicyName>
441 [b10]             <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
442 [b11]             <ekmi:KeyAlgorithm>
443 [b12]                 http://www.w3.org/2001/04/xmlenc#tripledes-cbc
444 [b13]             </ekmi:KeyAlgorithm>
445 [b14]             <ekmi:KeySize>192</ekmi:KeySize>
446 [b15]             <ekmi:Status>Active</ekmi:Status>
447 [b16]         <ekmi:Permissions>
448 [b17]             <ekmi:PermittedApplications ekmi:any="false">
449 [b18]                 <ekmi:PermittedApplication>
450 [b19]                     <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
451 [b20]                     <ekmi:ApplicationName>
452 [b21]                         Payroll Application
453 [b22]                     </ekmi:ApplicationName>
454 [b23]                     <ekmi:ApplicationVersion>1.0</ekmi:ApplicationVersion>
455 [b24]                     <ekmi:ApplicationDigestAlgorithm>
456 [b25]                         http://www.w3.org/2000/09/xmldsig#sha1
457 [b26]                     </ekmi:ApplicationDigestAlgorithm>

```

```

458 [b27]           <ekmi:ApplicationDigestValue>
459 [b28]             NIG4bKkt4cziEqFu0oBTM81efU=
460 [b29]           </ekmi:ApplicationDigestValue>
461 [b30]         </ekmi:PermittedApplication>
462 [b31]       </ekmi:PermittedApplications>
463 [b32]     <ekmi:PermittedDates ekmi:any="false">
464 [b33]       <ekmi:PermittedDate>
465 [b34]         <ekmi:StartDate>2008-01-01</ekmi:StartDate>
466 [b35]         <ekmi:EndDate>2008-12-31</ekmi:EndDate>
467 [b36]       </ekmi:PermittedDate>
468 [b37]     </ekmi:PermittedDates>
469 [b38]     <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
470 [b39]     <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
471 [b40]     <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
472 [b41]     <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
473 [b42]     <ekmi:PermittedNumberOfTransactions ekmi:any="true"
474 xsi:nil="true"/>
475 [b43]     <ekmi:PermittedTimes ekmi:any="false">
476 [b44]       <ekmi:PermittedTime>
477 [b45]         <ekmi:StartTime>07:00:00</ekmi:StartTime>
478 [b46]         <ekmi:EndTime>19:00:00</ekmi:EndTime>
479 [b47]       </ekmi:PermittedTime>
480 [b48]     </ekmi:PermittedTimes>
481 [b49]     <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
482 [b50]   </ekmi:Permissions>
483 [b51] </ekmi:KeyUsePolicy>
484 [b52]   <ekmi:EncryptionMethod
485 [b53]     Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
486 [b54]   <xenc:CipherData>
487 [b55]     <xenc:CipherValue>
488 [b56]       E9zWB/y93hVSzeTLiDcQoDxmLNxTuxSffMNwCJmtldIqzQHBnpdQ81g6DKdkCFjJM
489 [b57]       hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfglpU8tGFbpWZcd/ATpJD/2fw
490 [b58]       UJow/qimxi8+huUYJMtaGHtXuLlWtx27STRcRpIsY=
491 [b59]     </xenc:CipherValue>
492 [b60]   </xenc:CipherData>
493 [b61] </ekmi:Symkey>
494 [b62] </ekmi:SymkeyResponse>

```

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

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

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

500 [b05] identifies the **SymkeyRequestID** assigned by the SKS server for this request. In this example, the concatenated values of the Domain ID, Server ID and Request 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 7,476th unique request received on that SKS server. The client and the server use this value to associate asynchronous requests and responses for symmetric keys between themselves; however, the value is also returned for synchronous request/responses too.

506 [b06] 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.

510 [b07] is the start of the **KeyUsePolicy** element. This element contains details of the policy to which
511 SKCL implementations must conform when using the symmetric key.

512 [b08] identifies the unique **KeyUsePolicyID** (*KUPID*) which identifies this policy within the SKMS.

513 [b09] provides a descriptive name for this key-use policy, which is helpful to human readers when
514 identifying this policy.

515 [b10] identifies the **KeyClass** to which this symmetric key belongs. Key-classes are useful to
516 applications that wish to encrypt plaintext with a key that has specific characteristics. The requesting
517 application is expected to know what **KeyClass** it needs before it asks for a key corresponding to that
518 class.

519 [b11] is the start tag of the **KeyAlgorithm** element.

520 [b12] identifies the cryptographic algorithm that this symmetric key must be used with to for
521 cryptographic operations.

522 [b13] is the closing tag of the **KeyAlgorithm** element.

523 [b14] specifies the size of the symmetric encryption key in bits. While it is possible for application
524 developers to determine this programmatically from the key-object, this element provides this information
525 as a convenience.

526 [b15] indicates the **Status** of this **KeyUsePolicy** and whether it is an active policy or not. This is useful
527 in situations where an application may wish to re-use a symmetric key to encrypt related data to the data
528 originally encrypted with the symmetric key. While it is possible for the symmetric key object to be active
529 in the database, it is conceivable that the **KeyUsePolicy** used by the key has changed and the
530 application technically needs to use a new symmetric key to encrypt new data.

531 [b16] is the start of the **Permissions** element. This element provides a sophisticated mechanism for
532 controlling how, where, when and by which applications symmetric keys be used. While there are many
533 sub-elements within a **Permissions** element, not all **KeyUsePolicy** objects might use all **Permissions**
534 sub-elements. The example shown in this section only uses three of the possible **Permissions** sub-
535 elements.

536 [b17] is the start of the **PermittedApplications** element. This element allows SKMS policies to be
537 defined that allow only specific applications to use symmetric encryption keys associated with this policy.
538 The **ekmi:any="true"** attribute of the **PermittedApplications** element indicates that not just any
539 application on the client machine is permitted to use this symmetric key; only the specified applications
540 within the sub-elements of this element are permitted to use the symmetric key in question..

541 [b18] is the start of the first **PermittedApplication** element. This element identifies a specific
542 application within a list of **PermittedApplications** that is allowed to use the symmetric key. There can
543 be any number of **PermittedApplication** elements with **PermittedApplications**.

544 [b19] identifies the unique **ApplicationID** (as identified within the SKMS) of the **PermittedApplication**.

545 [b20] is the start of the **ApplicationName** element.

546 [b21] identifies the **ApplicationName** of the **PermittedApplication** (as identified within the SKMS).

547 [b22] is the closing tag of the **ApplicationName** element.

548 [b23] identifies the specific **ApplicationVersion** of the **PermittedApplication**. This is helpful when
549 there are multiple versions of a specific application, and the policy-makers need to distinguish between
550 the symmetric keys in use by a specific version of the application.

551 [b24] is the start of the **ApplicationDigestAlgorithm** element. This element permits enterprises to
552 ensure that only a cryptographically-verified application is authorized to use the symmetric encryption

553 key. This assumes that the implementation has an infrastructure where the SKCL is capable of
554 determining a cryptographic value to uniquely identify an application within the run-time environment.

555 [b25] identifies the W3C-specified URL of the cryptographic algorithm used to calculate the message
556 digest of the application's image.

557 [b26] is the closing tag of the **ApplicationDigestAlgorithm** element.

558 [b27] is the start of the **ApplicationDigestValue** element. This element permits enterprises to ensure
559 that only a cryptographically-verified application is authorized to use the symmetric encryption key. This
560 assumes that the implementation has an infrastructure where the SKCL is capable of determining a
561 cryptographic value to uniquely identify an application within the run-time environment.

562 [b28] identifies the Base64-encoded message digest of the **PermittedApplication**'s image, based on
563 the algorithm specified in the **ApplicationDigestAlgorithm** element.

564 [b29] is the closing tag of the **ApplicationDigestValue** element.

565 [b30] is the closing tag of the **PermittedApplication** element.

566 [b31] is the closing tag of the **PermittedApplications** element.

567 [b32] is the start of the **PermittedDates** element. This element permits enterprises to ensure that the
568 symmetric encryption key can be used only during a specified date period. This assumes that the
569 implementation has an infrastructure where the SKCL is capable of accurately determining the current
570 date within the run-time environment.

571 [b33] is the start of the first **PermittedDate** element. There can be any number of **PermittedDate**
572 elements within a **PermittedDates** element.

573 [b34] identifies the **StartDate** of the duration period during which the symmetric encryption key can be
574 used by authorized applications.

575 [b35] identifies the **EndDate** of the duration period during which the symmetric encryption key can be
576 used by authorized applications.

577 [b36] is the closing tag of the **PermittedDate** element.

578 [b37] is the closing tag of the **PermittedDates** element.

579 [b38] is an empty (null) **PermittedDays** element. This element permits enterprises to ensure that the
580 symmetric encryption key can be used only on specific days of the week. This assumes that the
581 implementation has an infrastructure where the SKCL is capable of accurately determining the current
582 day-of-week within the run-time environment. In this specific instance, the null element indicates that
583 this permission does not apply to this symmetric key, and therefore, there are no constraints on its use.
584 However, the key is still subject to all non-null permission clauses.

585 [b39] is an empty (null) **PermittedDuration** element. This element permits enterprises to ensure that
586 the symmetric encryption key can be used only for a specific duration of time once the symmetric key
587 has been used for the first time on the client. This assumes that the implementation has an
588 infrastructure where the SKCL is capable of accurately determining the current time within the run-time
589 environment. In this specific instance, the null element indicates that this permission does not apply to
590 this symmetric key, and therefore, there are no constraints on its use. However, the key is still subject to
591 all non-null permission clauses.

592 [b40] is an empty (null) **PermittedLevels** element. This element permits enterprises to ensure that the
593 symmetric encryption key can be used only by applications that are classified at a given level within the
594 Multi-Level Security (MLS) system as defined in the Bell-LaPadula model. In this specific instance, the

595 null element indicates that this permission does not apply to this symmetric key, and therefore, there are
596 no constraints on its use. However, the key is still subject to all non-null permission clauses.

597 **[b41]** is an empty (null) **PermittedLocations** element. This element permits enterprises to ensure that
598 the symmetric encryption key can be used only by applications at specified geographic locations on the
599 planet. This assumes that the implementation has an infrastructure where the SKCL is capable of
600 accurately determining the current GPS location of the client within the run-time environment. In this
601 specific instance, the null element indicates that this permission does not apply to this symmetric key,
602 and therefore, there are no constraints on its use. However, the key is still subject to all non-null
603 permission clauses.

604 **[b42]** is an empty (null) **PermittedNumberOfTransactions** element. This element permits enterprises
605 to ensure that the symmetric encryption key can be used by applications only for a specified number of
606 encryption transactions. In this specific instance, the null element indicates that this permission does
607 not apply to this symmetric key, and therefore, there are no constraints on its use. However, the key is
608 still subject to all non-null permission clauses.

609 **[b43]** is the start of the **PermittedTimes** element. This element permits enterprises to ensure that the
610 symmetric encryption key can be used only during a specified time period within any date. This
611 assumes that the implementation has an infrastructure where the SKCL is capable of accurately
612 determining the current time within the run-time environment.

613 **[b44]** is the start of the first **PermittedTime** element. There can be any number of **PermittedTime**
614 elements within a **PermittedTimes** element.

615 **[b45]** identifies the **StartTime** of the duration period during which the symmetric encryption key can be
616 used by authorized applications.

617 **[b46]** identifies the **EndTime** of the duration period during which the symmetric encryption key can be
618 used by authorized applications.

619 **[b47]** is the closing tag of the **PermittedTime** element.

620 **[b48]** is the closing tag of the **PermittedTimes** element.

621 **[b49]** is an empty (null) **PermittedUses** element. This element permits enterprises to ensure that the
622 symmetric encryption key can be used by applications for specific uses. In this specific instance, the null
623 element indicates that this permission does not apply to this symmetric key, and therefore, there are no
624 constraints on its use. However, the key is still subject to all non-null permission clauses.

625 **[b50]** is the closing tag of the **Permissions** element.

626 **[b51]** is the closing tag of the **KeyUsePolicy** element.

627 **[b52]-[b53]** identifies the encryption algorithm used in the **EncryptionMethod** element to encrypt the
628 symmetric encryption key itself, to transport to the requesting client. The symmetric key is encrypted
629 using the **PublicKey** or the requesting client. The **Algorithm** attribute uses the W3C-specified URLs for
630 identifying the encryption and padding algorithms.

631 **[b54]** is the start of the **CipherData** element. This element is from the W3C XML Encryption namespace
632 (as identified by the "xenc" qualifier in the element name).

633 **[b55]** is the start of the **CipherValue** element. This element contains the Base64-encoded ciphertext of
634 the symmetric encryption key.

635 **[b56] – [b58]** is the Base64-encoded ciphertext of the symmetric encryption key.

636 **[b59]** is the closing tag of the **CipherValue** element.

637 **[b60]** is the closing tag of the **CipherData** element.

638 [b61] is the closing tag of the **Symkey** element.
639 [b62] is the closing tag of the **SymkeyResponse** element.

640 3.3 Request for an existing symmetric key

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

647 The client application (that has been linked to the SKCL) will call an API method within the SKCL for the
648 appropriate symmetric key. After the SKCL has ensured that the application is authorized to make such
649 a request, and assuming that the client application needs the key with the **GlobalKeyID** value of **10514-1-
650 235**, the SKCL assembles the following SKSML request. (The SOAP message is secured using WSS,
651 but only the actual SKSML content is displayed and discussed here).

```
652 [c01] <ekmi:SymkeyRequest  
653 [c02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
654 [c03]     <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>  
655 [c04] </ekmi:SymkeyRequest>
```

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

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

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

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

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

665 3.4 Response with an existing symmetric key

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

671 3.5 Request for a new symmetric key of a specific KeyClass

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

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

```
680 [e01] <ekmi:SymkeyRequest  
681 [e02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
682 [e03]     <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
683 [e04]     <ekmi:KeyClasses>  
684 [e05]         <ekmi:KeyClass>HR-Class</ekmi:KeyClass>  
685 [e06]     </ekmi:KeyClasses>  
686 [e07] </ekmi:SymkeyRequest>
```

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

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

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

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

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

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

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

696 3.6 Response with a new symmetric key of a specific **KeyClass**

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

703 3.7 Request for multiple new symmetric keys

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

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

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

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

```
[g01] <ekmi:SymkeyRequest
[g02]   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
[g03]   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
[g04]   <ekmi:KeyClasses>
[g05]     <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
[g06]     <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
[g07]     <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
[g08]     <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
[g09]     <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
[g10]     <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
[g11]     <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
[g12]     <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
[g13]     <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>
[g14]   </ekmi:KeyClasses>
[g15] </ekmi:SymkeyRequest>
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

773 3.8 Response with multiple new symmetric keys

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

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

```
784 [h01] <ekmi:SymkeyResponse
785 [h02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
786 [h03]     xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
787 [h04]     <ekmi:Symkey>
788 [h05]         <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
789 [h06]         <ekmi:GlobalKeyID>10514-4-3792</ekmi:GlobalKeyID>
790 [h07]         <ekmi:KeyUsePolicy>
791 [h08]             <ekmi:KeyUsePolicyID>10514-9</ekmi:KeyUsePolicyID>
792 [h09]             <ekmi:PolicyName>DES-EDE KeyUsePolicy for EHR-CDC</ekmi:PolicyName>
793 [h10]             <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
794 [h11]             <ekmi:KeyAlgorithm>
795 [h12]                 http://www.w3.org/2001/04/xmlenc#tripledes-cbc
796 [h13]             </ekmi:KeyAlgorithm>
797 [h14]             <ekmi:KeySize>192</ekmi:KeySize>
798 [h15]             <ekmi:Status>Active</ekmi:Status>
799 [h16]             <ekmi:Permissions>
800 [h17]                 <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
801 [h18]                 <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
802 [h19]                 <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
803 [h20]                 <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
804 [h21]                 <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
805 [h22]                 <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
806 [h23]                 <ekmi:PermittedNumberOfTransactions
807 [h24]                     ekmi:any="true" xsi:nil="true"/>
808 [h25]                 <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
809 [h26]                 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
810 [h27]             </ekmi:Permissions>
811 [h28]         </ekmi:KeyUsePolicy>
812 [h29]         <ekmi:EncryptionMethod
813 [h30]             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
814 [h31]         <xenc:CipherData>
815 [h32]             <xenc:CipherValue>
816 [h33]                 E9zwB/y93hVSzeTLiDcQoDxmLNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJMa1w
```

```

817 [h34] hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfg1pU8tLWtx27STRcR/2fw2ava
818 [h35] ULWtx27STRcRJMtaGHtXuLLWtx27STRcRpIsY=
819 [h36] </xenc:CipherValue>
820 [h37] </xenc:CipherData>
821 [h38] </ekmi:Symkey>
822 [h39] <ekmi:Symkey>
823 [h40] <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
824 [h41] <ekmi:GlobalKeyID>10514-4-3793</ekmi:GlobalKeyID>
825 [h42] <ekmi:KeyUsePolicy>
826 [h43] <ekmi:KeyUsePolicyID>10514-12</ekmi:KeyUsePolicyID>
827 [h44] <ekmi:PolicyName>DES-EDE KeyUsePolicy for EHR-CR0</ekmi:PolicyName>
828 [h45] <ekmi:KeyClass>EHR-CR0</ekmi:KeyClass>
829 [h46] <ekmi:KeyAlgorithm>
830 [h47] http://www.w3.org/2001/04/xmlenc#tripledes-cbc
831 [h48] </ekmi:KeyAlgorithm>
832 [h49] <ekmi:KeySize>192</ekmi:KeySize>
833 [h50] <ekmi:Status>Active</ekmi:Status>
834 [h51] <ekmi:Permissions>
835 [h52] <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
836 [h53] <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
837 [h54] <ekmi:PermittedDate>
838 [h55] <ekmi:StartDate>2008-01-01</ekmi:StartDate>
839 [h56] <ekmi:EndDate>2009-12-31</ekmi:EndDate>
840 [h57] </ekmi:PermittedDate>
841 [h58] </ekmi:PermittedDates>
842 [h59] <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
843 [h60] <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
844 [h61] <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
845 [h62] <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
846 [h63] <ekmi:PermittedNumberOfTransactions
847 [h64] ekmi:any="true" xsi:nil="true"/>
848 [h65] <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
849 [h66] <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
850 [h67] </ekmi:Permissions>
851 [h68] </ekmi:KeyUsePolicy>
852 [h69] <ekmi:EncryptionMethod
853 [h70] Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
854 [h71] <xenc:CipherData>
855 [h72] <xenc:CipherValue>
856 [h73] qUiQXG0ca8EU871zBoXBjDoDxmLNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJM1
857 [h74] hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/2fw1
858 [h75] UJow/qimxi8+ huUYJMtaGHtXuLLWtx27STRcRpIsY=
859 [h76] </xenc:CipherValue>
860 [h77] </xenc:CipherData>
861 [h78] </ekmi:Symkey>
862 [h79] <ekmi:Symkey>
863 [h80] <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
864 [h81] <ekmi:GlobalKeyID>10514-4-3795</ekmi:GlobalKeyID>
865 ...
866 [h82] <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
867 ...
868 [h83] </ekmi:Symkey>
869 [h84] <ekmi:Symkey>
870 [h85] <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
871 [h86] <ekmi:GlobalKeyID>10514-4-3797</ekmi:GlobalKeyID>
872 ...
873 [h87] <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
874 ...
875 [h88] </ekmi:Symkey>

```



```

876 [h89]      <ekmi:Symkey>
877 [h90]      <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
878 [h91]      <ekmi:GlobalKeyID>10514-4-3798</ekmi:GlobalKeyID>
879      ...
880 [h92]      <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
881      ...
882 [h93]      </ekmi:Symkey>
883 [h94]      <ekmi:Symkey>
884 [h95]      <ekmi:GlobalKeyID>10514-4-3799</ekmi:GlobalKeyID>
885      ...
886 [h96]      <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
887      ...
888 [h97]      </ekmi:Symkey>
889 [h98]      <ekmi:Symkey>
890 [h99]      <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
891 [h100]     <ekmi:GlobalKeyID>10514-4-3801</ekmi:GlobalKeyID>
892      ...
893 [h101]     <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
894      ...
895 [h102]     </ekmi:Symkey>
896 [h103]     <ekmi:Symkey>
897 [h104]     <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
898 [h105]     <ekmi:GlobalKeyID>10514-4-3803</ekmi:GlobalKeyID>
899      ...
900 [h106]     <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
901      ...
902 [h107]     </ekmi:Symkey>
903 [h108]     <ekmi:Symkey>
904 [h109]     <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
905 [h110]     <ekmi:GlobalKeyID>10514-4-3805</ekmi:GlobalKeyID>
906      ...
907 [h111]     <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>
908      ...
909 [h112]     </ekmi:Symkey>
910 [h113]     </ekmi:SymkeyResponse>

```

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

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

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

916 [h05] identifies the **SymkeyRequestID** assigned by the SKS server for this request. In this example, the
917 concatenated values of the Domain ID, Server ID and Request ID indicate that the key belongs to the
918 organization represented by the PEN, 10514; it was generated on an SKS server with a Server ID of 4
919 and is the 78,122nd unique request received on that SKS server. The client and the server use this value
920 to associate asynchronous requests and responses for symmetric keys between themselves; however,
921 the value is also returned for synchronous request/responses too.

922 [h06] identifies the **GlobalKeyID** (GKID) assigned by the SKS server of this first symmetric key. In this
923 example, the concatenated values of the Domain ID, Server ID and Key ID indicate that the key belongs
924 to the organization represented by the PEN, 10514; it was generated on an SKS server with a Server ID
925 of 4 and is the 3792nd unique key generated on that SKS server.

926 [h07] is the start of the **KeyUsePolicy** element that applies just to this symmetric key. This element
927 contains details of the policy to which SKCL implementations must conform when using the symmetric
928 key.

929 [h08] identifies the unique **KeyUsePolicyID** (*KUPID*) which identifies this policy within the SKMS.

930 [h09] provides a descriptive name for this key-use policy, which is helpful to human readers when
931 identifying this policy.

932 [h10] identifies the **KeyClass** to which this symmetric key belongs. In the case of this example, the first
933 symmetric key in the response conforms to the **EHR-CDC** class which, presumably, might be a key that
934 covers data encrypted for/by the Center for Disease Control (CDC) within an Electronic Health Record
935 (EHR) system. Key-classes are useful to applications that wish to encrypt plaintext with a key that has
936 specific characteristics. The requesting application is expected to know what **KeyClass** it needs before
937 it asks for a key corresponding to that class.

938 [h11] is the start tag of the **KeyAlgorithm** element.

939 [h12] identifies the cryptographic algorithm that this symmetric key must be used with. For this
940 symmetric key example, the algorithm is the Triple Data Encryption Standard (3DES) with Cipher Block
941 Chaining (CBC) padding. The URL is a standard notation for this algorithm and padding as defined
942 within [XMLEncryption]..

943 [h13] is the closing tag of the **KeyAlgorithm** element.

944 [h14] specifies the size of the symmetric encryption key in bits. For this Triple-DES key, it is 192-bits.

945 [h15] indicates the **Status** of this **KeyUsePolicy** and whether it is an active policy or not. This is useful
946 in situations where an application may wish to re-use a symmetric key to encrypt related data to the data
947 originally encrypted with the symmetric key. While it is possible for the symmetric key object to be active
948 in the database, it is conceivable that the **KeyUsePolicy** used by the key has changed and the
949 application technically needs to use a new symmetric key to encrypt new data.

950 [h16] is the start of the **Permissions** element. This element provides a sophisticated mechanism for
951 controlling how, where, when and by which applications symmetric keys be used. While there are many
952 sub-elements within a **Permissions** element, not all **KeyUsePolicy** objects might use all **Permissions**
953 sub-elements<ekmi:RequestedKeyClass>Payroll-Tax-Class</ekmi:RequestedKeyClass>. The example
954 shown for this symmetric key indicates that there are no other specific restrictions on the use of this
955 symmetric key by authorized client applications; i.e. any authorized client application may use it at any
956 time, on any date, in any location for any purpose.

957 [h17] is the start and end of the null **PermittedApplications** element. This implies that there are no
958 restrictions on which application can use this symmetric key.

959 [h18] is the start and end of the null **PermittedDates** element. This implies that there are no date
960 restrictions on when this symmetric key can be used.

961 [h19] is the start and end of the null **PermittedDays** element. This implies that there are no day-of-week
962 restrictions on when this symmetric key can be used.

963 [h20] is the start and end of the null **PermittedDuration** element. This implies that there are no
964 restrictions to how long this symmetric key may be used.

965 [h21] is the start and end of the null **PermittedLevels** element. This implies that there are no
966 restrictions on the MLS security level in which this symmetric key can be used.

967 [h22] is the start and end of the null **PermittedLocations** element. This implies that there are no
968 geophysical restrictions where this symmetric key can be used.

969 [h23] - [h24] is the start and end of the null **PermittedNumberOfTransactions** element. This implies
970 that there are no restrictions on how many encryption transactions that can be performed by this
971 symmetric key.

972 [h25] is the start and end of the null **PermittedTimes** element. This implies that there are no time-of-day
973 restrictions when this symmetric key can be used.

974 [h26] is the start and end of the null **PermittedUses** element. This implies that there are no restrictions
975 how this symmetric key can be used by applications.

976 [h27] is the closing tag of the **Permissions** element.

977 [h28] is the closing tag of the **KeyUsePolicy** element.

978 [h29] and [h30] identify the encryption algorithm used in the **EncryptionMethod** element to encrypt the
979 symmetric encryption key itself, to transport to the requesting client. The symmetric key is encrypted
980 using the **PublicKey** or the requesting client. The **Algorithm** attribute uses the W3C-specified URLs for
981 identifying the encryption and padding algorithms.

982 [h31] is the start of the **CipherData** element. This element is from the W3C XML Encryption namespace
983 (as identified by the “xenc” qualifier in the element name).

984 [h32] is the start of the **CipherValue** element. This element contains the Base64-encoded ciphertext of
985 the symmetric encryption key.

986 [h33] – [h35] is the Base64-encoded ciphertext of the symmetric encryption key.

987 [h36] is the closing tag of the **CipherValue** element.

988 [h37] is the closing tag of the **CipherData** element.

989 [h38] is the closing tag of the first **Symkey** element within this **SymkeyResponse**.

990 [h39] - [h78] represents the second **Symkey** element in this **SymkeyResponse**. The differences in
991 this symmetric key element from the first, can be summarized as follows:

- 992 • [h41] identifies a different **GlobalKeyID** (10514-4-3793) for this symmetric key;
- 993 • [h43] identifies a different **KeyUsePolicy** (10514-12) for this symmetric key;
- 994 • [h45] identifies a different **KeyClass** (EHR-CRO) for this symmetric key;
- 995 • [h51] - [h67] defines a different **Permissions** element for this symmetric key;
- 996 • [h73] - [h75] contains a different **CipherValue** for this symmetric key;

997 [h79] – [h83] is the container for the third **Symkey** element in this response. For the sake of brevity, all
998 the usual **Symkey** elements have been dispensed with, but the unique **GlobalKeyID** and **KeyClass** are
999 shown to indicate the SKS server's response to the request. In this example, they are 10514-4-3795 and
1000 EHR-DEF respectively.

1001 [h84] – [h113] contain the remaining **Symkey** elements of this **SymkeyResponse**. Once again, for
1002 brevity, all the details of the **Symkey** elements are dispensed with, except for the unique GKIDs and
1003 KeyClasses.

1004 [h105] is the closing tag of the **SymkeyResponse** element.

1005 Note that it is possible for a request to contain the same **KeyClass** multiple times; there is no
1006 requirement that they need to be unique within a request if an application has a legitimate business need
1007 for multiple symmetric keys of the same **KeyClass**. The SKS server will respond with unique symmetric
1008 keys, all belonging to the **KeyClass** requested by the client application.

1009 An additional note is that the **SymkeyRequestID** is unchanged for every symmetric-key response
1010 element in this example. This is because a single request was responsible for the generation of all
1011 these keys, and as a result, every **Symkey** element contains the same **SymkeyRequestID**.

1012

1013 3.9 Request for a symmetric key with an Encryption Certificate

1014 Within an SKMS, all requests and responses are digitally signed to ensure message-authenticity and
1015 integrity. In addition, the symmetric-key payload in the response is also encrypted with the Public Key of
1016 the requesting client's X509 digital certificate for message-confidentiality.

1017 While it is always possible to build an SKMS that can find the encryption certificates of requesting
1018 clients, either within its database or from a Lightweight Directory Access Protocol (LDAP) directory on
1019 the network, it is sometimes efficient, or even necessary, to send the encryption certificate of the client
1020 with the symmetric key request.. The following example shows such a request.

```
1021 [i01] <ekmi:SymkeyRequest
1022 [i02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1023 [i03]     <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
1024 [i04]     <ekmi:X509EncryptionCertificate>
1025         MIIDfDCCAmSgAwIBAgIIAe/AvliGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdu3Ryb25nab
1026         S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvcjBTdHJvbmdLZXkgREVNTyBVc2Ugld
1027         T25seTEuXzUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUu
1028         A1UECmMhbm9yIFN0cm9uZ0t0eSBERU1PIFVzZSBPbm5MRcwFQYDVQKEw5TdHJvbmdBdXR0IEl2da
1029         S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvcjBTdHJvbmdLZXkgREVNTyBVc2Ugla
1030         T25seTEuXzUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUuZGUu
1031         NjEwWjBpMREwDwYKCZImiZPyLGBARMB0TEVMBMGA1UEAxMMU0tTIFNlcnZlci0xMSQwIgYDVQs2dw2
1032         ExtGb3IgU3Ryb25nS2V5IERFTU8gVXNlIE9ubHkxZzAVBgNVBAoTDlN0cm9uZ0FldGggSW5jMIIBd2
1033         NBGkqhkiG9w0BAQEFAAOCQA8AMIIBCgKCAQEAztppqRoU5A8plxx1Rz1QEUnlAAM1D5g9+isIr3wxa
1034         hbwjtFSMYilnY4iV77xU/nsM0nMZ7RxsLYKdCzQ10DVYqQwqmAvaJ5Z6SVy34gZ51YG+rSWE3NjFsd
1035         b0XW8RJYA/Tn6Lmht/qngrcaqgmtP0cAAiMRZ0WtCTmC2K/LEqDabXSyU6Hh8ySNE3njybvwmWpresf
1036         zsYokTdvWQqT6tKo10wJsdl1+hxm7DrnMLvMNq5reINfsKhDdX17wzhrBUx+hiYA/qo8tMXkL6wsd
1037         4PN5dYugtZpSzIdU05tIg58Avhzw07hy5oofBkFY22CeljQ36u0bMjuyGj6UYHs3rdfsds32rda
1038         YzCBnzANBkgqhkiG9w0BAQEFAA0BjQAwGyKCGYEAyAmxMZHYA8wHJ4UE4b61s51JVWe4Fygj4Mcf3a
1039         hvcNAQELBQADggEBACK05PtvZD4WPgl0e=
1040 [i05]     </ekmi:X509EncryptionCertificate>
1041 [i06] </ekmi:SymkeyRequest>
```

1042 [i01] is the start of the **SymkeyRequest** element.

1043 [i02] identifies the namespace to which this XML conforms, and the location of its XML Schema
1044 Definition (XSD).

1045 [i03] identifies the **GlobalKeyID** (GKID) being requested by the client application. The “zero” value for
1046 the Server ID and the Key ID indicates a request for a new symmetric key.

1047 [i04] is the start of the **X509EncryptionCertificate** element. All the lines between [i04] and [i05]
1048 represent the Base64-encoded X509-compliant digital certificate of the requesting client, with the
1049 encryption-usage enabled in the certificate.

1050 [i05] is the closing tag of the **X509EncryptionCertificate** element.

1051 [i06] is the closing tag of the **SymkeyRequest** element.

1052 3.10 Response with an SKS error

1053 While one hopes that all authorized requesters will get favorable responses from the SKS server, there
1054 are situations in which the client application can receive an error to a request for a symmetric key. The
1055 following XML shows one example of such an error response. Depending on the type of error, the actual
1056 message content might be different.

```

1057 [j01]    <ekmi:SymkeyResponse
1058 [j02]        xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1059 [j03]        xmlns:xenc='http://www.w3.org/2001/04/xmldenc#>'
1060 [j04]    <ekmi:SymkeyError>
1061 [j05]        <ekmi:SymkeyRequestID>10514-2-1044</ekmi:SymkeyRequestID>
1062 [j06]        <ekmi:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>
1063 [j07]        <ekmi:RequestedKeyClass>Payroll</ekmi:RequestedKeyClass>
1064 [j08]        <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
1065 [j09]        <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
1066 [j10]    </ekmi:SymkeyError>
1067 [j11]    </ekmi:SymkeyResponse>

```

1068 [j01] is the start of the **SymkeyResponse** element.

1069 [j02] and [j03] identify the namespaces to which this XML conforms, and the location of their XML Schema Definitions (XSD).

1071 [j04] is the start of the **SymkeyError** element, which tells the Symmetric Key Client Library (SKCL) that the request for a symmetric key resulted in an error.

1073 [j05] identifies the **SymkeyRequestID** assigned by the SKS server for this request. In this example, the concatenated values of the Domain ID, Server ID and Request 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 1,044th unique request received on that SKS server.

1077 [j06] indicates the **RequestedGlobalKeyID** the client requested. Returning the GKID in the error response allows the SKCL to associate the error message with the requesting application on the client machine.

1080 [j07] indicates the **RequestedKeyClass** the client requested. Returning the key-class in the error response allows the SKCL to associate the error message with the requesting application on the client machine.

1083 [j08] is an **ErrorCode** returned by the SKS server. The code may be one of the standard error codes defined in this specification, or may be a vendor-specific error code.

1085 [j09] is the text of the **ErrorMessage**, localized to the region and language of the requesting client application.

1087 [j10] is the closing tag of the **SymkeyError** tag.

1088 [j11] is the closing tag of the **SymkeyResponse** tag.

1089 3.11 Response with symmetric keys and errors

1090 When a client application requests multiple symmetric keys, the SKS server may respond in one of three ways. The SKS server may:

- 1092 i. Return all symmetric keys as requested;
- 1093 ii. Return no symmetric keys – i.e. it returns all errors;
- 1094 iii. Return some symmetric keys and some errors.

1095 The following SKSML shows the third case, where the server returns some symmetric keys and errors in response to a request for multiple keys (such as the one shown in **Section 3.7 Request for multiple new symmetric keys**).

1098 In a response that contains a mix of symmetric keys and errors, all symmetric keys precede all errors –
 1099 i.e. the **SymkeyResponse** element will not consist of **Symkeys** interspersed with **SymkeyErrors** in
 1100 between; all **Symkeys** (if any) will start from the top of the response till the first **SymkeyError** element.

```

1101 [k01]    <ekmi:SymkeyResponse
1102 [k02]    xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1103 [k03]    xmlns:xenc='http://www.w3.org/2001/04/xmldc#>
1104 [k04]    <ekmi:Symkey>
1105 [k05]        <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1106 [k06]        <ekmi:GlobalKeyID>10514-4-3792</ekmi:GlobalKeyID>
1107 [k07]        <ekmi:KeyUsePolicy>
1108 [k08]            <ekmi:KeyUsePolicyID>10514-9</ekmi:KeyUsePolicyID>
1109 [k09]            <ekmi:PolicyName>DES-EDE Policy for EHR-CDC</ekmi:PolicyName>
1110 [k10]            <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
1111 [k11]            <ekmi:KeyAlgorithm>
1112 [k12]                http://www.w3.org/2001/04/xmldc#tripleDES-cbc
1113 [k13]            </ekmi:KeyAlgorithm>
1114 [k14]            <ekmi:KeySize>192</ekmi:KeySize>
1115 [k15]            <ekmi:Status>Active</ekmi:Status>
1116 [k16]            <ekmi:Permissions>
1117 [k17]                <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
1118 [k18]                <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
1119 [k19]                <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
1120 [k20]                <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
1121 [k21]                <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
1122 [k22]                <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
1123 [k23]                <ekmi:PermittedNumberOfTransactions
1124 [k24]                    ekmi:any="true" xsi:nil="true"/>
1125 [k25]                <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
1126 [k26]                <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
1127 [k27]            </ekmi:Permissions>
1128 [k28]        </ekmi:KeyUsePolicy>
1129 [k29]        <ekmi:EncryptionMethod
1130 [k30]            Algorithm="http://www.w3.org/2001/04/xmldc#rsa-1_5"/>
1131 [k31]        <xenc:CipherData>
1132 [k32]            <xenc:CipherValue>
1133 [k33]                E9zWB/y93hVSzeTLiDcQoDxmLNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJ
1134 [k34]                hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfglpU8tlWtx27STRcR/2fw
1135 [k35]                UlWtx27STRcRJMtaGHtXuLlWtx27STRcRpIsY=
1136 [k36]            </xenc:CipherValue>
1137 [k37]        </xenc:CipherData>
1138 [k38]    </ekmi:Symkey>
1139 [k39]    <ekmi:Symkey>
1140 [k40]        <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1141 [k41]        <ekmi:GlobalKeyID>10514-4-3793</ekmi:GlobalKeyID>
1142 [k42]        <ekmi:KeyUsePolicy>
1143 [k43]            <ekmi:KeyUsePolicyID>10514-12</ekmi:KeyUsePolicyID>
1144 [k44]            <ekmi:PolicyName>DES-EDE Policy for EHR-CRO</ekmi:PolicyName>
1145 [k45]            <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
1146 [k46]            <ekmi:KeyAlgorithm>
1147 [k47]                http://www.w3.org/2001/04/xmldc#tripleDES-cbc
1148 [k48]            </ekmi:KeyAlgorithm>
1149 [k49]            <ekmi:KeySize>192</ekmi:KeySize>
1150 [k50]            <ekmi:Status>Active</ekmi:Status>
1151 [k51]            <ekmi:Permissions>
1152 [k52]                <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
1153 [k53]                <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
1154 [k54]                <ekmi:PermittedDate>
1155 [k55]                    <ekmi:StartDate>2008-01-01</ekmi:StartDate>

```

```

1156 [k56]         <ekmi:EndDate>2009-12-31</ekmi:EndDate>
1157 [k57]         </ekmi:PermittedDate>
1158 [k58]         </ekmi:PermittedDates>
1159 [k59]         <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
1160 [k60]         <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
1161 [k61]         <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
1162 [k62]         <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
1163 [k63]         <ekmi:PermittedNumberOfTransactions
1164 [k64]             ekmi:any="true" xsi:nil="true"/>
1165 [k65]         <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
1166 [k66]         <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
1167 [k67]         </ekmi:Permissions>
1168 [k68]     </ekmi:KeyUsePolicy>
1169 [k69]     <ekmi:EncryptionMethod
1170 [k70]         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
1171 [k71]     <xenc:CipherData>
1172 [k72]         <xenc:CipherValue>
1173 [k73]             qUiQXG0ca8EU871zBoXBjDoDxm1NxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCF
1174 [k74]             hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/
1175 [k75]             UJow/qimxi8+ huUYJMtaGHtXuLlWtx27STRcRpIsY=
1176 [k76]         </xenc:CipherValue>
1177 [k77]     </xenc:CipherData>
1178 [k78] </ekmi:Symkey>
1179 [k79] <ekmi:Symkey>
1180 [k80]     <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1181 [k81]     <ekmi:GlobalKeyID>10514-4-3795</ekmi:GlobalKeyID>
1182     ...
1183 [k82]         <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
1184     ...
1185 [k83] </ekmi:Symkey>
1186 [k84] <ekmi:Symkey>
1187 [k85]     <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1188 [k86]     <ekmi:GlobalKeyID>10514-4-3797</ekmi:GlobalKeyID>
1189     ...
1190 [k87]         <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
1191     ...
1192 [k88] </ekmi:Symkey>
1193 [k89] <ekmi:Symkey>
1194 [k90]     <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1195 [k91]     <ekmi:GlobalKeyID>10514-4-3798</ekmi:GlobalKeyID>
1196     ...
1197 [k92]         <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
1198     ...
1199 [k93] </ekmi:Symkey>
1200 [k94] <ekmi:Symkey>
1201 [k95]     <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1202 [k96]     <ekmi:GlobalKeyID>10514-4-3799</ekmi:GlobalKeyID>
1203     ...
1204 [k97]         <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
1205     ...
1206 [k98] </ekmi:Symkey>
1207 [k99] <ekmi:Symkey>
1208 [k100]     <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1209 [k101]     <ekmi:GlobalKeyID>10514-4-3801</ekmi:GlobalKeyID>
1210     ...
1211 [k102]         <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
1212     ...
1213 [k103] </ekmi:Symkey>
1214 [k104] <ekmi:SymkeyError>

```

```

1215 [k105]      <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1216 [k106]      <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
1217 [k107]      <ekmi:RequestedKeyClass>EHR-PAT</ekmi:RequestedKeyClass>
1218 [k108]      <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
1219 [k109]      <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
1220 [k110]      </ekmi:SymkeyError>
1221 [k111]      <ekmi:SymkeyError>
1222 [k112]      <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1223 [k113]      <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
1224 [k114]      <ekmi:RequestedKeyClass>EHR-PHY</ekmi:RequestedKeyClass>
1225 [k115]      <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
1226 [k116]      <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
1227 [k117]      </ekmi:SymkeyError>
1228 [k118]      </ekmi:SymkeyResponse>

```

1229 [k01] – [k103] is no different from the response shown in **Section 3.8 Response with multiple new**
1230 **symmetric keys**.

1231 [k104] - [k110] identifies the first **SymkeyError** returned by the SKS server. It is not unlike the error
1232 described in **Section 3.9 Response with an SKS error**.

1233 [k111] - [k117] identifies the second **SymkeyError** returned by the SKS server.

1234 [k118] is the closing tag of the **SymkeyResponse** tag.

1235 3.12 Response with a pending Request ID

1236 Some use-cases call for sending a request to the SKS server and getting a response, asynchronously.
1237 In these situations, SKSML allows for returning a response with a **SymkeyRequestID** in it. This
1238 indicates that the request is being processed and its status is currently pending. The client application
1239 may use this request identifier to poll the SKS server for an update on the request status.

1240 The format of the **SymkeyResponse** is as follows.

```

1241 [l01]      <ekmi:SymkeyResponse
1242 [l02]      xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1243 [l03]      xmlns:xenc='http://www.w3.org/2001/04/xmldsig#'>
1244 [l04]      <ekmi:SymkeyWorkInProgress>
1245 [l05]      <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
1246 [l06]      <ekmi:SymkeyRequestID>10514-4-7235</ekmi:SymkeyRequestID>
1247 [l07]      </ekmi:SymkeyWorkInProgress>
1248 [l08]      </ekmi:SymkeyResponse>

```

1249 [l01] – [l03] is the standard preamble to a **SymkeyResponse** element.

1250 [l04] identifies the start of the second **SymkeyWorkInProgress** element. This indicates that the server
1251 was unable to return a response at this time and as a result, has provided a request identifier that the
1252 client may use to query the server at a later time for a response.

1253 [l05] contains the **RequestedGlobalKeyID** element. This element contains either a request for a new
1254 symmetric key, or a request for an existing symmetric key. In this example, the request is for a new
1255 symmetric key.

1256 [l06] contains the **SymkeyRequestID** element. This element contains the unique request identifier
1257 provided by the SKS server for the request it received. The **SymkeyRequestID** appears identical to the
1258 **GlobalKeyID**; however, their meanings are very distinct.

1259 [l07] contains the closing tag of the **SymkeyWorkInProgress** element.

1260 [l08] is the closing tag of the *SymkeyResponse* element.

1261 3.13 Request for an update of a pending Request ID

1262 Some use-cases call for sending a request and getting a response, to and from the SKS server,
1263 asynchronously. In these situations, SKSML allows for returning a response with a **SymkeyRequestID**
1264 in it that indicates that the status of the request is currently pending. The client application may use this
1265 request identifier to poll the SKS server for an update on the request status.

1266 When a client makes a *SymkeyRequest* with a *SymkeyRequestID* in it, the SKS server may send back
1267 one of three responses: i) a *SymkeyWorkInProgress* with the same *SymkeyRequestID* in it, indicating
1268 that the request is still pending; ii) a *SymkeyError*, indicating that the request was processed, but the
1269 processing resulted in an error; and iii) a *Symkey* with a symmetric key in it, indicating a successful
1270 response.

1271 The format of the **SymkeyResponse** is as follows.

```
1272 [m01]      <ekmi:SymkeyRequest  
1273 [m02]          xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'>  
1274 [m03]          <ekmi:SymkeyRequestID>10514-4-7235</ekmi:SymkeyRequestID>  
1275 [m04]      </ekmi:SymkeyResponse>
```

1276 [m01] – [m02] is the standard preamble to a **SymkeyRequest** element.

1277 [m03] contains the **SymkeyRequestID** element. This element contains the unique request identifier
1278 provided by the SKS server when the client made the request for a symmetric key earlier. The
1279 *SymkeyRequestID* appears identical to the *GlobalKeyID*; however, their meanings are very distinct when
1280 they are wrapped in the appropriate element tags.

1281 [m04] contains the closing tag of the *SymkeyRequest* element.

1282 3.14 Request for a symmetric key-caching policy

1283 When a client application (that has been linked to the SKCL) needs to encrypt sensitive data, it will call
1284 an API method within the SKCL for a new symmetric key. After the SKCL has ensured that the
1285 application is authorized to make such a request (by verifying that the configured/passed-in credentials
1286 can access the cryptographic key-store module on the client containing the PrivateKey used for signing
1287 SKSML requests), the SKCL assembles the following SKSML request:

```
1288 [n01] <ekmi:KeyCachePolicyRequest  
1289 [n02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01"/>
```

1290 [n01] is the start of the **KeyCachePolicyRequest** element.

1291 [n02] identifies the namespace to which this XML conforms, and the location of its XML Schema
1292 Definition (XSD).

1293 The **KeyCachePolicyRequest** is an “empty” request. It does not need to specify any requesting
1294 parameter or element, since the SKS server only needs to know who requested the policy. The server
1295 derives this information from the SOAP Header and consequently has everything it needs to know – the
1296 digital signature to establish the identity of the requester, as well as to ensure message integrity in the
1297 request.

1298 While the **KeyCachePolicyRequest** element is very simple, the Web Service Security (WSS) envelope
1299 – which provides security for all SKSML messages – expands the size of the message. The same
1300 request shown above, is displayed below in its entirety, with its WSS envelope. Please note that some
1301 content – such as Base64-encoded binary content - has been reformatted for aesthetics and clarity of

the XML elements. The actual elements and data-types have been preserved from actual SKSML messages.

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

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

```

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" SOAP-ENV:mustUnderstand="1">
      <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary"
        ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509v3" wsu:Id="XWSSGID-1172790302111-1738806553">
        MIIDfDCCAmSgAwIBAgIIAe/AvliGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdu3Ryb25nab
S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvcjBTdHJvbmdLZXkgREVNTyBVc2Ugl
T25seTEuXzEueChMOU3Ryb25nQXV0aCBjbmlwHhcNMDYwNzI1MTcxMDMwWWhcNMDcwNzIa64dd3k
A1UECxMbm9yIFN0cm9uZ0t0eSBERU1PIFVzZSBPbm5MRcwFQYDVQKEw5TdHJvbmdBdXRoIEl2da
S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvcjBTdHJvbmdLZXkgREVNTyBVc2Ugia
T25seTEuXzEueChMOU3Ryb25nQXV0aCBjbmlwHhcNMDYwNzI1MTY0NjEwWWhcNMDcwNzI1s34wdd
NjEwWjBpMREwDwYKZiMiZPyLQBARMB0TEVMBMGA1UEAxMMU0tTIFNlcnZlci0xMSQwIgYDVQsdw2
ExtGb3IgU3Ryb25nS2V5IERFTU8gVXNlIE9ubHkxFzAVBgNVBAoTd0N0cm9uZ0F1dGggSW5jMIIBD2
NBgkqhkiG9w0BAQEFAA0CAQ8AMIIBCGKCAQEAztpqRoU5A8plxx1Rz1QEUnlAAM1D5g9+isIr3wxa
hbwtFSMYilnY4iV77xU/nsM0nMZ7RxsLYKdCzQ10DVYqQwqmAvaJ5Z6SVy34gZ51YG+rSWE3NjFsd
b0XW8RJYA/Tn6Lmht/qngrcaqmtP0cAAiMRZOWtCTmC2K/LEqDabXSyU6Hh8ySNE3njbvmWpresf
zsYokTdvNwQqT6tK010wJsdJ1+hxM7DnMLvMNq5reINfsKhDdX17wzhrBUx+hiYA/qo8tMXkL6wsd
4PN5dYugtZpSzIdU05tIg58Avhzw07hy5oofB1KFY22CeljQ36u0bMjuyGj6UYHs3rddfdfsds32rda
YzCBnzANBkgqhkiG9w0BAQEFAA0BjQAwgYkCgYEAyAmxMZhYA8wHJ4UE4b61s51JVWe4Fygj4Mcf3a
hvcNAQELBQADggEBAK05PtvZD4WPgl0e=
      </wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
            Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <InclusiveNamespaces
            xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"
            PrefixList="wsse SOAP-ENV"/>
          </ds:CanonicalizationMethod>
          <ds:SignatureMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          <ds:Reference URI="#XWSSGID-1172790300636-653454040">
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>lu4m+rp4oebgl9g+t3nRaZYqUle=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#XWSSGID-1172790300637708871805">
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>WcP0mTCbffcEHXhGf5rLEYWlrZg=</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>
svStAvBRRrF+g2biPl7uWHkJTQPil8t4phMb0ZQsZlQcn36tcMSj/a4+4LPnF0B3Y8y02lr10a1
fGqCPAWZnuEH34VQEM196rRwV258mgp8uwpXEYJIGpJqg89w8+/Nda0DccLQ2Biz7QM/HSM2ab
ogNJwqmbSyIaz0sn0cU=
        </ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference xmlns:wsu="http://docs.oasis-

```

```

1359 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
1360     wsu:Id="XWSSGID-1172790300633-442423344">
1361     <wsse:Reference URI="#XWSSGID-1172790302111-1738806553"
1362         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1363 wss-x509-token-profile-1.0#X509v3"/>
1364     </wsse:SecurityTokenReference>
1365     </ds:KeyInfo>
1366     </ds:Signature>
1367     <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1368 wss-wssecurity-utility-1.0.xsd" wsu:Id="XWSSGID-1172790300637708871805">
1369         <wsu:Created>2007-03-01T23:05:00Z</wsu:Created>
1370         <wsu:Expires>2007-03-01T23:05:05Z</wsu:Expires>
1371     </wsu:Timestamp>
1372 </wsse:Security>
1373 </SOAP-ENV:Header>
1374 <SOAP-ENV:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
1375 wssecurity-utility-1.0.xsd" wsu:Id="XWSSGID-1172790300636-653454040">
1376     <ekmi:KeyCachePolicyRequest xmlns:ekmi="http://docs.oasis-
1377 open.org/ekmi/2008/01"/>
1378 </SOAP-ENV:Body>
1379 </SOAP-ENV:Envelope>

```

1380 3.15 Response with a symmetric key-caching policy (1)

1381 After an SKS server has performed its operations of authenticating a **KeyCachePolicyRequest**,
1382 identifying the requester, determining policies that apply to the requester, it assembles the following
1383 response and returns it to the client. (The SOAP message, as indicated earlier, is secured using WSS,
1384 but only the actual SKSM content is displayed and discussed here).

```

1385 [o01] <ekmi:KeyCachePolicyResponse
1386 [o02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1387 [o03]     xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
1388 [o04]     <ekmi:KeyCachePolicy>
1389 [o05]         <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
1390 [o06]         <ekmi:PolicyName>No Caching Policy</ekmi:PolicyName>
1391 [o07]         <ekmi:Description>
1392 [o08]             This policy is for high-risk, always-connected machines on the
1393 [o09]             network, which will never cache symmetric keys locally. This
1394 [o10]             policy never expires (but checks monthly for any updates).
1395 [o11]         </ekmi:Description>
1396 [o12]         <ekmi:KeyClass>NoCachingClass</ekmi:KeyClass>
1397 [o13]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1398 [o14]         <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
1399 [o15]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1400 [o16]         <ekmi:Status>Active</ekmi:Status>
1401 [o17]     </ekmi:KeyCachePolicy>
1402 [o18] </ekmi:KeyCachePolicyResponse>

```

1403 [o01] is the start of the **KeyCachePolicyResponse** element.

1404 [o02] and [o03] identify the namespaces to which this XML conforms, and the location of their XML
1405 Schema Definitions (XSD).

1406 [o04] is the start of the first – and only - **KeyCachePolicy** element.

1407 [o05] identifies the **KeyCachePolicyID** (KCPID) assigned to this policy by the SKS server. In this
1408 example, the concatenated values of the Domain ID and Policy ID indicate that the key belongs to the
1409 organization represented by the PEN, 10514; and is the first key-caching policy within the SKMS.

1410 [o06] provides a descriptive name for this key-cache policy through the **PolicyName** element, which is
 1411 helpful to human readers when identifying this policy.

1412 [o07] is the start tag of the **Description** element.

1413 [o08] - [o10] provides a human-readable description about this key-cache policy.

1414 [o11] is the closing tag of the **Description** element.

1415 [o12] specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are
 1416 subject to this caching policy.

1417 [o13] specifies the date and time that this **KeyCachePolicy** is effective. This is accomplished through a
 1418 **StartDate** element. In this example, the policy is effective as of January 01, 2008.

1419 [o14] specifies the date and time that this **KeyCachePolicy** becomes invalid. This is accomplished
 1420 through a **EndDate** element. In this example, the use of the UNIX "epoch" date (January 01, 1969)
 1421 indicates that this policy never expires.

1422 [o15] specifies the frequency at which this client must check with the SKS server for updates to the key-
 1423 caching policy. This is specified in seconds in the **PolicyCheckInterval** element; in this example it is a
 1424 monthly interval.

1425 [o16] indicates the **Status** of this **KeyCachePolicy** and whether it is an active policy or not.

1426 [o17] is the closing tag of the **KeyCachePolicy** element.

1427 [o18] is the closing tag of the **KeyCachePolicyResponse** element.

1428 3.16 Response with a symmetric key-caching policy (2)

1429 This is a second example of a key-caching policy response that has additional elements in the policy
 1430 permitting caching and specify the number of unused and used (for encryption) symmetric keys that may
 1431 be cached by the client machine.

```

1432 [p01] <ekmi:KeyCachePolicyResponse
1433 [p02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1434 [p03]     xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
1435 [p04]     <ekmi:KeyCachePolicy>
1436 [p05]         <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
1437 [p06]         <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
1438 [p07]         <ekmi:Description>
1439 [p08]             This policy defines how company-issued laptops will manage symmetric
1440 [p09]             keys used for file/disk encryption in their local cache. This
1441 [p10]             policy must be used by all laptops that use the company EKMI.
1442 [p11]         </ekmi:Description>
1443 [p12]         <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
1444 [p13]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1445 [p14]         <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
1446 [p15]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1447 [p16]         <ekmi>Status>Active</ekmi>Status>
1448 [p17]         <ekmi:NewKeysCacheDetail>
1449 [p18]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1450 [p19]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1451 [p20]         </ekmi:NewKeysCacheDetail>
1452 [p21]         <ekmi:UsedKeysCacheDetail>
1453 [p22]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1454 [p23]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1455 [p24]         </ekmi:UsedKeysCacheDetail>
  
```

1456 [p25] </ekmi:KeyCachePolicy>
 1457 [p26] </ekmi:KeyCachePolicyResponse>

1458 [p01] is the start of the **KeyCachePolicyResponse** element.

1459 [p02] and [p03] identify the namespaces to which this XML conforms, and the location of their XML
 1460 Schema Definitions (XSD).

1461 [p04] is the start of the first – and only - **KeyCachePolicy** element.

1462 [p05] identifies the **KeyCachePolicyID** (KCPID) assigned by the SKS server for the key-caching policy
 1463 being returned.

1464 [p06] provides a descriptive name for this key-cache policy through the **PolicyName** element, which is
 1465 helpful to human readers when identifying this policy.

1466 [p07] is the start tag of the **Description** element.

1467 [p08] - [p10] provides a human-readable description about this key-cache policy.

1468 [p11] is the closing tag of the **Description** element.

1469 [p12] specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are
 1470 subject to this caching policy.

1471 [p13] specifies the date and time that this **KeyCachePolicy** is effective. This is accomplished through a
 1472 **StartDate** element. In this example, the policy is effective as of January 01, 2008.

1473 [p14] specifies the date and time that this **KeyCachePolicy** becomes invalid. This is accomplished
 1474 through a **EndDate** element. In this example, the policy expires on December 31, 2008.

1475 [p15] specifies the frequency at which this client must check with the SKS server for updates to the key-
 1476 caching policy. This is specified in seconds in the **PolicyCheckInterval** element; in this example it is a
 1477 monthly interval.

1478 [p16] indicates the **Status** of this **KeyCachePolicy** and whether it is an active policy or not.

1479 [p17] is the start of the **NewKeysCacheDetail** element, which provides details about how many new
 1480 symmetric keys – that haven't been used for any encryption transactions – may be cached by the client
 1481 and for how long.

1482 [p18] indicates the maximum number of new (unused) symmetric keys that may be cached by the client.
 1483 This is specified through the **MaximumKeys** element. When the client uses a symmetric key, this
 1484 reduces the number of new symmetric keys. In this case, the SKCL connects to the SKS server (if it is
 1485 on the network) and requests a new symmetric key to add to its new-key cache.

1486 [p19] indicates the maximum duration that new (unused) symmetric keys may be cached by the client.
 1487 This is specified through the **MaximumDuration** element in seconds. If there are any new keys that
 1488 exceed this duration limit, the SKCL deletes the specific symmetric key and replaces it with a new
 1489 symmetric key from the SKS server.

1490 [p20] is the closing tag of the **NewKeysCacheDetail** element.

1491 [p21] is the start of the **UsedKeysCacheDetail** element, which provides details about how many used
 1492 symmetric keys – those that HAVE been used for any encryption transactions – may be cached by the
 1493 client and for how long.

1494 [p22] indicates the maximum number of used symmetric keys that may be cached by the client through
 1495 the **MaximumKeys** element. If the client already has the maximum number of used-keys in its cache,

1496 using the First-In-First-Out (FIFO) method, it deletes the oldest symmetric key in the cache to replace
 1497 with the key that transitioned from the “new” to “used” status.

1498 [p23] indicates the maximum duration that used symmetric keys may be cached by the client through the
 1499 **MaximumDuration** element in seconds. If there are any used keys that exceed this duration limit, the
 1500 SKCL deletes the specific symmetric key. While this may temporarily reduce the number of used
 1501 symmetric keys in the cache, the SKCL takes the most conservative position when making this decision.

1502 [p24] is the closing tag of the **UsedKeysCacheDetail** element.

1503 [p25] is the closing tag of the **KeyCachePolicy** element.

1504 [p26] is the closing tag of the **KeyCachePolicyResponse** element.

1505 3.17 Response with multiple symmetric key-caching policies (3)

1506 This is a third example of a key-caching policy response that has multiple key-cache policies that apply
 1507 to different classes of symmetric keys.

```

1508 [q01] <ekmi:KeyCachePolicyResponse
1509 [q02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1510 [q03]     xmlns:xenc='http://www.w3.org/2001/04/xmenc#'>
1511 [q04]     <ekmi:KeyCachePolicy>
1512 [q05]         <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
1513 [q06]         <ekmi:PolicyName>No Caching Policy</ekmi:PolicyName>
1514 [q07]         <ekmi:Description>
1515 [q08]             This policy is for high-risk, always-connected machines on the
1516 [q09]             network, which will never cache symmetric keys locally. This
1517 [q10]             policy never expires (but checks monthly for any updates).
1518 [q11]         </ekmi:Description>
1519 [q12]         <ekmi:KeyClass>NoCachingClass</ekmi:KeyClass>
1520 [q13]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1521 [q14]         <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
1522 [q15]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1523 [q16]         <ekmi>Status>Active</ekmi>Status>
1524 [q17]     </ekmi:KeyCachePolicy>
1525 [q18]     <ekmi:KeyCachePolicy>
1526 [q19]         <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
1527 [q20]         <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
1528 [q21]         <ekmi:Description>
1529 [q22]             This policy defines how company-issued laptops will manage symmetric
1530 [q23]             keys used for file/disk encryption in their local cache. This
1531 [q24]             policy must be used by all laptops that use the company EKMI.
1532 [q25]         </ekmi:Description>
1533 [q26]         <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
1534 [q27]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1535 [q28]         <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
1536 [q29]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1537 [q30]         <ekmi>Status>Active</ekmi>Status>
1538 [q31]         <ekmi:NewKeysCacheDetail>
1539 [q32]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1540 [q33]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1541 [q34]         </ekmi:NewKeysCacheDetail>
1542 [q35]         <ekmi:UsedKeysCacheDetail>
1543 [q36]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1544 [q37]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1545 [q38]         </ekmi:UsedKeysCacheDetail>
1546 [q39]     </ekmi:KeyCachePolicy>
1547 [q40] </ekmi:KeyCachePolicy>
  
```

```

1548 [q41]      <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
1549 [q42]      <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
1550 [q43]      <ekmi:Description>
1551 [q44]          This policy defines how company-issued laptops will manage
1552 [q45]          symmetric keys used for file/disk encryption in their local
1553 [q46]          cache. This policy must be used by all laptops.
1554 [q47]      </ekmi:Description>
1555 [q48]      <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
1556 [q49]      <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1557 [q50]      <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
1558 [q51]      <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1559 [q52]      <ekmi:Status>Active</ekmi:Status>
1560 [q53]      <ekmi:NewKeysCacheDetail>
1561 [q54]          <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1562 [q55]          <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1563 [q56]      </ekmi:NewKeysCacheDetail>
1564 [q57]      <ekmi:UsedKeysCacheDetail>
1565 [q58]          <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1566 [q59]          <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1567 [q60]      </ekmi:UsedKeysCacheDetail>
1568 [q61]      <ekmi:KeyCachePolicy>
1569 [q62] </ekmi:KeyCachePolicyResponse>

```

1570

1571 [q01] is the start of the **KeyCachePolicyResponse** element.

1572 [q02] and [q03] identify the namespaces to which this XML conforms, and the location of their XML
 1573 Schema Definitions (XSD).

1574 [q04] is the start of the first of three **KeyCachePolicy** elements in this response.

1575 [q05] identifies the **KeyCachePolicyID** (KCPID) assigned to this policy by the SKS server. In this
 1576 example, the concatenated values of the Domain ID and Policy ID indicate that the key belongs to the
 1577 organization represented by the PEN, 10514; and is the first key-caching policy within the SKMS.

1578 [q06] provides a descriptive name for this key-cache policy through the **PolicyName** element, which is
 1579 helpful to human readers when identifying this policy.

1580 [q07] is the start tag of the **Description** element.

1581 [q08] - [q10] provides a human-readable description about this key-cache policy.

1582 [q11] is the closing tag of the **Description** element.

1583 [q12] specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are
 1584 subject to this caching policy.

1585 [q13] specifies the date and time that this **KeyCachePolicy** is effective. This is accomplished through a
 1586 **StartDate** element. In this example, the policy is effective as of January 01, 2008.

1587 [q14] specifies the date and time that this **KeyCachePolicy** becomes invalid. This is accomplished
 1588 through a **EndDate** element. In this example, the use of the UNIX "epoch" date (January 01, 1969)
 1589 indicates that this policy never expires.

1590 [q15] specifies the frequency at which this client must check with the SKS server for updates to the key-
 1591 caching policy. This is specified in seconds in the **PolicyCheckInterval** element; in this example it is a
 1592 monthly interval.

1593 [q16] indicates the **Status** of this **KeyCachePolicy** and whether it is an active policy or not.

1594 [q17] is the closing tag of the first **KeyCachePolicy** element.

1595 [q18] is the start of the second of three **KeyCachePolicy** elements in this response.

1596 [q19] identifies the **KeyCachePolicyID** (KCPID) assigned by the SKS server for the key-caching policy
 1597 being returned.

1598 [q20] provides the descriptive name for this key-cache policy through the **PolicyName** element.

1599 [q21] is the start tag of the **Description** element.

1600 [q22] - [q24] provides a human-readable description about this key-cache policy.

1601 [q25] is the closing tag of the **Description** element.

1602 [q26] specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are
 1603 subject to this caching policy.

1604 [q27] specifies the date and time that this **KeyCachePolicy** is effective. This is accomplished through a
 1605 **StartDate** element. In this example, the policy is effective as of January 01, 2008.

1606 [q28] specifies the date and time that this **KeyCachePolicy** becomes invalid. This is accomplished
 1607 through a **EndDate** element. In this example, the policy expires on December 31, 2008.

1608 [q29] specifies the frequency at which this client must check with the SKS server for updates to the key-
 1609 caching policy. This is specified in seconds in the **PolicyCheckInterval** element; in this example it is a
 1610 monthly interval.

1611 [q30] indicates the **Status** of this **KeyCachePolicy** and whether it is an active policy or not.

1612 [q31] is the start of the **NewKeysCacheDetail** element, which provides details about how many new
 1613 symmetric keys – that haven't been used for any encryption transactions – may be cached by the client
 1614 and for how long.

1615 [q32] indicates the maximum number of new (unused) symmetric keys that may be cached by the client.
 1616 This is specified through the **MaximumKeys** element. When the client uses a symmetric key, this
 1617 reduces the number of new symmetric keys. In this case, the SKCL connects to the SKS server (if it is
 1618 on the network) and requests a new symmetric key to add to its new-key cache.

1619 [q33] indicates the maximum duration that new (unused) symmetric keys may be cached by the client.
 1620 This is specified through the **MaximumDuration** element in seconds. If there are any new keys that
 1621 exceed this duration limit, the SKCL deletes the specific symmetric key and replaces it with a new
 1622 symmetric key from the SKS server.

1623 [q34] is the closing tag of the **NewKeysCacheDetail** element.

1624 [q35] is the start of the **UsedKeysCacheDetail** element, which provides details about how many used
 1625 symmetric keys – those that HAVE been used for any encryption transactions – may be cached by the
 1626 client and for how long.

1627 [q36] indicates the maximum number of used symmetric keys that may be cached by the client through
 1628 the **MaximumKeys** element. If the client already has the maximum number of used-keys in its cache,
 1629 using the First-In-First-Out (FIFO) method, it deletes the oldest symmetric key in the cache to replace
 1630 with the key that transitioned from the “new” to “used” status.

1631 [q37] indicates the maximum duration that used symmetric keys may be cached by the client through the
 1632 **MaximumDuration** element in seconds. If there are any used keys that exceed this duration limit, the
 1633 SKCL deletes the specific symmetric key. While this may temporarily reduce the number of used
 1634 symmetric keys in the cache, the SKCL takes the most conservative position when making this decision.

1635 [q38] is the closing tag of the **UsedKeysCacheDetail** element.

1636 [q39] is the closing tag of the second **KeyCachePolicy** element.

1637 [q40] is the start of the third **KeyCachePolicy** element in this response.

1638 [q41] identifies the **KeyCachePolicyID** (KCPID) assigned by the SKS server for the key-caching policy
1639 being returned.

1640 [q42] provides the descriptive name for this key-cache policy through the **PolicyName** element.

1641 [q43] is the start tag of the **Description** element.

1642 [q44] - [q46] provides a human-readable description about this key-cache policy.

1643 [q47] is the closing tag of the **Description** element.

1644 [q48] specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are
1645 subject to this caching policy.

1646 [q49] specifies the date and time that this **KeyCachePolicy** is effective.

1647 [q50] specifies the date and time that this **KeyCachePolicy** becomes invalid.

1648 [q51] specifies the frequency at which this client must check with the SKS server for updates to the key-
1649 caching policy.

1650 [q52] indicates the **Status** of this **KeyCachePolicy** and whether it is an active policy or not.

1651 [q53] is the start of the **NewKeysCacheDetail** element, which provides details about how many new
1652 symmetric keys may be cached by the client and for how long.

1653 [q54] indicates the maximum number of new (unused) symmetric keys that may be cached by the client.
1654 This is specified through the **MaximumKeys** element. When the client uses a symmetric key, this
1655 reduces the number of new symmetric keys. In this case, the SKCL connects to the SKS server (if it is
1656 on the network) and requests a new symmetric key to add to its new-key cache.

1657 [q55] indicates the maximum duration that new (unused) symmetric keys may be cached by the client.
1658 This is specified through the **MaximumDuration** element in seconds. If there are any new keys that
1659 exceed this duration limit, the SKCL deletes the specific symmetric key and replaces it with a new
1660 symmetric key from the SKS server.

1661 [q56] is the closing tag of the **NewKeysCacheDetail** element.

1662 [q57] is the start of the **UsedKeysCacheDetail** element, which provides details about how many used
1663 symmetric keys – those that HAVE been used for any encryption transactions – may be cached by the
1664 client and for how long.

1665 [q58] indicates the maximum number of used symmetric keys that may be cached by the client through
1666 the **MaximumKeys** element. If the client already has the maximum number of used-keys in its cache,
1667 using the First-In-First-Out (FIFO) method, it deletes the oldest symmetric key in the cache to replace
1668 with the key that transitioned from the “new” to “used” status.

1669 [q59] indicates the maximum duration that used symmetric keys may be cached by the client through the
1670 **MaximumDuration** element in seconds. If there are any used keys that exceed this duration limit, the
1671 SKCL deletes the specific symmetric key. While this may temporarily reduce the number of used
1672 symmetric keys in the cache, the SKCL takes the most conservative position when making this decision.

1673 [q60] is the closing tag of the **UsedKeysCacheDetail** element.

1674 [q61] is the closing tag of the third and final **KeyCachePolicy** element in this response.

1675 **[q62]** is the closing tag of the ***KeyCachePolicyResponse*** element.

4 Specification

4.1 Element <SymkeyRequest>

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

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

Schema Definition:

```
<xsd:element name="SymkeyRequest">
  <xsd:complexType>
    <xsd:choice>
      <xsd:sequence>
        <xsd:element
          name="GlobalKeyID"
          type="ekmi:GlobalKeyIDType"
          minOccurs="1"
          maxOccurs="unbounded">
        </xsd:element>
        <xsd:element
          name="KeyClasses"
          type="ekmi:KeyClassesType"
          minOccurs="0"
          maxOccurs="unbounded">
        </xsd:element>
        <xsd:element
          name="X509EncryptionCertificate"
          type="ekmi:X509CertificateType"
          minOccurs="0"
          maxOccurs="1">
        </xsd:element>
      </xsd:sequence>
      <xsd:sequence>
        <xsd:element
          name="SymkeyRequestID"
          type="ekmi:SymkeyRequestIDType"
          minOccurs="1"
          maxOccurs="unbounded">
        </xsd:element>
      </xsd:sequence>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

1723 The <SymkeyRequest> element consists of a choice of two sequences of elements; one or the other
1724 sequence MUST be present in a <SymkeyRequest>.

1725 1. <GlobalKeyID> [Required]

1726
1727 The first sequence choice consists of the <GlobalKeyID> element of type **GlobalKeyIDType**. It
1728 identifies the unique global key identifier of the requested symmetric key within the target
1729 Symmetric Key Management System (SKMS). If this sequence is chosen, there MUST be at
1730 least one <GlobalKeyID> element in a <SymkeyRequest>, but there may be an unbounded
1731 (unlimited) number of <GlobalKeyID> elements specified.

1732
1733 The <GlobalKeyID> element and **GlobalKeyIDType** is specified in Section 4.2.

1734 <KeyClasses> [Optional]

1735
1736 This element of type **KeyClassesType**, when specified, identifies at least one <KeyClass>
1737 element, but may specify an unbounded (unlimited) number of <KeyClass> elements within the
1738 <KeyClasses> set. Client applications may request one or more symmetric keys conforming to
1739 one or more key classes required by the application. If the client application is authorized to
1740 receive keys conforming to such key classes, the **SKS** server will generate and supply them.

1741
1742 When more than one <GlobalKeyID> for a new symmetric key is specified in the request, there
1743 MAY be only one <KeyClass> element within the <KeyClasses> set.

1744
1745 When the client requires more than one new symmetric key, and each key is required to be of a
1746 different key class, there MUST be only one <GlobalKeyID> element followed by as many
1747 <KeyClass> elements inside the <KeyClasses> set, as needed by the client application.

1748
1749 When a client requires multiple symmetric keys of two or more key classes, the client MUST
1750 send multiple requests to the **SKS** server. See examples 4 and 5 below in this section.

1751
1752 The <KeyClasses> and <KeyClass> elements and their respective types are specified in
1753 Section 4.3.

1754 <X509EncryptionCertificate> [Optional]

1755
1756 This element of type **X509EncryptionCertificateType**, when specified, identifies a PKI X509-
1757 compliant digital certificate whose corresponding private-key is owned/authorized for use by the
1758 requesting client. The <X509EncryptionCertificate> MUST meet the following requirements:

- 1759 a) It MUST be a valid X509 v3 digital certificate whose expiry is not earlier than the date
1760 and time the symmetric-key request is received by the **SKS** server;
- 1761 b) It MUST not be revoked by the Certificate Authority (CA) who issued the encryption
1762 certificate at the date and time the symmetric-key request is received by the **SKS** server;
- 1763 c) It MUST have its *keyEncipherment* bit set in the *keyUsage* extension of the
1764 certificate.

1765 Note: The **SKS** server will use the specified **X509EncryptionCertificate** if the security policy on
1766 the **SKS** server permits it. The security policy may have specified that only encryption
1767 certificates stored on the **SKS** server database or in an LDAP Directory known to the server be
1768 used; in which case the SKS server will ignore the encryption certificate in the
1769 **X509EncryptionCertificate** element and use what is stored on the **SKS** server or other location
1770 known to the server. In the event the **SKS** server uses an encryption certificate stored on the
1771 server-side, it is assumed that the requesting client has the corresponding private-key to decrypt
1772 the payload when extracted from the response.

1773 The <X509EncryptionCertificate> and its respective type is specified in Section 4.4.
 1774

1775 2. <SymkeyRequestID> [Required]
 1776
 1777 The second sequence choice consists of the <SymkeyRequestID> element of type
 1778 **SymkeyRequestIDType**. It identifies the unique symmetric-key request identifier within the target
 1779 Symmetric Key Management System (SKMS). If this sequence is chosen, there MUST be at
 1780 least one <SymkeyRequestID> element in a <SymkeyRequest>, but there may be an unbounded
 1781 (unlimited) number of <SymkeyRequestID> elements specified.
 1782
 1783 The presence of the <SymkeyRequestID> element in the <SymkeyRequest> implies that the
 1784 client had previously made a <SymkeyRequest> asynchronously, and instead of receiving a
 1785 <Symkey> or a <SymkeyError>, it received a <SymkeyWorkInProgress> response with a
 1786 <SymkeyRequestID> in it. The client is now following-up with the SKS Server to get an update
 1787 on its earlier request with the <SymkeyRequestID>.
 1788
 1789 The <SymkeyRequestID> element and the **SymkeyRequestIDType** is specified in Section 4.5.

1790 Some examples of the <SymkeyRequest> element are as follows:

1791 **Example 1 – A single new symmetric key request of a default key class:**

```

1792 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1793   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
1794 </ekmi:SymkeyRequest>
  
```

1795 **Example 2 – A request for three new symmetric keys of a default key class for each symmetric**
 1796 **key:**

```

1797 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1798   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
1799   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
1800   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
1801 </ekmi:SymkeyRequest>
  
```

1802 **Example 3 – A request for a single new symmetric key of a specific key class:**

```

1803 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1804   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
1805   <ekmi:KeyClasses>
1806     <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
1807   </ekmi:KeyClasses>
1808 </ekmi:SymkeyRequest>
  
```

1809 **Example 4 – A request for a two new symmetric keys with the same key class for each symmetric**
 1810 **key:**

```

1811 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1812   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
1813   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
1814   <ekmi:KeyClasses>
1815     <ekmi:KeyClass>FIN-FX</ekmi:KeyClass>
1816   </ekmi:KeyClasses>
1817 </ekmi:SymkeyRequest>
  
```

1818 **Example 5 – A request for a nine new symmetric keys of different key classes:**

```

1819 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1820   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
  
```

```

1821     <ekmi:KeyClasses>
1822         <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
1823         <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
1824         <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
1825         <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
1826         <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
1827         <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
1828         <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
1829         <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
1830         <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>
1831     </ekmi:KeyClasses>
1832 </ekmi:SymkeyRequest>

```

1833 **Example 6 – A follow-up request to a previously received symmetric-key request ID:**

```

1834     <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1835         <ekmi:SymkeyRequestID>10514-1-1234</ekmi:SymkeyRequestID>
1836     </ekmi:SymkeyRequest>

```

1837 **Example 7 – A follow-up request to previously received multiple symmetric-key request IDs:**

```

1838     <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1839         <ekmi:SymkeyRequestID>10514-1-1234</ekmi:SymkeyRequestID>
1840         <ekmi:SymkeyRequestID>10514-1-1345</ekmi:SymkeyRequestID>
1841         <ekmi:SymkeyRequestID>10514-1-1347</ekmi:SymkeyRequestID>
1842     </ekmi:SymkeyRequest>

```

1843 **Example 8 – A new symmetric key request of a default key class with the**
1844 **X509EncryptionCertificate sent by the client to the server:**

```

1845     <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1846         <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
1847         <ekmi:X509EncryptionCertificate>
1848 MIIDfDCCAmSgAwIBAgIIAe/AvLiGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdU3Ryb25n
1849 S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvciBTdHJvbmdLZXkgREVNTyBVc2Ug
1850 T25seTEwMDU3Ryb25nQXV0aCBjbWwHhcNMDYwNzI1MTcxMDMwWWhcNMDcwNzI1MTcy
1851 MDMwWjBtMREwDwYKZCI1ZPylG0BARMbmjEZMBcGA1UEAxMQUE9TIFJlZ2lzdGVyIDIyMjE1MTcy
1852 A1UECxMbRm9yIFN0cm9uZ0t0eSBBERU1PIFVzZSBPbm95MRcwFQYDVQKEw5TdHJvbmdBdXR0IElu
1853 YzCBnzANBgkqhkiG9w0BAQEFAA0BJQAwgYkCgYEAyAmxMZhY8wHJ4UE4b61s51JVVe4Fygj4MCf
1854 U7LA3JhpUS4TLX0XFwqrcmltL0iVG7YBfarJFluBFJW2X6q8FuvUpv4V9nJrgiWAPtkiRyIx96n
1855 qKXIkkUlQ4idlEglAZI9dEdf4Y5cqBBcygPYNBoTudglM7R47AjR4nr4ks8CAwEAAaOBqTCBpjAO
1856 BgNVHQ8BAf8EBAMCBLAwHQYDVRR00BBYEF0I0rWrZo0LdBRLVncRAwLBqVZpCMB8GA1UdIwQYMBaA
1857 FPTYwEH0JG4iFVHRnt2EWxGluAQVMBGGA1UdIAQMA8wDQYLKwYEAAdISg30BBAEW0gYDVROfBDMw
1858 MTAvoC2gK4YpaHR0cDovL2Rlbw8uc3Ryb25na2V5Lm9yZy9kZW1vLXN1Y1ljYS5jcmwwDQYJKoZI
1859 hvcNAQELBQADggEBAK05PtVZD4WPgl0e+EHUiApzFyCdRzf0pFZtxRwG9lR1PZUWUjmwTNfGFsL
1860 S6kyoHgUfVa5fpT1EU1mXUB/Lmo3hFGyprZjfmD7DwuBcYgmZhV7yHrmGOMIOXjFTACvHpM0vOce
1861 hVx2e4VE0yhBLu/LdH9awGGDp6Bk2XzxqQcs8y6Zz0XZAnPgKQZdjbfKERSsy/d1D8pk5baBk4bd
1862 Zh5680caUrbm9ZReRVTVaY5qiQpk0U+tDrBSj/HIL6GAqegYllkz6KYCy6RV0y6iVVSjHocDqdJr
1863 EVOR+ds6xn8mmodlERrILmuxiLpibPp609SfnDIXNlzLwe5g7ep3lc=
1864         </ekmi:X509EncryptionCertificate>
1865     </ekmi:SymkeyRequest>

```

1866 **Example 9 – A new symmetric key request with a specific key class and the**
1867 **X509EncryptionCertificate:**

```

1868     <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1869         <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
1870         <ekmi:KeyClasses>
1871             <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
1872         </ekmi:KeyClasses>
1873         <ekmi:X509EncryptionCertificate>

```

```

1874 MIIDfDCCAmSgAwIBAgIIAe/AvliGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdu3Ryb25n
1875 S2V5IERFTU8gU3Vib3JkaW5hdGUG0ExJDAiBgNVBAsTG0ZvcjBTdHJvbmddLZXkgREVNTyBvc2Ug
1876 T25seTEwBjBtMREwDwYKZCIiZmZlZGQwYmMwHhcNMDYwNzI1MTcxMDMwHhcNMDcwNzI1MTcy
1877 MDMwWjBtMREwDwYKZCIiZmZlZGQwYmMwHhcNMDYwNzI1MTcxMDMwHhcNMDcwNzI1MTcy
1878 A1UECXMbRm9yIFN0cm9uZ0tleSBERU1PIFVzZSBPbm5MRcwFQYDVQKEw5TdHJvbmddBdXR0IElu
1879 YzCBnzANBgkqhkiG9w0BAQEFAA0BjQAwgYkCgYEAyAmxMZhYA8wHJ4UE4b61s51JVWe4Fygj4MCf
1880 U7LA3JhpUS4TLX0XFWqrcmltL0iVG7YBFarJFJlUBFJW2X6q8FuvUprv4V9nJrgiwAPtkiRyIx96n
1881 qKXIxkUlQ4idlEgIAZI9dEdf4Y5cqBBCygPyNBoTudglM7R47AjR4nr4ks8CAwEAAaOBqTCBpjAO
1882 BgNVHQ8BAf8EBAMCBLAwHQYDVRO0BBYEF0IOwrZo0LdBRLLVncRAwLBqVZpCMB8GA1UdIwQYMBAA
1883 FPTYwEHOJG4iFVHRnt2EWxGluAQVMBGGA1UdIAQRMA8wDQYLLKwYEAAdISg30BBAEwOgYDVROfBDMw
1884 MTAvOC2gK4YpaHR0cDovL2RlbW8uc3Ryb25na2V5Lm9yZy9kZW1vLXN1Yi1jYS5jcmmwDQYJKoZI
1885 hvcNAQELBQADggEBAK05PtvZD4WPgl0e+EHUiaPzFyCdRzf0pFZtxRwG9lR1PZUWUjmwTNfGFsL
1886 S6kyoHgUfVa5fpt1EU1mXUB/Lmo3hFGyprZjfmD7DwuBcYgmZhv7yHrmGOMIOXjFTACvHpM0v0ce
1887 hVx2e4VE0yhBLu/LdH9awGGdp6Bk2XzxqCs8y6Zz0XZAnPgKQZdjbfKERSsy/d1D8pk5baBk4bd
1888 Zh5680caUrbm9ZReRVTVaY5qiQpk0U+tDrBSj/HIL6GAqegYllkz6KYCy6RV0y6iVVSjHocDqdJr
1889 EVOR+ds6xn8mnojdlERRILmxiLpibPp609SfnDIXnlzLwe5g7ep3lc=
1890 </ekmi:X509EncryptionCertificate>
1891 </ekmi:SymkeyRequest>

```

4.2 Element <GlobalKeyID>

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

Schema Definition:

```

<xsd:simpleType name="GlobalKeyIDType">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="5"/>
    <xsd:maxLength value="62"/>
    <xsd:pattern value="[0-9]{1,20}-[0-9]{1,20}-[0-9]{1,20}"/>
    <xsd:whiteSpace value="collapse"/>
  </xsd:restriction>
</xsd:simpleType>

```

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

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

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

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

When an SKMS manages the symmetric keys for a single enterprise, the **DomainID** part of the <GlobalKeyID> element in a <SymkeyRequest> MAY be zero ("0"). When an SKMS manages symmetric

1921 keys for multiple enterprises, the **DomainID** in the <GlobalKeyID> of a <SymkeyRequest> MUST be
 1922 positive and non-zero. In such a situation, the client application will request a symmetric key for the
 1923 domain in which it is authorized to request and receive keys.

1924 The **DomainID** in the <GlobalKeyID> element of a <SymkeyResponse> MUST always be positive and
 1925 non-zero. It will typically contain the PEN of the domain to which the symmetric key belongs.

1926 The **ServerID** part of the <GlobalKeyID> element MUST be a positive integer in the range of 0 (zero) to
 1927 18446744073709551615 (20-byte ASCII decimal).

1928 The **ServerID** part of the <GlobalKeyID> element of a <SymkeyRequest> MUST always be zero ("0").

1929 The **ServerID** part of the <GlobalKeyID> element of a <SymkeyResponse> MUST always be positive and
 1930 non-zero. It will typically contain the unique server identifier of the **SKS** server where the symmetric key
 1931 was generated.

1932 The **KeyID** part of the <GlobalKeyID> element MUST be a positive integer in the range of 0 (zero) to
 1933 18446744073709551615 (20-byte ASCII decimal).

1934 The **KeyID** part of the <GlobalKeyID> element of a <SymkeyRequest> MUST always be zero ("0").

1935 The **KeyID** part of the <GlobalKeyID> element of a <SymkeyResponse> MUST always be positive and
 1936 non-zero. It will typically contain the unique key identifier of the symmetric key within the **SKS** server
 1937 where the key was generated.

1938 **Example 1 – A <GlobalKeyID> value for a new symmetric key from an SKMS that serves a single**
 1939 **domain:**

1940 <ekmi:GlobalKeyID>0-0-0</ekmi:GlobalKeyID>

1941 **Example 2 – A <GlobalKeyID> value for a new symmetric key for the domain with the PEN 10514,**
 1942 **from an SKMS that serves multiple domains:**

1943 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>

1944 **Example 3 – A <GlobalKeyID> value for the 16,777,215th symmetric key generated on 2nd SKS**
 1945 **server for an enterprise with the PEN 10514, in either a <SymkeyRequest> or a <SymkeyResponse>:**

1946 <ekmi:GlobalKeyID>10514-2-16777215</ekmi:GlobalKeyID>

1947 **Example 4 – The maximum <GlobalKeyID> value possible (a 62-byte ASCII decimal), in a**
 1948 **<SymkeyRequest> or <SymkeyResponse>:**

1949 <ekmi:GlobalKeyID>
 1950 18446744073709551615-18446744073709551615-18446744073709551615
 1951 </ekmi:GlobalKeyID>

1952 4.3 Element <KeyClasses> and <KeyClass>

1953 The <KeyClasses> element of type **KeyClassesType**, when specified, identifies at least one
 1954 <KeyClass> element, but may specify an unbounded (unlimited) number of <KeyClass> elements within
 1955 the <KeyClasses> set.

1956 Schema Definition:

```
1957   <xsd:complexType name="KeyClassesType">
1958     <xsd:sequence>
1959       <xsd:element
1960         name="KeyClass"
1961         type="ekmi:KeyClassType"/>
```

```

1962         minOccurs="1"
1963         maxOccurs="unbounded"/>
1964     </xsd:sequence>
1965 </xsd:complexType>
1966
1967     <xsd:simpleType name="KeyClassType">
1968         <xsd:restriction base="xsd:string">
1969             <xsd:maxLength value="255"/>
1970         </xsd:restriction>
1971     </xsd:simpleType>

```

1972 Client applications may request one or more symmetric keys conforming to one or more key classes
1973 required by the application. If the client application is authorized to receive keys conforming to such key
1974 classes, the SKS server will generate and supply them.

1975
1976 The <KeyClasses> element is useful only when requesting new symmetric keys, i.e. symmetric
1977 encryption keys that have previously NOT been used for encrypting data. There is little reason for a
1978 client application to specify the <KeyClasses> element when requesting an existing (escrowed)
1979 symmetric key. This is because the SKS server will return the requested key to authorized clients with
1980 whatever key class is associated with the key, regardless of what key class is specified in the request.
1981 The key class will have been associated with the symmetric key at the time of its generation and cannot
1982 be changed once associated with a key.

1983
1984 When more than one <GlobalKeyID> is specified in the request, there MAY be only one <KeyClass>
1985 element within the <KeyClasses> set. When a key class is not specified in a request, it implies a
1986 request for symmetric key(s) of a default key class configured at the **SKS** server. The default key class
1987 for a site is site-specific.

1988
1989 When the client requires more than one symmetric key, and each key needs to be of a different key
1990 class, there MUST be only one <GlobalKeyID> element followed by as many <KeyClass> elements
1991 inside the <KeyClasses> set as needed by the client application. (Example 5 in this section).

1992
1993 When a client requires many symmetric keys – say five keys – and two or more keys belong to the same
1994 key class, the client MUST send multiple requests to the **SKS** server. One request will contain multiple
1995 <GlobalKeyID> elements with one <KeyClass> element in the <KeyClasses> set, and the other request
1996 will contain one <GlobalKeyID> element and multiple <KeyClass> elements within the <KeyClasses>
1997 set. (Examples 4 and 5 in this section).

1998 **Example 1 – A symmetric key request of a default key class (when no KeyClass is specified):**

```

1999     <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2000         <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
2001     </ekmi:SymkeyRequest>

```

2002 **Example 2 – A request for multiple new symmetric keys, each of a default key class:**

```

2003     <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2004         <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
2005         <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
2006     </ekmi:SymkeyRequest>

```

2007 **Example 3 – A request for a new symmetric key of a specific key class:**

```

2008     <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2009         <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
2010         <ekmi:KeyClasses>
2011             <ekmi:KeyClass>256-Bit-Class</ekmi:KeyClass>

```



```

2061     hvCNAQELBQADggEBACK05PtvZD4WPgl0e+EHUiApzFyCdRzf0pFZtxRwG9lR1PZUWUjmwTNfGFsL
2062     S6kyoHgUfVa5fpT1EU1mXUB/Lmo3hFGyprZjfmD7DwuBcYgmZHv7yHrmGOMIOXjFTACvHpM0v0ce
2063     hVx2e4VE0yhBLu/LdH9awGGdp6Bk2XzxQcs8y6Zz0XZAnPgKQZdjbfKERSsy/d1D8pk5baBk4bd
2064     Zh5680caUrbm9ZReRTVaY5qiQpk0U+tDrBSj/HIL6GAqegYllkz6KYCy6RV0y6iVVSjHocDqdJr
2065     EVOR+ds6xn8mmodlERrILmuxiLpibPp609SfnDIXNlzLwe5g7ep3lc=
2066     </ekmi:X509EncryptionCertificate>
2067 </ekmi:SymkeyRequest>

```

2068 4.5 Element <SymkeyRequestID>

2069 The <SymkeyRequestID> element is the unique identifier of a request for a symmetric encryption key
 2070 within an SKMS. Every request for a symmetric key received by the **SKS** server MUST be assigned a
 2071 unique <SymkeyRequestID> as specified in this section.

2072 Schema Definition:

```

2073     <xsd:simpleType name="SymkeyRequestIDType">
2074         <xsd:restriction base="xsd:string">
2075             <xsd:minLength value="5"/>
2076             <xsd:maxLength value="62"/>
2077             <xsd:pattern value="[0-9]{1,20}-[0-9]{1,20}-[0-9]{1,20}"/>
2078             <xsd:whiteSpace value="collapse"/>
2079         </xsd:restriction>
2080     </xsd:simpleType>

```

2081 The <SymkeyRequestID> element is of the *SymkeyRequestIDType*, and is a string identifier consisting
 2082 of five parts concatenated together:

- 2083 1. A positive integer identifying the **Domain ID**. The **DomainID** identifies the IANA-issued Private
 2084 Enterprise Number (PEN) as published at <http://www.iana.org/assignments/enterprise-numbers>
 2085 and is used by the **SKS** server to constrain the ownership of objects within the SKMS;
- 2086 2. A literal hyphen ("-") without surrounding spaces;
- 2087 3. A positive integer identifying the unique Server ID of the server within the SKMS of the above-
 2088 mentioned domain, that originally received the request;
- 2089 4. Another literal hyphen ("-") without surrounding spaces;
- 2090 5. A positive integer identifying the unique Request ID on the server that received the request;

2091 Combined, the five components of this element make up a unique identifier for a request of a symmetric
 2092 key within the SKMS. Since all enterprise are expected to use only the PENs assigned to them, and
 2093 assuming they do, the <SymkeyRequestID> is unique across the internet.

2094 The **DomainID** in the <SymkeyRequestID> element of a <SymkeyRequest> or <SymkeyResponse> MUST
 2095 always be a non-zero, positive integer in the range of 0 (zero) to 18446744073709551615 (20-byte
 2096 ASCII decimal). It will typically contain the PEN of the domain to which the SKMS belongs.

2097 The **ServerID** part of the <SymkeyRequestID> element of a <SymkeyRequest> or <SymkeyResponse>
 2098 MUST always be a non-zero, positive integer and be in the range of one ("1") to
 2099 18446744073709551615 (20-byte ASCII decimal). It will typically contain the unique server identifier of
 2100 the **SKS** server where the symmetric key request was received.

2101 The **RequestID** part of the <SymkeyRequestID> element of a <SymkeyRequest> or <SymkeyResponse>
 2102 MUST always be a non-zero, positive integer and be in the range of one ("1") to
 2103 18446744073709551615 (20-byte ASCII decimal). It will typically contain the unique request identifier
 2104 on the **SKS** server where the symmetric key request was received.

2105 While the <SymkeyRequestID> and the <GlobalKeyID> appear identical in structure, they are completely
2106 different elements and must be managed by the **SKS** server separately.

2107 **Example 1 – A <SymkeyRequestID> value for the 16,777,215th symmetric-key request received on**
2108 **the 2nd SKS server for an enterprise with the PEN 10514, in either a <SymkeyRequest> or a**
2109 **<SymkeyResponse>:**

2110 <ekmi:SymkeyRequestID>10514-2-16777215</ekmi:SymkeyRequestID>

2111 **Example 2 – The maximum <SymkeyRequestID> value possible (a 62-byte ASCII decimal), in a**
2112 **<SymkeyRequest> or <SymkeyResponse>:**

2113 <ekmi:SymkeyRequestID>
2114 18446744073709551615-18446744073709551615-18446744073709551615
2115 </ekmi:SymkeyRequestID>

2116 4.6 Element <SymkeyResponse>

2117 The <SymkeyResponse> element is the response returned by an **SKS** server upon being sent a valid
2118 <SymkeyRequest> by a client application.

2119 While <SymkeyResponse> is a top-level element within this specification, it **MUST** be enclosed within a
2120 **SOAP Body** element of a **SOAP Envelope** to conform to the security requirements of this specification.
2121 The **SOAP Header** of the **SOAP Envelope** **MUST** enclose a **Security** element conforming to [WSS] with
2122 a **ValueType** attribute containing the value [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3)
2123 [x509-token-profile-1.0#X509v3](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3). The **Security** element must conform to all other requirements of the
2124 specified security profile in [WSS] to form a well-formed, secure message.

2125 Schema Definition:

```
2126 <xsd:element name="SymkeyResponse">
2127   <xsd:complexType>
2128     <xsd:choice>
2129       <xsd:sequence>
2130         <xsd:element
2131           name="Symkey"
2132           type="ekmi:SymkeyType"
2133           minOccurs="1"
2134           maxOccurs="unbounded"/>
2135         <xsd:element
2136           name="SymkeyWorkInProgress"
2137           type="ekmi:SymkeyWorkInProgressType"
2138           minOccurs="0"
2139           maxOccurs="unbounded"/>
2140         <xsd:element
2141           name="SymkeyError"
2142           type="ekmi:SymkeyErrorType"
2143           minOccurs="0"
2144           maxOccurs="unbounded"/>
2145       </xsd:sequence>

2147     <xsd:sequence>
2148       <xsd:element
2149         name="SymkeyWorkInProgress"
2150         type="ekmi:SymkeyWorkInProgressType"
2151         minOccurs="1"
2152         maxOccurs="unbounded"/>
2153     </xsd:sequence>
```

```

2154         name="SymkeyError"
2155         type="ekmi:SymkeyErrorType"
2156         minOccurs="0"
2157         maxOccurs="unbounded"/>
2158     </xsd:sequence>

2160     <xsd:sequence>
2161     <xsd:element
2162         name="SymkeyError"
2163         type="ekmi:SymkeyErrorType"
2164         minOccurs="1"
2165         maxOccurs="unbounded"/>
2166     </xsd:sequence>
2167 </xsd:choice>
2168 </xsd:complexType>
2169 </xsd:element>

```

2170 The <SymkeyResponse> element consists of a choice of one of three sequences of children elements:

- 2171 • A sequence of a required <Symkey> element followed with an optional
- 2172 <SymkeyWorkInProgress> and/or an optional <SymkeyError> element;
- 2173 • A sequence of a required <SymkeyWorkInProgress> element followed by an optional
- 2174 <SymkeyError> element; or
- 2175 • A required <SymkeyError> element .

2176 In any <SymkeyResponse> element that consists of the first two choices, all <Symkey> elements MUST
2177 precede the first <SymkeyWorkInProgress> element and all <SymkeyWorkInProgress> elements MUST
2178 precede the first <SymkeyError> element..

2179 1. <Symkey> [Required]

2180

2181 This element of type **SymkeyType**, is returned by the **SKS** server in response to a successful
2182 processing of a <SymkeyRequest>. There MAY be more than one <Symkey> element in the
2183 <SymkeyResponse> if the client application made a request for multiple symmetric keys.

2184

2185 The <Symkey> element and the **SymkeyType** are specified in Section 4.7.

2186 2. <SymkeyWorkInProgress> [Required and/or Optional]

2187

2188 This element of type **SymkeyWorkInProgressType**, contains a response to a pending process
2189 of a request for one or more symmetric keys. There MAY be more than one
2190 <SymkeyWorkInProgress> element in the <SymkeyResponse> if the client application made a
2191 request for multiple symmetric keys and the request resulted in multiple pending responses.

2192

2193 When the <SymkeyResponse> element contains at least one <Symkey> element, the
2194 <SymkeyWorkInProgress> is an optional element. When the <SymkeyResponse> element
2195 contains only <SymkeyWorkInProgress> and <SymkeyError> elements, the
2196 <SymkeyWorkInProgress> is required.

2197

2198 The <SymkeyWorkInProgress> element and the **SymkeyWorkInProgressType** are specified in
2199 Section 4.8.

2200 3. <SymkeyError> [Required and/or Optional]

2201

2202 This element of type **SymkeyErrorType**, contains a response to a failed attempt in processing a
2203 request for one or more symmetric keys. There MAY be more than one <SymkeyError> element
2204 in the <SymkeyResponse> if the client application made a request for multiple symmetric keys

2205 and the request resulted in multiple errors.
2206
2207 When the <SymkeyResponse> element contains at least one <Symkey> or
2208 <SymkeyWorkInProgress> element, the <SymkeyError> is an optional element; otherwise the
2209 <SymkeyError> is required.
2210
2211 The <SymkeyError> element and **SymkeyErrorType** are specified in Section 4.9.

2212 Some high-level examples of the <SymkeyResponse> element are as follows:

2213 **Example 1 – A response with a single symmetric key:**

```

2214 <ekmi:SymkeyResponse
2215     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2216     <ekmi:Symkey>.....</ekmi:Symkey>
2217 </ekmi:SymkeyResponse>

```

2218 **Example 2 – A response with three symmetric keys:**

```

2219 <ekmi:SymkeyResponse
2220     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2221     <ekmi:Symkey>.....</ekmi:Symkey>
2222     <ekmi:Symkey>.....</ekmi:Symkey>
2223     <ekmi:Symkey>.....</ekmi:Symkey>
2224 </ekmi:SymkeyResponse>

```

2225 **Example 3 – A response with a single work-in-progress element:**

```

2226 <ekmi:SymkeyResponse
2227     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2228     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2229 </ekmi:SymkeyResponse>

```

2230 **Example 4 – A response with multiple work-in-progress elements:**

```

2231 <ekmi:SymkeyResponse
2232     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2233     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2234     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2235     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2236 </ekmi:SymkeyResponse>

```

2237 **Example 5 – A response with multiple symmetric-key and work-in-progress elements:**

```

2238 <ekmi:SymkeyResponse
2239     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2240     <ekmi:Symkey>.....</ekmi:Symkey>
2241     <ekmi:Symkey>.....</ekmi:Symkey>
2242     <ekmi:Symkey>.....</ekmi:Symkey>
2243     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2244     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2245     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2246 </ekmi:SymkeyResponse>

```

2247 **Example 6 – A response with an error:**

```

2248 <ekmi:SymkeyResponse
2249     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2250     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2251 </ekmi:SymkeyResponse>

```


2252 **Example 7 – A response with multiple errors:**

```
2253 <ekmi:SymkeyResponse
2254     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2255     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2256     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2257 </ekmi:SymkeyResponse>
```

2258 **Example 8 – A response with one symmetric key and one error:**

```
2259 <ekmi:SymkeyResponse
2260     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2261     <ekmi:Symkey>.....</ekmi:Symkey>
2262     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2263 </ekmi:SymkeyResponse>
```

2264 **Example 9 – A response with multiple symmetric keys and multiple errors:**

```
2265 <ekmi:SymkeyResponse
2266     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2267     <ekmi:Symkey>.....</ekmi:Symkey>
2268     <ekmi:Symkey>.....</ekmi:Symkey>
2269     <ekmi:Symkey>.....</ekmi:Symkey>
2270     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2271     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2272 </ekmi:SymkeyResponse>
```

2273 **Example 10 – A response with multiple symmetric keys, multiple work-in-progress and multiple**
2274 **error elements:**

```
2275 <ekmi:SymkeyResponse
2276     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2277     <ekmi:Symkey>.....</ekmi:Symkey>
2278     <ekmi:Symkey>.....</ekmi:Symkey>
2279     <ekmi:Symkey>.....</ekmi:Symkey>
2280     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2281     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2282     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2283     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2284     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2285 </ekmi:SymkeyResponse>
```

2286 **4.7 Element <Symkey>**

2287 The <Symkey> element is the *raison d'être* of the **SKSML** protocol. The element of type **SymkeyType**,
2288 contains the symmetric key returned by the **SKS** server, in response to a successful processing of a
2289 <SymkeyRequest> from a client application.

2290 **Schema Definition:**

```
2291 <xsd:complexType name="SymkeyType">
2292     <xsd:sequence>
2293         <xsd:element name="SymkeyRequestID" type="ekmi:SymkeyRequestID"/>
2294         <xsd:element name="GlobalKeyID" type="ekmi:GlobalKeyIDType"/>
2295         <xsd:element name="KeyUsePolicy" type="ekmi:KeyUsePolicyType"/>
2296         <xsd:element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>
2297         <xsd:element ref="xenc:CipherData"/>
2298     </xsd:sequence>
2299 </xsd:complexType>
```

2300 When a request for a symmetric key is successful, there MUST be at least one <Symkey> element in a
2301 <SymkeyResponse> element. There MAY be more than one <Symkey> element in the response if the
2302 client application made a request for multiple symmetric keys and the **SKS** server processed the request
2303 successfully.

2304 In the event an **SKS** server receives the request asynchronously, the response SHALL contain a
2305 <SymkeyWorkInProgress> element implying that the request is being processed and is pending
2306 completion. The <SymkeyError> element is specified in Section 4.7.

2307 In the event of an error in processing the request, there SHALL be no <Symkey> element in the
2308 response; there SHALL be a <SymkeyError> element, instead. The <SymkeyError> element is
2309 specified in Section 4.9.

2310 The <Symkey> element consists of a sequence of the following child elements:

2311 1. <SymkeyRequestID> [Required]

2312
2313 This element of type **SymkeyRequestIDType** identifies the unique identifier of the request
2314 made by the client within an SKMS. There SHALL be only one <SymkeyRequestID> within a
2315 <Symkey> element.

2316
2317 The <SymkeyRequestID> element and the **SymkeyRequestIDType** are specified in Section 4.5.

2318 2. <GlobalKeyID> [Required]

2319
2320 This element of type **GlobalKeyIDType** identifies the unique identifier of the symmetric key
2321 within an SKMS. There SHALL be only one <GlobalKeyID> within a <Symkey> element.

2322
2323 The <GlobalKeyID> element and the **GlobalKeyIDType** are specified in Section 4.2.

2324 3. <KeyUsePolicy> [Required]

2325
2326 This element of type **KeyUsePolicyType**, defines how the symmetric key in this <Symkey>
2327 element may be used by applications. There SHALL be only one <KeyUsePolicy> element
2328 within a <Symkey> element.

2329
2330 The <KeyUsePolicy> element and **KeyUsePolicyType** are specified in Section 4.10.

2331 4. <EncryptionMethod> [Required]

2332
2333 This element of type **EncryptionMethodType** from [XMLEncryption] describes how the
2334 symmetric key in this <Symkey> element is encrypted for transport between the **SKS** Server and
2335 the client application.

2336
2337 The <EncryptionMethod> MUST specify one of the following two transport algorithms in the
2338 **Algorithm** attribute of the element:

2339
2340 - http://www.w3.org/2001/04/xmlenc#rsa-1_5
2341 - <http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p>

2342 5. <CipherData> [Required]

2343
2344 This element of **CipherDataType** from [XMLEncryption] contains the encrypted symmetric-key.
2345 As specified in [XMLEncryption], the content of this element is Base-64 encoded and is of the
2346 XML Schema **base64Binary** type.

2347 Some high-level examples of the <Symkey> element are as follows. Details about the <KeyUsePolicy>
2348 element have been elided for brevity:

Example 1 – A response with a symmetric key:

```
<ekmi:SymkeyResponse
  xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
  <ekmi:Symkey>
    <ekmi:SymkeyRequestID>10514-1-7455</ekmi:SymkeyRequestID>
    <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
    <ekmi:KeyUsePolicy>....</ekmi:KeyUsePolicy>
    <ekmi:EncryptionMethod
      Algorithm="http://www.w3.org/2001/04/xmldenc#rsa-1_5"/>
    <xenc:CipherData>
      <xenc:CipherValue>
        E9zWB/y93hVSzeTLiDcQoDxmLNxTux+SffMNwCJmt1dIqzQHBnpgQ8
        lg6DKdkCFjJMHqhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxj
        fg1pU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
      </xenc:CipherValue>
    </xenc:CipherData>
  </ekmi:Symkey>
</ekmi:SymkeyResponse>
```

Example 2 – A response with multiple symmetric keys:

```
<ekmi:SymkeyResponse
  xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
  <ekmi:Symkey>
    <ekmi:SymkeyRequestID>10514-1-12455</ekmi:SymkeyRequestID>
    <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
    <ekmi:KeyUsePolicy>....</ekmi:KeyUsePolicy>
    <ekmi:EncryptionMethod
      Algorithm="http://www.w3.org/2001/04/xmldenc#rsa-oaep-mgf1p"/>
    <xenc:CipherData>
      <xenc:CipherValue>
        E9zWB/y93hVSzeTLiDcQoDxmLNxTux+SffMNwCJmt1dIqzQHBnpgQ8
        lg6DKdkCFjJMHqhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxj
        fg1pU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
      </xenc:CipherValue>
    </xenc:CipherData>
  </ekmi:Symkey>
  <ekmi:Symkey>
    <ekmi:SymkeyRequestID>10514-1-12467</ekmi:SymkeyRequestID>
    <ekmi:GlobalKeyID>10514-1-236</ekmi:GlobalKeyID>
    <ekmi:KeyUsePolicy>....</ekmi:KeyUsePolicy>
    <ekmi:EncryptionMethod
      Algorithm="http://www.w3.org/2001/04/xmldenc#rsa-oaep-mgf1p"/>
    <xenc:CipherData>
      <xenc:CipherValue>
        Qbg65cy93hVSzeTLiDcQoDxmLNxTux+SffMNwCJmt1dIqzQHBnpgQ8
        7k6DKdkCFjJMHqhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxj
        uyecU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
      </xenc:CipherValue>
    </xenc:CipherData>
  </ekmi:Symkey>
</ekmi:SymkeyResponse>
```

4.8 Element <SymkeyWorkInProgress>

The <SymkeyWorkInProgress> element is returned in a <SymkeyResponse> when a client either makes an asynchronous request for a symmetric-key from an SKS server.

2402 The <SymkeyWorkInProgress> element of type **SymkeyWorkInProgressType**, contains a unique
2403 request ID returned by the **SKS** server, in response to receiving a <SymkeyRequest> from a client
2404 application.

2405 **Schema Definition:**

```
2406     <xsd:complexType name="SymkeyWorkInProgressType">
2407         <xsd:sequence>
2408             <xsd:element
2409                 name="RequestedGlobalKeyID"
2410                 type="ekmi:GlobalKeyIDType"
2411                 minOccurs="1"
2412                 maxOccurs="1">
2413             </xsd:element>
2414
2415             <xsd:element
2416                 name="RequestedKeyClass"
2417                 type="ekmi:KeyClassType"
2418                 minOccurs="0"
2419                 maxOccurs="1">
2420             </xsd:element>
2421
2422             <xsd:element
2423                 name="SymkeyRequestID"
2424                 type="ekmi:SymkeyRequestIDType"
2425                 minOccurs="1"
2426                 maxOccurs="1">
2427             </xsd:element>
2428
2429             <xsd:element
2430                 name="RequestCheckInterval"
2431                 type="ekmi:RequestCheckIntervalType"
2432                 minOccurs="1"
2433                 maxOccurs="1">
2434             </xsd:element>
2435         </xsd:sequence>
2436     </xsd:complexType>
```

2435 **Schema Definition:**

```
2436     <xsd:simpleType name="RequestCheckIntervalType">
2437         <xsd:restriction base="xsd:positiveInteger">
2438             <xsd:minInclusive value="60"/>
2439             <xsd:maxInclusive value="86400"/>
2440         </xsd:restriction>
2441     </xsd:simpleType>
```

2442 The <SymkeyWorkInProgress> element consists of a sequence of the following child elements:

- 2443 1. <RequestedGlobalKeyID> [Required]

2444

2445 This element of type **GlobalKeyIDType** identifies the identifier of the requested symmetric key
2446 within an SKMS. There SHALL be only one <RequestedGlobalKeyID> within a
2447 <SymkeyWorkInProgress> element.

2448

2449 The **GlobalKeyIDType** is specified in Section 4.2.

- 2450 2. <RequestedKeyClass> [Optional]

2451

This element of type **KeyClassType**, identifies the class of the symmetric-key that was requested, if any. If present, there SHALL be only one <RequestedKeyClass> element within a <SymkeyWorkInProgress> element.

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

3. <SymkeyRequestID> [Required]

This element of type **SymkeyRequestIDType** identifies the unique request identifier returned by the **SKS** server. This request identifier may be used by the client to query the **SKS** server for an update on the request at a future time.

The <SymkeyRequestID> element and **SymkeyRequestIDType** are specified in Section 4.5.

4. <RequestCheckInterval> [Required]

This element of **RequestCheckIntervalType** defines the number of seconds a client MUST wait after it has received a <SymkeyWorkInProgress> response before it may query the **SKS** server for an update on the request. The value also indicates that the requesting client MUST wait the same interval between update queries in the event it continues to get a <SymkeyWorkInProgress> response from the server.

The **RequestCheckIntervalType** is a positiveInteger type from the XML Schema Definition and restricts the minimum value to be 60 seconds, and the maximum value to be 604,800 seconds (1-week).

Some high-level examples of the <SymkeyWorkInProgress> element are as follows.:

Example 1 – A response with a work-in-progress element and a check-request frequency interval of 5-minutes:

```
<ekmi:SymkeyResponse xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
  <ekmi:SymkeyWorkInProgress>
    <ekmi:RequestedGlobalKeyID>10514-1-235</ekmi:RequestedGlobalKeyID>
    <ekmi:SymkeyRequestID>10514-1-17964</ekmi:SymkeyRequestID>
    <ekmi:RequestCheckInterval>300</ekmi:RequestCheckInterval>
  </ekmi:SymkeyWorkInProgress>
</ekmi:SymkeyResponse>
```

Example 2 – A response with a work-in-progress element and a check-request frequency interval of 30-minutes:

```
<ekmi:SymkeyResponse xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
  <ekmi:SymkeyWorkInProgress>
    <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
    <ekmi:RequestedKeyClass>HR-Class</ekmi:RequestedKeyClass>
    <ekmi:SymkeyRequestID>10514-3-4955</ekmi:SymkeyRequestID>
    <ekmi:RequestCheckInterval>1800</ekmi:RequestCheckInterval>
  </ekmi:SymkeyWorkInProgress>
</ekmi:SymkeyResponse>
```

4.9 Element <SymkeyError>

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

Schema Definition:

```

2498 <xsd:complexType name="SymkeyErrorType">
2499   <xsd:sequence>
2500     <xsd:element name="SymkeyRequestID" type="ekmi:SymkeyRequestIDType">
2501     <xsd:element name="RequestedGlobalKeyID" type="ekmi:GlobalKeyIDType"/>
2502     <xsd:element
2503       name="RequestedKeyClass"
2504       type="ekmi:KeyClassType"
2505       minOccurs="0"/>
2506     <xsd:element name="ErrorCode">
2507       <xsd:simpleType>
2508         <xsd:restriction base="xsd:string">
2509           <xsd:maxLength value="255"/>
2510         </xsd:restriction>
2511       </xsd:simpleType>
2512     </xsd:element>
2513     <xsd:element name="ErrorMessage">
2514       <xsd:simpleType>
2515         <xsd:restriction base="xsd:string">
2516           <xsd:maxLength value="1024"/>
2517         </xsd:restriction>
2518       </xsd:simpleType>
2519     </xsd:element>
2520   </xsd:sequence>
2521 </xsd:complexType>

```

2522 When a request for a symmetric key fails despite successfully being processed by the SOAP layer, there
 2523 MUST be at least one <SymkeyError> element in a <SymkeyResponse> element. When a
 2524 <SymkeyRequest> fails at the SOAP layer, the response SHALL consist of a **SOAPFault**.

2525 There MAY be more than one <SymkeyError> element in the response if the client application made a
 2526 request for multiple symmetric keys and the **SKS** server failed in processing the request for more than
 2527 one symmetric key.

2528 The <SymkeyError> element consists of a sequence of the following child elements:

2529 1. <SymkeyRequestID> [Required]

2530
 2531 This element of type **SymkeyRequestIDType** identifies the unique identifier of the request
 2532 made by the client within an SKMS. There SHALL be only one <SymkeyRequestID> within a
 2533 <SymkeyError> element.

2534
 2535 The <SymkeyRequestID> element and the **SymkeyRequestIDType** are specified in Section 4.5.

2536 2. <RequestedGlobalKeyID> [Required]

2537
 2538 This element of type **GlobalKeyIDType** identifies the unique identifier of the symmetric key
 2539 requested by the client application. There SHALL be only one <RequestedGlobalKeyID> within
 2540 a <SymkeyError> element.

2541
 2542 The **GlobalKeyIDType** is specified in Section 4.2.

2543 3. <RequestedKeyClass> [Optional]

2544
 2545 This element of type **KeyClassType** identifies the key-class of the symmetric key requested by
 2546 the client application. If the <RequestedKeyClass> element is not embedded in the
 2547 <SymkeyError> element, this implies that the requested symmetric key was for the default key-
 2548 class of the SKMS.

2549
 2550 The **KeyClassType** is specified in Section 4.3.

2551 4. <ErrorCode> [Required]

2552
2553 This element of type **String** identifies a mnemonic code identifying the error the **SKS** Server
2554 experienced in processing the client's symmetric key request.

2555
2556 The <ErrorCode> element SHALL return one of the codes identified in Appendix C of this
2557 specification.

2558 5. <ErrorMessage> [Required]

2559
2560 This element of type **String** identifies a localized message describing the error the **SKS** Server
2561 experienced in processing the client's symmetric key request.

2562
2563 The <ErrorMessage> element SHALL return the appropriate localized version of the message
2564 corresponding to the <ErrorCode> element from Appendix C of this specification.

2565 Some high-level examples of the <SymkeyError> element are as follows.

2566 **Example 1 – An error within a response:**

```
2567 <ekmi:SymkeyResponse
2568   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2569   <ekmi:SymkeyError>
2570     <ekmi:SymkeyRequestID>10514-2-34345</ekmi:SymkeyRequestID>
2571     <ekmi:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>
2572     <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
2573     <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
2574   </ekmi:SymkeyError>
2575 </ekmi:SymkeyResponse>
```

2576 **Example 2 – Multiple errors within a response:**

```
2577 <ekmi:SymkeyResponse
2578   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2579   <ekmi:SymkeyError>
2580     <ekmi:SymkeyRequestID>10514-1-4564345</ekmi:SymkeyRequestID>
2581     <ekmi:RequestedGlobalKeyID>10514-1-0</ekmi:RequestedGlobalKeyID>
2582     <ekmi:ErrorCode>SKS-100001</ekmi:ErrorCode>
2583     <ekmi:ErrorMessage>Invalid GlobalKeyID</ekmi:ErrorMessage>
2584   </ekmi:SymkeyError>
2585   <ekmi:SymkeyError>
2586     <ekmi:SymkeyRequestID>10514-1-4564349</ekmi:SymkeyRequestID>
2587     <ekmi:RequestedGlobalKeyID>10514-1-7522</ekmi:RequestedGlobalKeyID>
2588     <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
2589     <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
2590   </ekmi:SymkeyError>
2591 </ekmi:SymkeyResponse>
```

2592 **4.10 Element <KeyUsePolicy>**

2593 The <KeyUsePolicy> element defines rules that conforming implementations of the **SKCL** MUST adhere
2594 to when using the symmetric key sent by the **SKS** Server. It is an integral part of the <Symkey>
2595 element.

2596 **Schema Definition:**

```
2597 <xsd:complexType name="KeyUsePolicyType" mixed="true">
```



```

2598     <xsd:sequence>
2599         <xsd:element name="KeyUsePolicyID" type="ekmi:TwoPartIDType"/>
2600         <xsd:element name="PolicyName">
2601             <xsd:simpleType>
2602                 <xsd:restriction base="xsd:string">
2603                     <xsd:maxLength value="255"/>
2604                 </xsd:restriction>
2605             </xsd:simpleType>
2606         </xsd:element>
2607         <xsd:element name="KeyClass" type="ekmi:KeyClassType"/>
2608         <xsd:element name="KeyAlgorithm" type="ekmi:EncryptionAlgorithmType"/>
2609         <xsd:element name="KeySize" type="ekmi:KeySizeType"/>
2610         <xsd:element name="Status" type="ekmi:StatusType"/>
2611         <xsd:element name="Permissions" type="ekmi:PermissionsType"/>
2612     </xsd:sequence>
2613 </xsd:complexType>

```

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

1. <KeyUsePolicyID> [Required]

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

The **TwoPartIDType** is specified in Section 4.11.

2. <PolicyName> [Required]

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

3. <KeyClass> [Required]

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

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

4. <KeyAlgorithm> [Required]

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

The <KeyAlgorithm> element is specified in Section 4.12.

5. <KeySize> [Required]

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

Note: It is possible to determine the size of a symmetric key in an **SKCL** implementation without having to send the size in the response. So, why include it? It is our belief that while network bandwidth and compute performance of devices are increasing steadily, encryption is desired in

many small and portable devices. Consequently, it will speed up applications in cryptographic processing if they do not have to determine the size of each key they use. While “protocol purity” demands that implementation issues do not show up in protocol design, we believe it is justified in this case.

The **KeySizeType** is specified in Section 4.13.

6. <Status> [Required]

The <Status> element, of type **StatusType**, identifies the current status of the symmetric key. There SHALL be only one <Status> element within a <KeyUsePolicy> element.

The **StatusType** is specified in Section 4.14.

7. <Permissions> [Required]

The <Permissions> element, of type **PermissionsType**, defines what is permissible to client applications with the symmetric key this element is associated with. It is the responsibility of the conforming **SKCL** implementation to enforce these rules.

An important distinction of this element – unlike most access control rules – is that the absence of sub-elements in the <Permissions> element implies that all permissions are allowed. The presence of sub-elements in this element provide rules to the **SKCL** about what actions are permitted.

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

The **PermissionsType** is specified in Section 4.15.

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

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

```
<ekmi:KeyUsePolicy>
  <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>
  <ekmi:PolicyName>DES-EDE KeyUsePolicy</ekmi:PolicyName>
  <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
  <ekmi:KeyAlgorithm>
    http://www.w3.org/2001/04/xmlenc#tripledes-cbc
  </ekmi:KeyAlgorithm>
  <ekmi:KeySize>192</ekmi:KeySize>
  <ekmi:Status>Active</ekmi:Status>
  <ekmi:Permissions>
    <ekmi:PermittedApplications ekmi:any="false">
      <ekmi:PermittedApplication>
        <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
        <ekmi:ApplicationName>Payroll Application</ekmi:ApplicationName>
        <ekmi:Version>1.0</ekmi:Version>
        <ekmi:DigestAlgorithm>
          http://www.w3.org/2000/09/xmldsig#sha1
        </ekmi:DigestAlgorithm>
        <ekmi:DigestValue>
          229ea73a5e76eabd183663d332b283948a9202a1
        </ekmi:DigestValue>
      </ekmi:PermittedApplication>
    </ekmi:PermittedApplications>
    <ekmi:PermittedDates ekmi:any="false">
      <ekmi:PermittedDate>
```

```

2703         <ekmi:StartDate>2008-01-01</ekmi:StartDate>
2704         <ekmi:EndDate>2008-12-31</ekmi:EndDate>
2705     </ekmi:PermittedDate>
2706 </ekmi:PermittedDates>
2707 <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2708 <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2709 <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2710 <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2711 <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
2712 <ekmi:PermittedTimes ekmi:any="false">
2713     <ekmi:PermittedTime>
2714         <ekmi:StartTime>07:00:00</ekmi:StartTime>
2715         <ekmi:EndTime>19:00:00</ekmi:EndTime>
2716     </ekmi:PermittedTime>
2717 </ekmi:PermittedTimes>
2718 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2719 </ekmi:Permissions>
2720 </ekmi:KeyUsePolicy>

```

2721 **Example 2 – A <KeyUsePolicy> with no restrictions on the key:**

```

2722 <ekmi:KeyUsePolicy>
2723     <ekmi:KeyUsePolicyID>10514-2</ekmi:KeyUsePolicyID>
2724     <ekmi:PolicyName>Laptop KeyUsePolicy</ekmi:PolicyName>
2725     <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
2726     <ekmi:KeyAlgorithm>
2727         http://www.w3.org/2001/04/xmlenc#aes256-cbc
2728     </ekmi:KeyAlgorithm>
2729     <ekmi:KeySize>256</ekmi:KeySize>
2730     <ekmi:Status>Active</ekmi:Status>
2731     <ekmi:Permissions>
2732         <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
2733         <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
2734         <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2735         <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2736         <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2737         <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2738         <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
2739         <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
2740         <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2741     </ekmi:Permissions>
2742 </ekmi:KeyUsePolicy>

```

2743 **4.11 Type *TwoPartIDType***

2744 The ***TwoPartIDType*** is used to create identifiers for many elements within the **SKSML**. It is a simple
 2745 concatenation of two integers with a hyphen between them ("-") to create an XML Schema ***String*** type.

2746 The ***TwoPartIDType*** has a minimum length of three (3) characters, and a maximum length of 41
 2747 characters.

2748 **Schema Definition:**

```

2749 <xsd:simpleType name="TwoPartIDType">
2750     <xsd:restriction base="xsd:string">
2751         <xsd:minLength value="3"/>
2752         <xsd:maxLength value="41"/>
2753         <xsd:pattern value="[1-9][0-9]{0,19}-[1-9][0-9]{0,19}"/>
2754         <xsd:whiteSpace value="collapse"/>

```

```
2755     </xsd:restriction>
2756 </xsd:simpleType>
```

2757 The **TwoPartIDType** is used in the <ApplicationID>, the <KeyCachePolicyID> and the
2758 <KeyUsePolicyID> elements within the **SKSML**.

2759 Some examples of the <KeyUsePolicy> element are as follows.

2760 **Example 1 – A TwoPartIDType used to identify an ApplicationID:**

```
2761     <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
```

2762 **Example 2 – A TwoPartIDType used to identify a KeyUsePolicyID:**

```
2763     <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>
```

2764 **Example 3 – A minimum-length TwoPartIDType :**

```
2765     <ekmi:KeyCachePolicyID>5-4</ekmi:KeyCachePolicyID>
```

2766 **Example 4 – A maximum-length TwoPartIDType :**

```
2767     <ekmi:ApplicationID>
2768         18446744073709551615-18446744073709551615
2769     </ekmi:ApplicationID>
```

2770 4.12 Element <KeyAlgorithm>

2771 The element <KeyAlgorithm> , of type **EncryptionAlgorithmType**, is used to identify the cryptographic
2772 algorithm to be used with the symmetric keys in the <SymkeyResponse>.

2773 Schema Definition:

```
2774     <xsd:simpleType name="EncryptionAlgorithmType">
2775         <xsd:restriction base="xsd:anyURI">
2776             <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
2777             <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
2778             <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes192-cbc"/>
2779             <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
2780         </xsd:restriction>
2781     </xsd:simpleType>
```

2782 The algorithms currently supported by this specification are the algorithms defined in **[XMLEncryption]**.
2783 As new algorithms are added to **[XMLEncryption]**, they will be added to the enumerated list in this
2784 element. Currently, the following four algorithms are supported:

2785 1. Triple Data Encryption Standard (3DES)

2786

2787 Within the context of this specification, and as specified in **[XMLEncryption]**, the form of 3DES
2788 supported within **SKSML** is a 192-bit key with a 64-bit Initialization Vector. Of the key bits, the
2789 first 64 are used in the first DES operation, the second 64 bits in the second (middle) DES
2790 operation, and the third 64 bits in the third (last) DES operation. Each of these 64 bits of key
2791 contain 56 effective bits and 8 parity bits.

2792

2793 The algorithm standard is denoted by the URL: [http://www.w3.org/2001/04/xmlenc#tripledes-](http://www.w3.org/2001/04/xmlenc#tripledes-cbc)
2794 [cbc](http://www.w3.org/2001/04/xmlenc#tripledes-cbc).

2795 2. Advanced Encryption Standard (AES) – 128-bit

2796

2797 Within the context of this specification, and as specified in **[AES]**, this is a 128-bit symmetric key

2798 used in the Cipher Block Chaining (CBC) mode.
 2799
 2800 The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#aes128-cbc>.

2801 3. Advanced Encryption Standard (AES) – 192-bit
 2802
 2803 Within the context of this specification, and as specified in **[AES]**, this is a 192-bit symmetric key
 2804 used in the Cipher Block Chaining (CBC) mode.
 2805
 2806 The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#aes192-cbc>.

2807 4. Advanced Encryption Standard (AES) – 256-bit
 2808
 2809 Within the context of this specification, and as specified in **[AES]**, this is a 256-bit symmetric key
 2810 used in the Cipher Block Chaining (CBC) mode.
 2811
 2812 The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#aes256-cbc>.

2813 There SHALL be only one <KeyAlgorithm> element within a <KeyUsePolicy> element.
 2814 Some examples of the <KeyAlgorithm> element are as follows; other elements of the <KeyUsePolicy>
 2815 element are not displayed for brevity:

2816 **Example 1 – An example using the Triple-DES key algorithm:**

```
2817 <ekmi:KeyUsePolicy>
2818   ...
2819   <ekmi:KeyAlgorithm>
2820     http://www.w3.org/2001/04/xmlenc#tripledes-cbc
2821   </ekmi:KeyAlgorithm>
2822   ...
2823 </ekmi:KeyUsePolicy>
```

2824 **Example 2 – An example using the AES-128 key algorithm:**

```
2825 <ekmi:KeyUsePolicy>
2826   ...
2827   <ekmi:KeyAlgorithm>
2828     http://www.w3.org/2001/04/xmlenc#aes128-cbc
2829   </ekmi:KeyAlgorithm>
2830   ...
2831 </ekmi:KeyUsePolicy>
```

2832 4.13 Element <KeySize>

2833 The element <KeySize> , of type **KeySizeType**, is used to identify the size of the symmetric key, in
 2834 binary digits (bits) in the <SymkeyResponse>.

2835 **Schema Definition:**

```
2836 <xsd:simpleType name="KeySizeType">
2837   <xsd:restriction base="xsd:unsignedShort">
2838     <xsd:totalDigits value="3"/>
2839     <xsd:fractionDigits value="0"/>
2840     <xsd:enumeration value="128"/>
2841     <xsd:enumeration value="192"/>
2842     <xsd:enumeration value="256"/>
2843   </xsd:restriction>
2844 </xsd:simpleType>
```

2845 There SHALL be only one <KeySize> element within a <KeyUsePolicy> element.

2846 Currently, the following three key-sizes are supported:

- 2847 1. 128-bits when used with the **AES-192** algorithm
- 2848 2. 192-bits when used with the **AES-192** or the **3DES** algorithms
- 2849 3. 256-bits when used with the **AES-256** algorithm

2850 Some examples of the <KeySize> element are as follows; other elements of the <KeyUsePolicy>
2851 element are not displayed for brevity:

2852 **Example 1 – An example using a 128-bit key size:**

```
2853 <ekmi:KeyUsePolicy>
2854   ...
2855   <ekmi:KeySize>128</ekmi:KeySize>
2856   ...
2857 </ekmi:KeyUsePolicy>
```

2858 **Example 2 – An example using a 192-bit key size:**

```
2859 <ekmi:KeyUsePolicy>
2860   ...
2861   <ekmi:KeySize>192</ekmi:KeySize>
2862   ...
2863 </ekmi:KeyUsePolicy>
```

2864 **Example 3 – An example using a 256-bit key size:**

```
2865 <ekmi:KeyUsePolicy>
2866   ...
2867   <ekmi:KeySize>256</ekmi:KeySize>
2868   ...
2869 </ekmi:KeyUsePolicy>
```

2870 4.14 Element <Status>

2871 The element <Status>, of type **StatusType**, is used to identify the current status of an object . It is
2872 used in almost every element within the SKMS.

2873 **Schema Definition:**

```
2874 <xsd:simpleType name="StatusType">
2875   <xsd:restriction base="xsd:string">
2876     <xsd:enumeration value="Active"/>
2877     <xsd:enumeration value="Default"/>
2878     <xsd:enumeration value="Inactive"/>
2879     <xsd:enumeration value="Other"/>
2880   </xsd:restriction>
2881 </xsd:simpleType>
```

2882 Where it exists within an element, there SHALL be only one <Status> element within the enclosing
2883 element.

2884 The <Status> element can contain one of four **String** type values:

- 2885 1. The **Active** value indicates that the element that makes up the document-root is currently active
2886 in the SKMS and conforming **SKCL** implementations may use it within applications.

- 2887 2. The **Default** value indicates that the element that makes up the document root is the default
2888 element in the SKMS, is also active, and conforming **SKCL** implementations may use it within
2889 applications.
- 2890 3. The **Inactive** value indicates that the element that makes up the document root is not active in
2891 the SKMS, and conforming **SKCL** implementations may NOT use it within applications.
- 2892 4. The **Other** value indicates that the element that makes up the document root has a meaning that
2893 is application-specific. However, conforming **SKCL** implementations may NOT use it within
2894 applications.

2895 Some examples of the <Status> element are shown below; other parts of their enclosing elements are
2896 not shown for brevity:

2897 **Example 1 – An example with an Active status within a <KeyUsePolicy> element:**

```
2898 <ekmi:KeyUsePolicy>  
2899 ...  
2900 <ekmi:Status>Active</ekmi:Status>  
2901 ...  
2902 </ekmi:KeyUsePolicy>
```

2903 **Example 2 – An example with an Inactive status within a <KeyUsePolicy> element:**

```
2904 <ekmi:KeyUsePolicy>  
2905 ...  
2906 <ekmi:Status>Inactive</ekmi:Status>  
2907 ...  
2908 </ekmi:KeyUsePolicy>
```

2909 **Example 3 – An example with a Default status within a <KeyUsePolicy> element:**

```
2910 <ekmi:KeyUsePolicy>  
2911 ...  
2912 <ekmi:Status>Default</ekmi:Status>  
2913 ...  
2914 </ekmi:KeyUsePolicy>
```

2915 4.15 Element <Permissions>

2916 The <Permissions> element, of the type **PermissionsType** is at the heart of the <KeyUsePolicy>
2917 element. It provides guidance to conforming **SKCL** implementations on who may use the symmetric key,
2918 when they may use it, for what purposes, for how long and in which locations. For applications that
2919 conform to the Multi-Level Security (MLS) model, there is a provision for specifying which levels are
2920 permitted use of the key. There is also an element that allows for extending the <Permissions> element
2921 to accommodate rules that have not been envisioned in the current specification.

2922 There SHALL be only one <Permissions> element within a <KeyUsePolicy> element.

2923 **Schema Definition:**

```
2924 <xsd:complexType name="PermissionsType">  
2925 <xsd:sequence>  
2926 <xsd:element  
2927     name="PermittedApplications"  
2928     type="ekmi:PermittedApplicationsType"  
2929     minOccurs="1"  
2930     nillable="true"/>  
2931 <xsd:element  
2932     name="PermittedDates"
```



```

2933         type="ekmi:PermittedDatesType"
2934         minOccurs="1"
2935         nillable="true"/>
2936     <xsd:element
2937         name="PermittedDays"
2938         type="ekmi:PermittedDaysType"
2939         minOccurs="1"
2940         nillable="true"/>
2941     <xsd:element
2942         name="PermittedDuration"
2943         type="ekmi:PermittedDurationType"
2944         minOccurs="1"
2945         nillable="true"/>
2946     <xsd:element
2947         name="PermittedLevels"
2948         type="ekmi:PermittedLevelsType"
2949         minOccurs="1"
2950         nillable="true"/>
2951     <xsd:element
2952         name="PermittedLocations"
2953         type="ekmi:PermittedLocationsType"
2954         minOccurs="1"
2955         nillable="true"/>
2956     <xsd:element
2957         name="PermittedNumberOfTransactions"
2958         type="ekmi:PermittedNumberOfTransactionsType"
2959         minOccurs="1"
2960         nillable="true"/>
2961     <xsd:element
2962         name="PermittedTimes"
2963         type="ekmi:PermittedTimesType"
2964         minOccurs="1"
2965         nillable="true"/>
2966     <xsd:element
2967         name="PermittedUses"
2968         type="ekmi:PermittedUsesType"
2969         minOccurs="1"
2970         nillable="true"/>
2971     <xsd:element
2972         name="Other"
2973         type="xsd:anyType"
2974         minOccurs="0"/>
2975 </xsd:sequence>
2976 </xsd:complexType>

```

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

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

The <PermittedApplications> element is specified in Section 4.16.
2. A required <PermittedDates> element which identifies the date ranges during which applications are permitted use of the symmetric key in question. While the <PermittedDates> element is required, it may be empty (NULL). The absence of sub-elements in the

<PermittedDates> element implies that applications are permitted to use the key on any date. Identifying specific date ranges restricts the use of the key to only the duration between the identified dates.

The <PermittedDates> element is specified in Section 4.17.

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

The <PermittedDays> element is specified in Section 4.18.

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

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

The <PermittedDuration> element is specified in Section 4.19.

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

The <PermittedLevels> element is specified in Section 4.20.

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

The <PermittedLocations> element is specified in Section 4.21.

7. A required <PermittedNumberOfTransactions> element which identifies the number of encryption transactions that applications are permitted, with the use of the symmetric key in question. While the <PermittedNumberOfTransactions> element is required, it may be empty (NULL). The absence of content – the number of transactions – in the <PermittedNumberOfTransactions> element implies that applications are permitted to use the key for as many encryption transactions as necessary. Identifying a specific, non-zero, positive number of transactions in this element restricts the use of the key to only the limit identified in the

3041 element.

3042

3043 The <PermittedNumberOfTransactions> element is specified in Section 4.22.

3044 8. A required <PermittedTimes> element which identifies the times of day during which
3045 applications are permitted the use of the symmetric key in question. While the
3046 <PermittedTimes> element is required, it may be empty (NULL). The absence of sub-elements
3047 in the <PermittedTimes> element implies that applications are permitted to use the key at any
3048 time of day. Identifying specific times restricts the use of the key to only the duration of the
3049 identified times.

3050

3051 The <PermittedTimes> element is specified in Section 4.23.

3052 9. A required <PermittedUses> element which identifies application-uses that applications are
3053 permitted with the symmetric key in question. While the <PermittedUses> element is required, it
3054 may be empty (NULL). The absence of sub-elements in the <PermittedUses> element implies
3055 that applications are permitted to use the key for any purpose. Identifying specific uses restricts
3056 the use of the key to only the identified uses.

3057

3058 The <PermittedUses> element is specified in Section 4.24.

3059 10. The optional <Other> element allows implementers to specify permissions that cannot be
3060 addressed with the above-mentioned categories, for restricting the use of the symmetric key in
3061 question.

3062

3063 While the <Other> element provides flexibility for implementations, the disadvantage of the
3064 element is that it may render a specific implementation incompatible with other **SKMS**
3065 implementations that use the **SKSML** standard.

3066

3067 **It is strongly recommended that implementers avoid the use of the <Other> element**
3068 **unless they definitely do not expect to inter-operate with other SKCL implementations. If**
3069 **there is a strong need for capability that does not exist within the current specification of**
3070 **the <Permissions> element, implementers are encouraged to contact the OASIS EKMI TC**
3071 **with their requirements.**

3072 When all sub-elements of the <Permissions> element are empty, there are no restrictions on the use of
3073 the symmetric key other than that the application calling on the **SKCL** be authorized to access the key in
3074 question. However, when there are elements defined within the sub-elements of the <Permissions>
3075 element, conforming **SKCL** implementations must comply with all the permission elements, evaluating
3076 the most restrictive permissions first and in decreasing order of restriction, before allowing the use of the
3077 key.

3078 For example, if a <Permissions> element specifies that a key may be used on Weekdays, between the
3079 hours of 0900 and 1700 Hours, then a request for a symmetric key on a Saturday at 1105 would deny
3080 use of the key in question, since it violates the more restrictive permission of being allowed for use only
3081 on weekdays.

3082 **It should be noted that it is the primary responsibility of a conforming SKCL to enforce the**
3083 **<Permission> elements' rules. The SKS server will generate the key – or return an existing key -**
3084 **when an authorized client with appropriate access requests it. However, it is up to the SKCL**
3085 **implementation to comply with the rules in the <Permissions> element.**

3086 In another example, if a <Permissions> element specifies a <PermittedDuration> of 600 seconds from
3087 the start of use of the key, and there is also present a <PermittedNumberOfTransactions> element with
3088 a value of 10 (encryption transactions), conforming **SKCL** implementations must evaluate both
3089 permissions before each transaction and determine if they are both within the specified thresholds before
3090 using the key. If the 600 seconds expire before the 10 encryption transactions have been completed, or

3091 if the 10 encryption transactions are completed before 600 seconds have expired, conforming **SKCL**
3092 implementations **MUST** not use the key in question anymore.

3093 Some examples of the <Permissions> element are as shown below; the enclosing <KeyUsePolicy>
3094 element, <Symkey> element and <SymkeyResponse> elements are not displayed for brevity:

3095 **Example 1 – A <Permissions> element that permits a single application the use of the symmetric**
3096 **key in question, between January 01, 2008 and December 31, 2008 and between the hours of 0700**
3097 **and 1900. There are, however, no restrictions on what days of the week the key may be used, the**
3098 **locations where it may be used, at which MLS level it may be used (if it applies), the number of**
3099 **data files/transactions that may be encrypted with the key or the uses of the key within that**
3100 **application:**

```
3101     <ekmi:Permissions>
3102         <ekmi:PermittedApplications ekmi:any="false">
3103             <ekmi:PermittedApplication>
3104                 <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
3105                 <ekmi:ApplicationName>Payroll Application</ekmi:ApplicationName>
3106                 <ekmi:Version>1.0</ekmi:Version>
3107                 <ekmi:DigestAlgorithm>
3108                     http://www.w3.org/2000/09/xmldsig#sha1
3109                 </ekmi:DigestAlgorithm>
3110                 <ekmi:DigestValue>
3111                     229ea73a5e76eabd183663d332b283948a9202a1
3112                 </ekmi:DigestValue>
3113             </ekmi:PermittedApplication>
3114         </ekmi:PermittedApplications>
3115         <ekmi:PermittedDates ekmi:any="false">
3116             <ekmi:PermittedDate>
3117                 <ekmi:StartDate>2008-01-01</ekmi:StartDate>
3118                 <ekmi:EndDate>2008-12-31</ekmi:EndDate>
3119             </ekmi:PermittedDate>
3120         </ekmi:PermittedDates>
3121         <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
3122         <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
3123         <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
3124         <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
3125         <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
3126         <ekmi:PermittedTimes ekmi:any="false">
3127             <ekmi:PermittedTime>
3128                 <ekmi:StartTime>07:00:00</ekmi:StartTime>
3129                 <ekmi:EndTime>19:00:00</ekmi:EndTime>
3130             </ekmi:PermittedTime>
3131         </ekmi:PermittedTimes>
3132         <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
3133     </ekmi:Permissions>
```

3134 **Example 2 – A <Permissions> element that permits two specific applications the use of the**
3135 **symmetric key in question, between January 01, 2009 and January 31, 2009.**

```
3136     <ekmi:Permissions>
3137         <ekmi:PermittedApplications ekmi:any="false">
3138             <ekmi:PermittedApplication>
3139                 <ekmi:ApplicationID>10514-24</ekmi:ApplicationID>
3140                 <ekmi:ApplicationName>
3141                     Employee Tax Reporting Application
3142                 </ekmi:ApplicationName>
3143                 <ekmi:Version>3.3</ekmi:Version>
3144                 <ekmi:DigestAlgorithm>
```

```

3145         http://www.w3.org/2000/09/xmldsig#sha1
3146     </ekmi:DigestAlgorithm>
3147     <ekmi:DigestValue>
3148         af96d65a7a2415239c8eb8be1347f704322957a4
3149     </ekmi:DigestValue>
3150 </ekmi:PermittedApplication>
3151 <ekmi:PermittedApplication>
3152     <ekmi:ApplicationID>10514-25</ekmi:ApplicationID>
3153     <ekmi:ApplicationName>
3154         IRS Tax Reporting Application
3155     </ekmi:ApplicationName>
3156     <ekmi:Version>2.1</ekmi:Version>
3157     <ekmi:DigestAlgorithm>
3158         http://www.w3.org/2000/09/xmldsig#sha1
3159     </ekmi:DigestAlgorithm>
3160     <ekmi:DigestValue>
3161         a4f5925185ffe12c1a91ea3de90fc086b34b34b2
3162     </ekmi:DigestValue>
3163 </ekmi:PermittedApplication>
3164 </ekmi:PermittedApplications>
3165 <ekmi:PermittedDates ekmi:any="false">
3166     <ekmi:PermittedDate>
3167         <ekmi:StartDate>2009-01-01</ekmi:StartDate>
3168         <ekmi:EndDate>2009-12-31</ekmi:EndDate>
3169     </ekmi:PermittedDate>
3170 </ekmi:PermittedDates>
3171 <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
3172 <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
3173 <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
3174 <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
3175 <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
3176 <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
3177 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
3178 </ekmi:Permissions>

```

3179 **Example 3 – A <Permissions> element that permits all applications the use of the symmetric key**
3180 **in question, for 100 transactions for encrypting credit card numbers.**

```

3181 <ekmi:Permissions>
3182     <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
3183     <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
3184     <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
3185     <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
3186     <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
3187     <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
3188     <ekmi:PermittedNumberOfTransactions ekmi:any="false">
3189         100
3190     </ekmi:PermittedNumberOfTransactions>
3191     <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
3192     <ekmi:PermittedUses ekmi:any="false">
3193         <ekmi:PermittedUse>CCN</ekmi:PermittedUse>
3194     </ekmi:PermittedUses>
3195 </ekmi:Permissions>

```

3196 **Example 4 – A <Permissions> element that permits all applications the use of the symmetric key**
3197 **in question, for 600 seconds once the SKCL starts using the key.**

```

3198 <ekmi:Permissions>
3199     <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
3200     <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>

```

```

3201     <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
3202     <ekmi:PermittedDuration ekmi:any="false">600</ekmi:PermittedDuration>
3203     <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
3204     <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
3205     <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
3206     <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
3207     <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
3208 </ekmi:Permissions>

```

3209 **Example 5 – A <Permissions> element that permits a specific application the use of the**
 3210 **symmetric key in question, at specific geographic locations only on weekdays between the hours**
 3211 **of 0800 and 1700, and only when the application is operating at the Secret security level within an**
 3212 **MLS system.**

```

3213 <ekmi:Permissions>
3214   <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
3215   <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
3216   <ekmi:PermittedDays ekmi:any="false">
3217     <ekmi:PermittedDay>Weekday</ekmi:PermittedDay>
3218   </ekmi:PermittedDays>
3219   <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
3220   <ekmi:PermittedLevels ekmi:any="false">
3221     <ekmi:PermittedLevel>Secret</ekmi:PermittedLevel>
3222   </ekmi:PermittedLevels>
3223   <ekmi:PermittedLocations ekmi:any="false">
3224     <ekmi:PermittedLocation>
3225       <ekmi:LocationName>Facility A51</ekmi:LocationName>
3226       <ekmi:Latitude>37.385562</ekmi:Latitude>
3227       <ekmi:Longitude>-121.993387</ekmi:Latitude>
3228     </ekmi:PermittedLocation>
3229     <ekmi:PermittedLocation>
3230       <ekmi:LocationName>Facility DC-VA01</ekmi:LocationName>
3231       <ekmi:Latitude>88.485362</ekmi:Latitude>
3232       <ekmi:Longitude>-21.453648</ekmi:Latitude>
3233     </ekmi:PermittedLocation>
3234   </ekmi:PermittedLocations>
3235   <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
3236   <ekmi:PermittedTimes ekmi:any="false">
3237     <ekmi:PermittedTime>
3238       <ekmi:StartTime>08:00:00</ekmi:StartTime>
3239       <ekmi:EndTime>17:00:00</ekmi:EndTime>
3240     </ekmi:PermittedTime>
3241   </ekmi:PermittedTimes>
3242   <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
3243 </ekmi:Permissions>

```

3244 **4.16 Element <PermittedApplications> and <PermittedApplication>**

3245 The element <PermittedApplications>, of type **PermittedApplicationsType** and its only child-
 3246 element <PermittedApplication> of type **ApplicationsType** are used to define the list of applications
 3247 that are permitted to use a symmetric key within a specific <Symkey> element.

3248 **Schema Definition:**

```

3249   <xsd:complexType name="PermittedApplicationsType">
3250     <xsd:sequence>
3251       <xsd:element
3252         name="PermittedApplication"
3253         type="ekmi:ApplicationsType"

```

```

3254         minOccurs="0"
3255         maxOccurs="unbounded"/>
3256     </xsd:sequence>
3257     <xsd:attribute ref="ekmi:any" use="required"/>
3258 </xsd:complexType>

```

Schema Definition:

```

3259
3260
3261     <xsd:complexType name="ApplicationsType">
3262         <xsd:sequence>
3263             <xsd:element name="ApplicationID" type="ekmi:TwoPartIDType"/>
3264             <xsd:element name="ApplicationName">
3265                 <xsd:simpleType>
3266                     <xsd:restriction base="xsd:string">
3267                         <xsd:maxLength value="256"/>
3268                         <xsd:whiteSpace value="preserve"/>
3269                     </xsd:restriction>
3270                 </xsd:simpleType>
3271             </xsd:element>
3272             <xsd:element name="Version" minOccurs="0">
3273                 <xsd:simpleType>
3274                     <xsd:restriction base="xsd:string">
3275                         <xsd:maxLength value="32"/>
3276                         <xsd:whiteSpace value="preserve"/>
3277                     </xsd:restriction>
3278                 </xsd:simpleType>
3279             </xsd:element>
3280             <xsd:group ref="ekmi:MessageDigestGroup" minOccurs="0"/>
3281             <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
3282         </xsd:sequence>
3283     </xsd:complexType>

```

Schema Definition:

```

3284
3285
3286     <xsd:group name="MessageDigestGroup">
3287         <xsd:sequence>
3288             <xsd:element name="DigestAlgorithm">
3289                 <xsd:simpleType>
3290                     <xsd:restriction base="xsd:anyURI">
3291                         <xsd:enumeration value="http://www.w3.org/2000/09/xmlsig#sha1"/>
3292                         <xsd:enumeration value="http://www.w3.org/2001/04/xmldsig#sha256"/>
3293                         <xsd:enumeration value="http://www.w3.org/2001/04/xmldsig#sha512"/>
3294                     </xsd:restriction>
3295                 </xsd:simpleType>
3296             </xsd:element>
3297             <xsd:element name="DigestValue">
3298                 <xsd:simpleType>
3299                     <xsd:restriction base="xsd:base64Binary">
3300                         <xsd:maxLength value="1024"/>
3301                     </xsd:restriction>
3302                 </xsd:simpleType>
3303             </xsd:element>
3304         </xsd:sequence>
3305     </xsd:group>

```

Schema Definition:

```

3307     <xsd:attribute name="any">
3308         <xsd:simpleType>
3309             <xsd:restriction base="xsd:string">

```



```

3310         <xsd:enumeration value="false"/>
3311         <xsd:enumeration value="true"/>
3312     </xsd:restriction>
3313 </xsd:simpleType>
3314 </xsd:attribute>

```

3315 There SHALL be only one <PermittedApplications> element within the <Permissions> element.
 3316 However, there MAY be an unbounded (unlimited) number of <PermittedApplication> elements within
 3317 a <PermittedApplications> element.

3318 The <PermittedApplications> element SHALL have one attribute named "any", that will have a
 3319 "false" or "true" value, based on the following:

- 3320 • When the <PermittedApplications> element is null (i.e. it does not have a single
 3321 <PermittedApplication> sub-element in it), the value of the "any" attribute SHALL be set to
 3322 "true" AND the XML Schema Instance (XSI) "nil" attribute SHALL be set to "true".
- 3323 • When the <PermittedApplications> element is not-null (i.e. it has at least one
 3324 <PermittedApplication> sub-element in it), the value of the "any" attribute SHALL be set to
 3325 "false" AND the XML Schema Instance (XSI) "nil" attribute SHALL NOT be present.

3326 A null <PermittedApplications> element specifies that ALL applications are permitted use of the
 3327 symmetric key, subject to complying with all other permission clauses in the <KeyUsePolicy> element.

3328 The <PermittedApplication> sub-element of type **ApplicationsType**, provides details of the
 3329 application which is permitted use of the symmetric key in question. The <PermittedApplication>
 3330 element consists of the following sub-elements:

- 3331 1. The <ApplicationID> element identifies the unique identifier assigned to this application within
 3332 the SKMS. It is a **TwoPartIDType** as specified in Section 4.10. There SHALL be only one
 3333 <ApplicationID> element within a <PermittedApplication> element.
 - 3334 2. The <ApplicationName> element identifies the name assigned to this application within the
 3335 SKMS. It is an XSD **String** with a maximum length of 256 characters. There SHALL be only
 3336 one <ApplicationName> element within a <PermittedApplication> element.
 - 3337 3. An optional <Version> element identifying the version number of this application within the
 3338 **SKMS**. It is an XSD **String** with a maximum length of 32 characters. There MAY be only one
 3339 <Version> element within a <PermittedApplication> element.
 - 3340 4. The <MessageDigestGroup> group which identifies the message digest value of the
 3341 application's binary image, along with the message digest algorithm used to calculate the digest
 3342 value. The <MessageDigestGroup> consists of the following elements:
 - 3343 a) The <DigestAlgorithm> element of the XSD type **anyURI**, which supports one of the
 3344 following three digest algorithms:
 - 3345 i. <http://www.w3.org/2000/09/xmldsig#sha1>
 - 3346 ii. <http://www.w3.org/2001/04/xmldsig#sha256>
 - 3347 iii. <http://www.w3.org/2001/04/xmldsig#sha512>
 - 3348 b) The <DigestValue> element of the XSD type **base64Binary**.
- 3349 There SHALL be only one <MessageDigestGroup> group within a <PermittedApplication>
 3350 element.
- 3351 5. An optional <Other> element that provides implementers the ability to carry other information
 3352 about the application that may be relevant to their **SKMS**. Implementers are cautioned that the
 3353 use of the <Other> element may not be supported by other **SKCL** implementations, and may

3354 break interoperability between two **SKMS** implementations. Should there be a strong need for
3355 additional features in the <PermittedApplication> element, implementers are encouraged to
3356 contact the OASIS EKMI TC with their requirements.

3357 NOTE: The **SKSML** specification does not specify how an **SKCL** implementation will determine the
3358 message digest of an application that needs to use the symmetric key in question. It is left to the
3359 implementers of the **SKCL** to determine the message digest of the calling application using the specified
3360 algorithm, and verify that the digest values match before the **SKCL** uses the symmetric key on behalf of
3361 the application.

3362 Some examples of the <PermittedApplications> element are shown below; other parts of their
3363 enclosing elements are not shown for brevity:

3364 **Example 1 – An example of a <PermittedApplications> element with two child**
3365 **<PermittedApplication> elements with specific version numbers and message digest values:**

```
3366     <ekmi:PermittedApplications ekmi:any="false">
3367         <ekmi:PermittedApplication>
3368             <ekmi:ApplicationID>10514-24</ekmi:ApplicationID>
3369             <ekmi:ApplicationName>Employee Tax
3370 Reporting</ekmi:ApplicationName>
3371             <ekmi:Version>3.3</ekmi:Version>
3372             <ekmi:DigestAlgorithm>
3373                 http://www.w3.org/2000/09/xmlsig#sha1
3374             </ekmi:DigestAlgorithm>
3375             <ekmi:DigestValue>G4bsdfKkt4cziEqFFu0oBTM81efU=</ekmi:DigestValue>
3376         </ekmi:PermittedApplication>
3377         <ekmi:PermittedApplication>
3378             <ekmi:ApplicationID>10514-25</ekmi:ApplicationID>
3379             <ekmi:ApplicationName>IRS Tax Reporting
3380 Application</ekmi:ApplicationName>
3381             <ekmi:Version>2.1</ekmi:Version>
3382             <ekmi:DigestAlgorithm>
3383                 http://www.w3.org/2001/04/xmlenc#sha256
3384             </ekmi:DigestAlgorithm>
3385             <ekmi:DigestValue>
3386
3387                 ab7b85c9410d48c54fc7939c391be4028e7305085191c56e7b3740f2cbdbbc79
3388             </ekmi:DigestValue>
3389         </ekmi:PermittedApplication>
3390     </ekmi:PermittedApplications>
```

3391 **Example 2 – An example of a <PermittedApplications> element with one child**
3392 **<PermittedApplication> element that applies to all versions of the application; the message**
3393 **digest value and algorithm are not used in this example:**

```
3394     <ekmi:PermittedApplications ekmi:any="false">
3395         <ekmi:PermittedApplication>
3396             <ekmi:ApplicationID>10514-14</ekmi:ApplicationID>
3397             <ekmi:ApplicationName>E-Commerce Payment</ekmi:ApplicationName>
3398         </ekmi:PermittedApplication>
3399     </ekmi:PermittedApplications>
```

3400 **Example 3 – An example of a null <PermittedApplications> element specifying that ALL**
3401 **applications are permitted the use of the symmetric key:**

```
3402     <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
```

4.17 Element <PermittedDates> and <PermittedDate>

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

Schema Definition:

```
<xsd:complexType name="PermittedDatesType">
  <xsd:sequence>
    <xsd:element name="PermittedDate" minOccurs="0" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="StartDate">
            <xsd:simpleType>
              <xsd:restriction base="xsd:date">
                <xsd:pattern value="\p{Nd}{4}-\p{Nd}{2}-\p{Nd}{2}"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="EndDate">
            <xsd:simpleType>
              <xsd:restriction base="xsd:date">
                <xsd:pattern value="\p{Nd}{4}-\p{Nd}{2}-\p{Nd}{2}"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute ref="ekmi:any" use="required"/>
</xsd:complexType>
```

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

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

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

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

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

- The <StartDate> element identifies the date from which applications MAY start using the symmetric key in question. It is an XSD **Date** type that MUST be specified in a specific pattern

(see examples) where the first four digits specify the year, the second two digits specify the calendar month in the year, and the last two digits specify the calendar date of the month.

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

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

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

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

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

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

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

```
<ekmi:PermittedDates ekmi:any="false">
  <ekmi:PermittedDate>
    <ekmi:StartDate>2009-01-01</ekmi:StartDate>
    <ekmi:EndDate>2009-01-31</ekmi:EndDate>
  </ekmi:PermittedDate>
</ekmi:PermittedDates>
```

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

```
<ekmi:PermittedDates ekmi:any="false">
  <ekmi:PermittedDate>
    <ekmi:StartDate>2008-07-01</ekmi:StartDate>
    <ekmi:EndDate>2008-07-03</ekmi:EndDate>
  </ekmi:PermittedDate>
  <ekmi:PermittedDate>
    <ekmi:StartDate>2008-07-07</ekmi:StartDate>
    <ekmi:EndDate>2009-07-12</ekmi:EndDate>
  </ekmi:PermittedDate>
</ekmi:PermittedDates>
```

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

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

4.18 Element <PermittedDays> and <PermittedDay>

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

Schema Definition:

```
<xsd:complexType name="PermittedDaysType">
  <xsd:sequence>
    <xsd:element
      name="PermittedDay"
      type="ekmi:PermittedDayType"
      minOccurs="0"
      maxOccurs="unbounded">
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute ref="ekmi:any" use="required"/>
</xsd:complexType>
```

Schema Definition:

```
<xsd:simpleType name="PermittedDayType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Sunday"/>
    <xsd:enumeration value="Monday"/>
    <xsd:enumeration value="Tuesday"/>
    <xsd:enumeration value="Wednesday"/>
    <xsd:enumeration value="Thursday"/>
    <xsd:enumeration value="Friday"/>
    <xsd:enumeration value="Saturday"/>
    <xsd:enumeration value="Weekday"/>
    <xsd:enumeration value="Weekend"/>
  </xsd:restriction>
</xsd:simpleType>
```

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

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

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

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

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

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

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

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

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

```
<ekmi:PermittedDays ekmi:any="false">
  <ekmi:PermittedDay>Monday</ekmi:PermittedDay>
  <ekmi:PermittedDay>Wednesday</ekmi:PermittedDay>
  <ekmi:PermittedDay>Friday</ekmi:PermittedDay>
</ekmi:PermittedDays>
```

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

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

4.19 Element <PermittedDuration>

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

Schema Definition:

```
<xsd:complexType name="PermittedDurationType">
  <xsd:simpleContent>
    <xsd:extension base="ekmi:DurationType">
      <xsd:attribute ref="ekmi:any" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Schema Definition:

```
<xsd:simpleType name="DurationType">
  <xsd:restriction base="xsd:positiveInteger">
    <xsd:minInclusive value="1"/>
    <xsd:maxInclusive value="18446744073709551615"/>
  </xsd:restriction>
</xsd:simpleType>
```

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

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

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

3593 A null <PermittedDuration> element specifies that applications are permitted use of the symmetric key
3594 indefinitely, subject to complying with all other permission clauses in the <Permissions> element.

3595 The <PermittedDuration> element, of the XSD *positiveInteger* type, identifies the number of seconds
3596 for which the symmetric key in question may be used, ONCE the key has been used by conforming
3597 **SKCL** implementations for the first time. The values for <PermittedDuration> may range between 1
3598 and 18446744073709551615.

3599 As long as the symmetric has not been used by an **SKCL** on a client device (it might be cached for many
3600 days/weeks/months depending on the <KeyCachePolicy> in effect within the SKMS for that device) the
3601 effective lifetime of the symmetric key may be well past the number of seconds specified in
3602 <PermittedDuration> when calculated from the time of the key's generation time on the **SKS** server. It
3603 is the responsibility of the **SKCL** implementation, when presented with a <PermittedDuration> element
3604 in a <KeyUsePolicy> of a symmetric key, to keep track of the date/time when the symmetric key in
3605 question was first used on the client device, and how long the key will last after that.

3606 Examples of the <PermittedDuration> element are shown below; other parts of their enclosing
3607 elements are not shown for brevity:

3608 **Example 1 – An example of a <PermittedDuration> element specifying that the symmetric key**
3609 **may be used only for a single 24-hour period from the moment it is first used by an SKCL:**

3610 `<ekmi:PermittedDuration ekmi:any="false">86400</ekmi:PermittedDuration>`

3611 **Example 2 – An example of a <PermittedDuration> element specifying that the symmetric key**
3612 **may be used only for week from the moment it is first used by an SKCL:**

3613 `<ekmi:PermittedDuration ekmi:any="false">604800</ekmi:PermittedDuration>`

3614 **Example 3 – An example of a <PermittedDuration> element specifying that the symmetric key**
3615 **may be used only 5 minutes from the moment it is first used by an SKCL:**

3616 `<ekmi:PermittedDuration ekmi:any="false">300</ekmi:PermittedDuration>`

3617 **Example 4 – An example of a null <PermittedDuration> element specifying that the symmetric**
3618 **key may be used indefinitely by an SKCL:**

3619 `<ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>`

3620 4.20 Element <PermittedLevels> and <PermittedLevel>

3621 The element <PermittedLevels>, of the type *LevelClassificationType*, is used to define the security
3622 level at which applications are permitted use of a symmetric key. This element is useful only to
3623 applications and systems that conform to the multi-level security (MLS) system as defined in the Bell-
3624 LaPadula model.

3625 Schema Definition:

```
3626 <xsd:complexType name="PermittedLevelsType">
3627   <xsd:sequence>
3628     <xsd:element
3629       name="PermittedLevel"
3630       type="ekmi:LevelClassificationType"
3631       minOccurs="0"
3632       maxOccurs="unbounded">
3633     </xsd:element>
3634     <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
3635   </xsd:sequence>
```



```
3636     <xsd:attribute ref="ekmi:any" use="required"/>
3637 </xsd:complexType>
```

3638 Schema Definition:

```
3639 <xsd:simpleType name="LevelClassificationType">
3640   <xsd:restriction base="xsd:string">
3641     <xsd:enumeration value="Unclassified"/>
3642     <xsd:enumeration value="Confidential"/>
3643     <xsd:enumeration value="Secret"/>
3644     <xsd:enumeration value="Top-Secret"/>
3645   </xsd:restriction>
3646 </xsd:simpleType>
```

3647 There SHALL be only one <PermittedLevels> element within the <Permissions> element. However,
3648 there MAY be an unbounded (unlimited) number of <PermittedLevel> elements within the
3649 <PermittedLevels> element. (Practically, it makes no sense to have more than the known levels;
3650 however, this specification leaves itself open to the possibility that other levels may be defined).

3651 The <PermittedLevels> element SHALL have one attribute named "any", that will have a "false" or
3652 "true" value, based on the following:

- 3653 • When the <PermittedLevels> element is null (i.e. it does not have a single <PermittedLevel>
3654 sub-element in it), the value of the "any" attribute SHALL be set to "true" AND the XML Schema
3655 Instance (XSI) "nil" attribute SHALL be set to "true".
- 3656 • When the <PermittedLevels> element is not-null (i.e. it has at least one <PermittedLevel>
3657 sub-element in it), the value of the "any" attribute SHALL be set to "false" AND the XML Schema
3658 Instance (XSI) "nil" attribute SHALL NOT be present.

3659 A null <PermittedLevels> element specifies that applications at ANY level are permitted use of the
3660 symmetric key, subject to complying with all other permission clauses in the <Permissions> element.

3661 The <PermittedLevel> sub-element, of the **LevelClassificationType**, identifies the precise MLS level
3662 at which the symmetric key in question may be used. The <PermittedLevel> SHALL contain one of the
3663 following four (4) enumerated values:

- 3664 1. Unclassified
- 3665 2. Confidential
- 3666 3. Secret
- 3667 4. Top-Secret

3668 Examples of the <PermittedLevels> element are shown below; other parts of their enclosing elements
3669 are not shown for brevity:

3670 **Example 1 – An example of a <PermittedLevels> element specifying that the symmetric key may**
3671 **be used only by applications at the Confidential level:**

```
3672 <ekmi:PermittedLevels ekmi:any="false">
3673   <ekmi:PermittedLevel>Confidential</ekmi:PermittedLevel>
3674 </ekmi:PermittedLevels>
```

3675 **Example 2 – An example of a <PermittedLevels> element specifying that the symmetric key may**
3676 **be used only by applications at the Secret or Top-Secret level:**

```
3677 <ekmi:PermittedLevels ekmi:any="false">
3678   <ekmi:PermittedLevel>Secret</ekmi:PermittedLevel>
```

```
3679     <ekmi:PermittedLevel>Top-Secret</ekmi:PermittedLevel>
3680 </ekmi:PermittedLevels>
```

3681 **Example 3 – An example of a null <PermittedLevels> element specifying that the symmetric key**
3682 **may be used at any level:**

```
3683     <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
```

3684 4.21 Element <PermittedLocations> and <PermittedLocation>

3685 The element <PermittedLocations>, of the type *PermittedLocationsType*, is used to define the
3686 geographically physical locations where applications are permitted use of a symmetric key. This
3687 element is useful only to applications that have the ability to determine the Global Positioning System
3688 (GPS) location of the client device intending to use the symmetric key.

3689 Schema Definition:

```
3690     <xsd:complexType name="PermittedLocationsType">
3691       <xsd:sequence>
3692         <xsd:element name="PermittedLocation" minOccurs="1" maxOccurs="unbounded">
3693           <xsd:complexType>
3694             <xsd:sequence>
3695               <xsd:element name="LocationName">
3696                 <xsd:simpleType>
3697                   <xsd:restriction base="xsd:string">
3698                     <xsd:maxLength value="256"/>
3699                     <xsd:whiteSpace value="preserve"/>
3700                   </xsd:restriction>
3701                 </xsd:simpleType>
3702               </xsd:element>
3703               <xsd:group
3704                 ref="ekmi:LocationCoordinateGroup"
3705                 minOccurs="0"
3706                 maxOccurs="unbounded"/>
3707               <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
3708             </xsd:sequence>
3709           </xsd:complexType>
3710         </xsd:element>
3711       </xsd:sequence>
3712     <xsd:attribute ref="ekmi:any" use="required"/>
3713   </xsd:complexType>
```

3714 Schema Definition:

```
3715     <xsd:group name="LocationCoordinateGroup">
3716       <xsd:sequence>
3717         <xsd:element name="Latitude">
3718           <xsd:simpleType>
3719             <xsd:restriction base="xsd:decimal">
3720               <xsd:totalDigits value="10"/>
3721               <xsd:fractionDigits value="7"/>
3722             </xsd:restriction>
3723           </xsd:simpleType>
3724         </xsd:element>
3725         <xsd:element name="Longitude">
3726           <xsd:simpleType>
3727             <xsd:restriction base="xsd:decimal">
3728               <xsd:totalDigits value="10"/>
3729               <xsd:fractionDigits value="7"/>
3730             </xsd:restriction>
3731           </xsd:simpleType>
3732         </xsd:element>
3733       </xsd:sequence>
3734     </xsd:group>
```

```

3730         </xsd:restriction>
3731     </xsd:simpleType>
3732 </xsd:element>
3733 </xsd:sequence>
3734 </xsd:group>

```

3735 There SHALL be only one <PermittedLocations> element within the <Permissions> element.
 3736 However, there MAY be an unbounded (unlimited) number of <PermittedLocation> sub-elements
 3737 within the <PermittedLocations> element.

3738 The <PermittedLocations> element SHALL have one attribute named “any”, that will have a “false” or
 3739 “true” value, based on the following:

- 3740 • When the <PermittedLocations> element is null (i.e. it does not have a single
 3741 <PermittedLocation> sub-element in it), the value of the “any” attribute SHALL be set to “true”
 3742 AND the XML Schema Instance (XSI) “nil” attribute SHALL be set to “true”.
- 3743 • When the <PermittedLocations> element is not-null (i.e. it has at least one
 3744 <PermittedLocation> sub-element in it), the value of the “any” attribute SHALL be set to
 3745 “false” AND the XML Schema Instance (XSI) “nil” attribute SHALL NOT be present.

3746 A null <PermittedLocations> element specifies that applications are permitted use of the symmetric
 3747 key at ANY physical location, subject to complying with all other permission clauses in the
 3748 <Permissions> element.

3749 The <PermittedLocation> element, of the **PermittedLocationType**, identifies the precise geographical
 3750 location where the symmetric key in question may be used. The <PermittedLocation> SHALL contain
 3751 the following elements:

- 3752 1. The <LocationName> element identifies a human-readable name of the physical location. It is
 3753 an XSD **String** type element, with a maximum length of 256 characters.

3754 There SHALL be only one <LocationName> element within a <PermittedLocation> element.
 3755

- 3756 2. An optional **LocationCoordinateGroup** which, when present, SHALL contain the following two
 3757 elements:

- 3758 a) The <Latitude> element of XSD **Decimal** type, that identifies the horizontal coordinate
 3759 location of the client device on the Earth, measured in *degrees* and expressed as a
 3760 decimal with the *minutes* and *seconds* part of the measurement expressed as a single
 3761 fraction.

3762 When used, there SHALL be only one <Latitude> element within the
 3763 <PermittedLocation> element.
 3764

- 3765 b) The <Longitude> element of XSD **Decimal** type, that identifies the vertical coordinate
 3766 location of the client device on the Earth, measured in *degrees* and expressed as a
 3767 decimal with the *minutes* and *seconds* part of the measurement expressed as a single
 3768 fraction.

3769 When used, there SHALL be only one <Longitude> element within the
 3770 <PermittedLocation> element.
 3771

3772 Some examples of the <PermittedLocations> element are shown below; other parts of their enclosing
 3773 elements are not shown for brevity:

3774 **Example 1 – An example of a <PermittedLocations> element specifying that the symmetric key**
 3775 **may be used only by applications at a single named location:**

```

3776     <ekmi:PermittedLocations ekmi:any="false">
3777         <ekmi:PermittedLocation>
3778             <ekmi:LocationName>StrongAuth Server Room</ekmi:LocationName>
3779         </ekmi:PermittedLocation>
3780     </ekmi:PermittedLocations>

```

3781 **Example 2 – An example of a <PermittedLocations> element specifying that the symmetric key**
3782 **may be used only by applications at a single location at the given GPS coordinates:**

```

3783     <ekmi:PermittedLocations ekmi:any="false">
3784         <ekmi:PermittedLocation>
3785             <ekmi:LocationName>StrongAuth Server Room</ekmi:LocationName>
3786             <ekmi:Latitude>37.385653 </ekmi:Latitude>
3787             <ekmi:Longitude>-121.993192 </ekmi:Longitude>
3788         </ekmi:PermittedLocation>
3789     </ekmi:PermittedLocations>

```

3790 **Example 3 – An example of a <PermittedLocations> element specifying that the symmetric key**
3791 **may be used only by applications at multiple locations:**

```

3792     <ekmi:PermittedLocations ekmi:any="false">
3793         <ekmi:PermittedLocation>
3794             <ekmi:LocationName>Humongous Headquarters</ekmi:LocationName>
3795         </ekmi:PermittedLocation>
3796         <ekmi:PermittedLocation>
3797             <ekmi:LocationName> Humongous Primary Data Center</ekmi:LocationName>
3798             <ekmi:Latitude>37.385653 </ekmi:Latitude>
3799             <ekmi:Longitude>-121.993192 </ekmi:Longitude>
3800         </ekmi:PermittedLocation>
3801         <ekmi:PermittedLocation>
3802             <ekmi:LocationName>Humongous DR Data Center</ekmi:LocationName>
3803             <ekmi:Latitude>68.845901 </ekmi:Latitude>
3804             <ekmi:Longitude>11.393385 </ekmi:Longitude>
3805         </ekmi:PermittedLocation>
3806     </ekmi:PermittedLocations>

```

3807 **Example 4 – An example of a null <PermittedLocations> element specifying that the symmetric**
3808 **key may be used at any location on the planet:**

```

3809     <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>

```

3810 4.22 Element <PermittedNumberOfTransactions>

3811 The element <PermittedNumberOfTransactions>, of type **PermittedNumberOfTransactionsType** is
3812 used to define the number of **encryption** transactions that applications are permitted with a symmetric
3813 key within a specific <Symkey> element, once the **SKCL** has started using the symmetric key in question.
3814 It does not limit the number of **decryption** transactions with the same symmetric key.

3815 Schema Definition:

```

3816     <xsd:complexType name="PermittedNumberOfTransactionsType">
3817         <xsd:simpleContent>
3818             <xsd:extension base="ekmi:NumberOfTransactionsType">
3819                 <xsd:attribute ref="ekmi:any" use="required"/>
3820             </xsd:extension>
3821         </xsd:simpleContent>
3822     </xsd:complexType>

```

3823 **Schema Definition:**

```
3824 <xsd:simpleType name="NumberOfTransactionsType">
3825   <xsd:restriction base="xsd:positiveInteger">
3826     <xsd:minInclusive value="1"/>
3827     <xsd:maxInclusive value="18446744073709551615"/>
3828   </xsd:restriction>
3829 </xsd:simpleType>
```

3830 There SHALL be only one <PermittedNumberOfTransactions> element within the <Permissions>
3831 element.

3832 The <PermittedNumberOfTransactions> element SHALL have one attribute named "any", that will
3833 have a "false" or "true" value, based on the following:

- 3834 • When the <PermittedNumberOfTransactions> element is null (i.e. it does not have any content
3835 in it), the value of the "any" attribute SHALL be set to "true" AND the XML Schema Instance
3836 (XSI) "nil" attribute SHALL be set to "true".
- 3837 • When the <PermittedNumberOfTransactions> element is not-null (i.e. it has a positive integer
3838 content in it), the value of the "any" attribute SHALL be set to "false" AND the XML Schema
3839 Instance (XSI) "nil" attribute SHALL NOT be present.

3840 A null <PermittedNumberOfTransactions> element specifies that applications are permitted use of the
3841 symmetric key for an unlimited number of encryption transactions, subject to complying with all other
3842 permission clauses in the <Permissions> element.

3843 The value of <PermittedNumberOfTransactions> element, of the XSD *positiveInteger* type, MAY
3844 range between 1 and 18446744073709551615.

3845 Some examples of the <PermittedNumberOfTransactions> element are shown below; other parts of
3846 their enclosing elements are not shown for brevity:

3847 **Example 1 – An example of a <PermittedNumberOfTransactions> element specifying that the**
3848 **symmetric key may be used only for a single encryption transaction by an SKCL:**

```
3849 <ekmi:PermittedNumberOfTransactions ekmi:any="false">
3850   1
3851 </ekmi:PermittedNumberOfTransactions>
```

3852 **Example 2 – An example of a <PermittedNumberOfTransactions> element specifying that the**
3853 **symmetric key may be used only for 100 transactions by an SKCL:**

```
3854 <ekmi:PermittedNumberOfTransactions ekmi:any="false">
3855   100
3856 </ekmi:PermittedNumberOfTransactions>
```

3857 **Example 3 – An example of a null <PermittedNumberOfTransactions> element specifying that the**
3858 **symmetric key may be used for an unlimited number of encryption transactions by an SKCL:**

```
3859 <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
```

3860 **4.23 Element <PermittedTimes> and <PermittedTime>**

3861 The element <PermittedTimes>, of the type *PermittedTimesType* and its only child-element
3862 <PermittedTime>, which is an anonymous XSD *ComplexType*, are used to define sets of times during
3863 the day between which applications are permitted to use a symmetric key within a specific <Symkey>
3864 element.

Schema Definition:

```
<xsd:complexType name="PermittedTimesType">
  <xsd:sequence>
    <xsd:element name="PermittedTime" minOccurs="0" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="StartTime">
            <xsd:simpleType>
              <xsd:restriction base="xsd:time">
                <xsd:pattern value="\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="EndTime">
            <xsd:simpleType>
              <xsd:restriction base="xsd:time">
                <xsd:pattern value="\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute ref="ekmi:any" use="required"/>
</xsd:complexType>
```

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

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

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

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

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

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

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

3915 Conforming **SKCL** implementations SHALL NOT start using the symmetric before the onset of
3916 the <StartTime> on the client machine.

3917 2. The <EndTime> element identifies the time until which applications may use the symmetric key
3918 in question. It is an XSD *Time* type that MUST be specified in a specific pattern (see examples)
3919 where the first two digits specify the hour, the second two digits specify the minutes and the last
3920 two digits specify the seconds in a 24 hour format.

3921
3922 There SHALL be only one <EndTime> element within a <PermittedTime> element.

3923
3924 Conforming **SKCL** implementations SHALL NOT use the symmetric after the end of the
3925 <EndTime> on the client machine.

3926 Some examples of the <PermittedTimes> element are shown below; other parts of their enclosing
3927 elements are not shown for brevity:

3928 **Example 1 – An example of a <PermittedTimes> element with a single<PermittedTime> element.**
3929 **The <StartTime> specifies 9:00AM on the client machine while the <EndTime> specifies 5:00PM:**

```
3930 <ekmi:PermittedTimes ekmi:any="false">  
3931   <ekmi:PermittedTime>  
3932     <ekmi:StartTime>09:00:00</ekmi:StartTime>  
3933     <ekmi:EndTime>17:00:00</ekmi:EndTime>  
3934   </ekmi:PermittedTime>  
3935 </ekmi:PermittedTimes>
```

3936 **Example 2 – An example of a <PermittedTimes> element with two <PermittedTime> elements. For**
3937 **the first <PermittedTime> element , the <StartTime> element specifies 6:00AM while the**
3938 **<EndTime> element specifies 12:00 Noon. For the second <PermittedTime> element, the**
3939 **<StartTime> element specifies 3:00 PM in the afternoon, while the <EndTime> element specifies**
3940 **7:00PM in the evening. This policy might imply that a symmetric key with this <PermittedTimes>**
3941 **element cannot be used during a lunch break of 12:00 Noon to 3:00PM:**

```
3942 <ekmi:PermittedTimes ekmi:any="false">  
3943   <ekmi:PermittedTime>  
3944     <ekmi:StartTime>06:00:00</ekmi:StartTime>  
3945     <ekmi:EndTime>12:00:00</ekmi:EndTime>  
3946   </ekmi:PermittedTime>  
3947   <ekmi:PermittedTime>  
3948     <ekmi:StartTime>15:00:00</ekmi:StartTime>  
3949     <ekmi:EndTime>19:00:00</ekmi:EndTime>  
3950   </ekmi:PermittedTime>  
3951 </ekmi:PermittedTimes>
```

3952 **Example 3 – An example of a null <PermittedTimes> element, specifying that the key may be**
3953 **used at any time:**

```
3954 <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
```

3955 4.24 Element <PermittedUses> and <PermittedUse>

3956 The element <PermittedUses>, of the type *PermittedUsesType*, is used to define the specific ways in
3957 which applications are permitted to use a symmetric key within a specific <Symkey> element.

3958 Schema Definition:


```

3959     <xsd:complexType name="PermittedUsesType" mixed="true">
3960       <xsd:sequence>
3961         <xsd:element name="PermittedUse" minOccurs="0" maxOccurs="unbounded">
3962           <xsd:simpleType>
3963             <xsd:restriction base="xsd:string">
3964               <xsd:maxLength value="256"/>
3965               <xsd:whiteSpace value="preserve"/>
3966             </xsd:restriction>
3967           </xsd:simpleType>
3968         </xsd:element>
3969         <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
3970       </xsd:sequence>
3971       <xsd:attribute ref="ekmi:any" use="required"/>
3972     </xsd:complexType>

```

3973 There SHALL be only one <PermittedUses> element within the <Permissions> element. However,
 3974 there MAY be an unbounded (unlimited) number of <PermittedUse> sub-elements within the
 3975 <PermittedUses> element.

3976 The <PermittedUses> element SHALL have one attribute named “any”, that will have a “false” or “true”
 3977 value, based on the following:

- 3978 • When the <PermittedUses> element is null (i.e. it does not have a single <PermittedUse> sub-
 3979 element in it), the value of the “any” attribute SHALL be set to “true” AND the XML Schema
 3980 Instance (XSI) “nil” attribute SHALL be set to “true”.
- 3981 • When the <PermittedUses> element is not-null (i.e. it has at least one <PermittedUse> sub-
 3982 element in it), the value of the “any” attribute SHALL be set to “false” AND the XML Schema
 3983 Instance (XSI) “nil” attribute SHALL NOT be present.

3984 A null <PermittedUses> element specifies that applications are permitted use of the symmetric key for
 3985 ANY purpose, subject to complying with all other permission clauses in the <Permissions> element.

3986 Examples of the <PermittedUses> element are shown below; other parts of their enclosing elements are
 3987 not shown for brevity:

3988 **Example 1 – An example of a <PermittedUses> element specifying that the symmetric key may be**
 3989 **used only by VPN applications for session encryption keys:**

```

3990     <ekmi:PermittedUses ekmi:any="false">
3991       <ekmi:PermittedUse>VPN</ekmi:PermittedUse>
3992     </ekmi:PermittedUses>

```

3993 **Example 2 – An example of a <PermittedUses> element specifying that the symmetric key may be**
 3994 **used only by applications on laptops and Personal Digital Assistants (PDA):**

```

3995     <ekmi:PermittedUses ekmi:any="false">
3996       <ekmi:PermittedUse>Laptop</ekmi:PermittedUse>
3997       <ekmi:PermittedUse>PDA</ekmi:PermittedUse>
3998     </ekmi:PermittedUses>

```

3999 **Example 3 – An example of a null <PermittedUses> element specifying that the symmetric key**
 4000 **may be used for any purpose:**

```

4001     <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>

```

4.25 Element <KeyCachePolicyRequest>

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

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

Schema Definition:

```
<xsd:element name="KeyCachePolicyRequest">
  <xsd:complexType>
    <xsd:annotation>
      <xsd:documentation>
        No elements/attributes are defined for KeyCachePolicyRequest.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:complexType>
</xsd:element>
```

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

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

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

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

4.26 Element <KeyCachePolicyResponse>

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

Schema Definition:

```
<xsd:element name="KeyCachePolicyResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element
        name="KeyCachePolicy"
        type="ekmi:KeyCachePolicyType"
        minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

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

4.27 Element <KeyCachePolicy>

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

Schema Definition:

```
<xsd:element name="KeyCachePolicyResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element
        name="KeyCachePolicy"
        type="ekmi:KeyCachePolicyType"
        minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="KeyCachePolicyType" mixed="true">
  <xsd:sequence>
    <xsd:element name="KeyCachePolicyID" type="ekmi:TwoPartIDType"/>
    <xsd:element name="PolicyName">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="255"/>
          <xsd:whiteSpace value="preserve"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Description" nillable="true">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="2048"/>
          <xsd:whiteSpace value="preserve"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="KeyClass" type="ekmi:KeyClassType"/>
    <xsd:element name="StartDate" type="xsd:dateTime"/>
    <xsd:element name="EndDate" type="xsd:dateTime" nillable="true"/>
    <xsd:element name="PolicyCheckInterval">
      <xsd:simpleType>
        <xsd:restriction base="xsd:nonNegativeInteger">
          <xsd:minInclusive value="0"/>
          <xsd:maxInclusive value="2592000"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Status" type="ekmi:StatusType"/>
    <xsd:element
      name="NewKeysCacheDetail"
      type="ekmi:KeyCacheDetailType"
      minOccurs="0"/>
    <xsd:element
      name="UsedKeysCacheDetail"
      type="ekmi:KeyCacheDetailType"
      minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

4102 The <KeyCachePolicy> element is of the **KeyCachePolicyType** and consists of the following child
4103 elements:

- 4104 1. <KeyCachePolicyID> [Required]
4105
4106 The <KeyCachePolicyID> element, of type **TwoPartIDType**, identifies the unique policy object
4107 within the **SKMS**. There SHALL be only one <KeyCachePolicyID> element within a
4108 <KeyCachePolicy> element.
4109
4110 The **TwoPartIDType** is specified in Section 4.11.
- 4111 2. <PolicyName> [Required]
4112
4113 The <PolicyName> element, of type XSD **String**, with a maximum length of 255 characters,
4114 identifies a unique name given to this <KeyCachePolicy>. There SHALL be only one
4115 <PolicyName> element within a <KeyCachePolicy> element.
- 4116 3. <Description> [Required]
4117
4118 The <Description> element, of type XSD **String**, with a maximum length of 2048 characters,
4119 provides a human-readable description of this policy. There SHALL be only one
4120 <Description> element within a <KeyCachePolicy> element.
4121
4122 The <Description> **MAY be an empty element, but MUST exist within the <KeyCachePolicy>**
4123 **element.**
- 4124 4. <KeyClass> [Required]
4125
4126 This element of type **KeyClassType** identifies the key-class of the symmetric key to which this
4127 policy applies.
- 4128 5. <StartDate> [Required]
4129
4130 The <StartDate> element, of type XSD **dateTime**, specifies the date and time at which this
4131 policy becomes effective. There SHALL be only one <StartDate> element within a
4132 <KeyCachePolicy> element.
- 4133 6. <EndDate> [Required]
4134
4135 The <EndDate> element, of type XSD **dateTime**, specifies the date and time at which this policy
4136 expires. There SHALL be only one <EndDate> element within a <KeyCachePolicy> element.
4137
4138 The <EndDate> **MAY be an empty element, but MUST exist within the <KeyCachePolicy>**
4139 **element.**
- 4140 7. <PolicyCheckInterval> [Required]
4141
4142 **The <PolicyCheckInterval> element, of type XSD nonNegativeInteger, specifies the**
4143 **frequency at which the client SHALL check the SKS server for updates to this policy. This**
4144 **frequency is specified in seconds and SHALL NOT exceed 2592000 seconds (30 calendar**
4145 **days). There SHALL be only one <PolicyCheckInterval> element within a**
4146 **<KeyCachePolicy> element.**
- 4147 8. <Status> [Required]
4148
4149 The <Status> element, of type **StatusType**, identifies the current status of this policy within the
4150 SKMS. There SHALL be only one <Status> element within a <KeyCachePolicy> element.

The **StatusType** is specified in Section 4.14.

9. <NewKeysCacheDetail> [Required]

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

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

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

The **KeyCacheDetailType** is specified in Section 4.28.

10. <UsedKeysCacheDetail> [Required]

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

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

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

The **KeyCacheDetailType** is specified in Section 4.28.

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

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

```
<ekmi:KeyCachePolicy>
  <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
  <ekmi:PolicyName>
    Corporate Laptop Symmetric Key Caching Policy
  </ekmi:PolicyName>
  <ekmi:Description>
    This policy defines how company-issued laptops will manage
    symmetric keys used for file/disk encryption in each laptop's
    local cache. This policy must be used by all laptops that
    use the company EKMI.
  </ekmi:Description>
  <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
  <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
  <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
  <ekmi:Status>Active</ekmi:Status>
  <ekmi:NewKeysCacheDetail>
    <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
    <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
  </ekmi:NewKeysCacheDetail>
```

```

4204     <ekmi:UsedKeysCacheDetail>
4205         <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
4206         <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
4207     </ekmi:UsedKeysCacheDetail>
4208 </ekmi:KeyCachePolicy>

```

4209 **Example 2 – A <KeyCachePolicy> that is effective starting January 01, 2008 and never expires. It**
4210 **does NOT permit any caching of symmetric keys through the absence of the detail elements on**
4211 **caching:**

```

4212 <ekmi:KeyCachePolicy>
4213     <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
4214     <ekmi:PolicyName>
4215         No Caching Policy
4216     </ekmi:PolicyName>
4217     <ekmi:Description>
4218         This policy is for high-risk, always-connected machines on the
4219         network, which will never cache symmetric keys locally. This
4220         policy never expires (but checks monthly for any updates).
4221     </ekmi:Description>
4222     <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
4223     <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
4224     <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
4225     <ekmi:Status>Active</ekmi:Status>
4226 </ekmi:KeyCachePolicy>

```

4227 4.28 Type *KeyCacheDetailType*

4228 The ***KeyCacheDetailType*** type allows **SKS** servers to specify precisely how many symmetric keys
4229 MAY be cached on the client machine, and for how long.

4230 Schema Definition:

```

4231 <xsd:complexType name="KeyCacheDetailType">
4232     <xsd:sequence>
4233         <xsd:element name="MaximumKeys" minOccurs="1">
4234             <xsd:simpleType>
4235                 <xsd:restriction base="xsd:integer">
4236                     <xsd:minInclusive value="0"/>
4237                     <xsd:maxInclusive value="18446744073709551615"/>
4238                 </xsd:restriction>
4239             </xsd:simpleType>
4240         </xsd:element>
4241         <xsd:element name="MaximumDuration" minOccurs="1">
4242             <xsd:simpleType>
4243                 <xsd:restriction base="xsd:integer">
4244                     <xsd:minInclusive value="0"/>
4245                     <xsd:maxInclusive value="18446744073709551615"/>
4246                 </xsd:restriction>
4247             </xsd:simpleType>
4248         </xsd:element>
4249     </xsd:sequence>
4250 </xsd:complexType>

```

4251 The ***KeyCacheDetailType*** consists of the following child elements:

- 4252 1. <MaximumKeys> [Required]

4254 The <MaximumKeys> element, of type XSD ***Integer***, specifies the maximum number of symmetric

4255 keys that MAY be cached on a client machine. It SHALL be a positive number between the
4256 values 0 and 18446744073709551615. There SHALL be only one <MaximumKeys> element
4257 within an element that uses the **KeyCacheDetailType**.

4258 2. <MaximumDuration> [Required]

4259
4260 The <MaximumDuration> element, of type XSD **Integer**, specifies the maximum number of
4261 seconds that symmetric keys MAY be cached on a client machine. It SHALL be a positive
4262 number between the values 0 and 18446744073709551615. There SHALL be only one
4263 <MaximumDuration> element within an element that uses the **KeyCacheDetailType**.

4264 Examples of the **KeyCacheDetailType** when used in the <KeyCachePolicy> element are as follows.

4265 **Example 1 – A <KeyCachePolicy> that is valid between January 01, 2008 and December 31, 2008.**
4266 **It requires the client to check for policy updates every day and allows 3 new and 3 used keys to**
4267 **be cached for up to 90 days:**

```
4268 <ekmi:KeyCachePolicy>
4269   <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
4270   <ekmi:PolicyName>
4271     Corporate Laptop Symmetric Key Caching Policy
4272   </ekmi:PolicyName>
4273   <ekmi:Description>
4274     This policy defines how company-issued laptops will manage
4275     symmetric keys used for file/disk encryption in their local
4276     cache. This policy must be used by all laptops that use
4277     the company EKMI.
4278   </ekmi:Description>
4279   <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
4280   <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
4281   <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
4282   <ekmi>Status>Active</ekmi>Status>
4283   <ekmi:NewKeysCacheDetail>
4284     <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
4285     <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
4286   </ekmi:NewKeysCacheDetail>
4287   <ekmi:UsedKeysCacheDetail>
4288     <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
4289     <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
4290   </ekmi:UsedKeysCacheDetail>
4291 </ekmi:KeyCachePolicy>
```

4292 **Example 1 – A <KeyCachePolicy> that is valid between January 01, 2008 and December 31, 2008.**
4293 **It requires the client to check for policy updates every day and allows 1 new and 0 used keys to**
4294 **be cached for upto 15 days:**

```
4295 <ekmi:KeyCachePolicy>
4296   <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
4297   <ekmi:PolicyName>
4298     Corporate Laptop Symmetric Key Caching Policy
4299   </ekmi:PolicyName>
4300   <ekmi:Description>
4301     This policy defines how company-issued laptops will manage
4302     symmetric keys used for file/disk encryption in each laptop's
4303     local cache. This policy must be used by all laptops that
4304     use the company EKMI.
4305   </ekmi:Description>
4306   <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
4307   <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
```



```

4308     <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
4309     <ekmi:Status>Active</ekmi:Status>
4310     <ekmi:NewKeysCacheDetail>
4311         <ekmi:MaximumKeys>1</ekmi:MaximumKeys>
4312         <ekmi:MaximumDuration>1296000</ekmi:MaximumDuration>
4313     </ekmi:NewKeysCacheDetail>
4314     <ekmi:UsedKeysCacheDetail>
4315         <ekmi:MaximumKeys>0</ekmi:MaximumKeys>
4316         <ekmi:MaximumDuration>1296000</ekmi:MaximumDuration>
4317     </ekmi:UsedKeysCacheDetail>
4318 </ekmi:KeyCachePolicy>

```

4319 4.29 Use of Web Services Security (WSS)

4320 While it has been mentioned earlier in this specification, it is explicitly noted here that conforming
4321 implementations of the SKSML protocol MUST enclose all SKSML messages between participants in an
4322 SKMS in the **SOAP Body** of a **SOAP Envelope**. Additionally, the contents of the **SOAP Body** MUST be
4323 secured using digital signatures conforming to [XMLSignature] in the **SOAP Header**. Specifically, the
4324 **SOAP Header** of the **SOAP Envelope** MUST enclose a **Security** element conforming to [WSS] with a
4325 **ValueType** attribute containing the value [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3)
4326 [x509-token-profile-1.0#X509v3](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3). The **Security** element must conform to all other requirements of the
4327 specified security profile in [WSS] to form a well-formed, secure message.

4328 While the payload (the symmetric-key in the <Symkey> element) is already encrypted by the SKS server
4329 using the Public Key of the requesting client, the contents of the **SOAP Body** MAY, additionally, be
4330 secured with encryption using [XMLEncryption], if desired. The choice of using the requesting client's
4331 Public Key or a SOAP-layer Public Key to encrypt the **SOAP Body** is left to the implementers of the
4332 SKMS.

4333 The SOAP security objects must be layered on the SKSML messages **after** the SKSML message has
4334 been fully constructed (and secured in the case of the symmetric-key payload) by the SKS server.

4335 *Note: In the event that SKMS sites choose to encrypt the SOAP Body's contents at the SOAP layer (in*
4336 *addition to the symmetric key encryption performed at the SKSML layer), the **Security** element in some*
4337 *implementations of [WSS] will have two <BinarySecurityToken> elements with digital certificates in them:*
4338 *one for the sender's signing certificate, and the other for the recipient's encryption certificate. While*
4339 *SKMS sites have the choice of encrypting the contents of the SOAP Body, they do NOT have a choice*
4340 *about signing the contents of the SOAP Body - all SKSML messages enclosed in the SOAP Body MUST*
4341 *be signed by clients and servers within an SKMS.*

4342 4.30 Use of SKMS Error Codes & Messages

4343 SKSML defines many messages between the Symmetric Key Client Library (SKCL) implementations and
4344 Symmetric Key Services (SKS) servers. Given the complexity of layered-technology, there are ample
4345 opportunities for components to fail during processing.

4346 The EKMI TC believes that adoption of this technology can be encouraged by standardizing codes for
4347 errors and informational messages used by SKMS clients and servers. At the same time, the TC
4348 recognizes the value of providing flexibility for vendor implementations to innovate beyond what the TC
4349 has envisioned.

4350 To ensure a baseline level of conformance, the TC has chosen to standardize some common error-
4351 codes and messages, while creating a process for vendors to request and receive a block of codes
4352 reserved for the vendor's use within their implementation of SKSML. These codes are defined in
4353 **Appendix C** and the process is described in **Appendix D** of this specification.

4354 Conforming implementations of SKSML MUST implement the Standard Error Codes and Message within
4355 their implementations. It is left up to the vendor to determine if they wish to apply for, and receive, a
4356 reserved block of codes for their own implementation's unique codes and messages.

5 Conformance

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

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

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

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

Appendix A. Acknowledgments

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

Participants:

- Ezer Farhi, Associate
- Benjamin Tomhave, Btplc
- Tim Bruce, CA*
- June Leung, Associate
- Shaheen N Abdul Jabbar, Individual
- Ken Adler, Individual
- Stefan Drees*. Individual
- Marc Massar, Individual
- Michael Nelson, Individual
- Davi Ottenheimer, Individual
- Allen Schaaf, Individual
- Harry Haury, NuParadigm Government Systems, Inc.
- Tomas Gustavsson, PrimeKey Solutions AB
- Anil Saldhana, Red Hat
- Arshad Noor, Associate
- Sandi Roddy, US Department of Defense (DoD)
- Thomas Hardjono, Associate
- Upendra Mardikar, Associate
- Eric Lengvenis, Wells Fargo

Appendix B. Revision History

Version	Date	Author	Notes
DRAFT 4	June 08, 2008	Arshad Noor	Initial version
DRAFT 5	June 17, 2008	Arshad Noor	- Moved non-normative sections to their own document. - KeyClass element was added to KeyCachePolicy. - KeyCachePolicy is now embedded inside a KeyCachePolicyResponse.
DRAFT 6	July 7, 2008	Arshad Noor	- Modified Permissions object to include all sub-elements on a mandatory basis. - Modified all abbreviations in elements to expand to full names.
PR 1	July 22, 2008	Arshad Noor	- Modified Title page information to conform with OASIS standards. - Brought some Background information into this document from the information document, into Section 2. - Added a Conformance section to Section 4.
PR 2	11/04/08	Arshad Noor	- Added SymkeyWorkInProgress, SymkeyRequestID and RequestCheckInterval elements to support sending and receiving request/responses asynchronously. - Modified SymkeyResponse to include SymkeyWorkInProgress as a valid response to a key request. - Modified Symkey to include the SymkeyRequestID to support asynchronous request/responses. - Modified SymkeyError to include the SymkeyRequestID to support asynchronous request/responses. - Modified SymkeyRequest to send a SymkeyRequestID to poll the SKS server on the status of a symmetric-key request.. - Added Standard Error Codes & Messages in Appendix C. - Added Appendix D to define the process for vendors to apply for a reserved block of codes.
PR 2 (2.1)	11/11/08	Arshad Noor	Typographical corrections in Sections 3.3, 3.8 and 3.11
PR 2 (2.2)	11/18/08	Arshad Noor	- Added X509EncryptionCertificateType and X509EncryptionCertificate element to support sending a PKI X509-compliant digital certificate from the client to the server for encrypting the symmetric-key payload in the response

Appendix C. SKMS Error Codes and Messages

The OASIS EKMI TC has determined that it is useful to standardize on the structure and content of error and informational messages within SKSML so that implementations and their users are clear about the problem they are dealing within EKMI.

Structure

The following structure is proposed for Symmetric Key Management System (SKMS) messages. The message will consist of a three-part string, with each part separated from the others by a hyphen (“-”). The 3-part message consists of the following:

- The first part is the fixed string “**SKMS**”
- The second part is a fixed string consisting of one of the following choices:
 - **ERR**
 - **MSG**
- The third, and last, part is a 5-digit integer identifying the message

Thus, an SKMS error message might look like the following: **SKMS-ERR-NNNNN**; and an SKMS informational message might look like the following: **SKMS-MSG-NNNNN**

Five-digit Codes

The 5-digit integer is divided into the following groups to ensure consistency amongst implementations:

- 00001 – 10000 Reserved for OASIS EKMI TC use (as described below);
- 10001 – 99999 Reserved for vendor implementations of SKSML on a first-come, first-served basis (the process is described below);

SKMS Standard Code-ranges

The 5-digit code range reserved for OASIS EKMI TC SKMS standards use will be reserved as follows:

Code-range	Reserved for
00001 - 00099	Authentication related errors and messages
00100 - 00199	Authorization related errors and messages
00200 - 00299	Cryptographic-module related errors and messages
00300 - 00399	Key-cache and KeyCachePolicy related errors and messages
00400 - 00499	Key-usage and KeyUsePolicy related errors and messages
00500 - 00599	Symmetric Key Client Library related errors and messages
00600 - 00699	Symmetric Key Services server related errors and messages
00700 - 00799	Request checking related errors and messages
00800 - 00899	Miscellaneous errors and messages

Code-range	Reserved for
00900 - 10000	Future OASIS EKMI TC use

4423

4424 *Note: The {0} symbol at the end of each message is a placeholder for a parameter that can be used by*
4425 *implementations for adding additional information pertaining to the error. The additional information will*
4426 *be useful to administrators and software developers in helping them focus on the part of the system*
4427 *where the underlying problem has manifested itself.*

4428 Authentication ERROR Codes (00001 - 00099)

Code	Message
SKMS-ERR-00001	Authentication failure – invalid signature: {0}
SKMS-ERR-00002	Authentication failure – invalid status: {0}
SKMS-ERR-00003	Authentication failure – unverifiable certificate: {0}
SKMS-ERR-00004	Authentication failure – expired certificate: {0}
SKMS-ERR-00005	Authentication failure – revoked certificate: {0}
SKMS-ERR-00006	Authentication failure – revoked certificate issuer: {0}
SKMS-ERR-00007	Authentication failure – missing certificate: {0}
SKMS-ERR-00008	Authentication failure – missing certificate keyUsage: {0}
SKMS-ERR-00009	Authentication failure – missing certificate crlDistributionPoint: {0}
SKMS-ERR-00010	Authentication failure – missing certificate authorityInfoAccess: {0}
SKMS-ERR-00011	Authentication failure – invalid certificate Subject DN: {0}
SKMS-ERR-00012	Authentication failure – invalid certificate Validity: {0}
SKMS-ERR-00013	Authentication failure – invalid certificate keyUsage: {0}
SKMS-ERR-00014	Authentication failure – invalid certificate crlDistributionPoint: {0}
SKMS-ERR-00015	Authentication failure – invalid certificate authorityInfoAccess: {0}
SKMS-ERR-00016	Authentication failure – unreachable certificate crlDistributionPoint: {0}
SKMS-ERR-00017	Authentication failure – unreachable certificate authorityInfoAccess: {0}
SKMS-ERR-00099	Authentication failure – other authentication error: {0}

4429

4430 Authorization ERROR Codes (00100 - 00199)

Code	Message
SKMS-ERR-00101	Authorization failure – invalid signature: {0}
SKMS-ERR-00102	Authorization failure – invalid status: {0}

Code	Message
SKMS-ERR-00103	Authorization failure – invalid requester: {0}
SKMS-ERR-00104	Authorization failure – invalid request: {0}
SKMS-ERR-00105	Authorization failure – invalid identifier: {0}
SKMS-ERR-00106	Authorization failure – invalid keyclass: {0}
SKMS-ERR-00107	Authorization failure – invalid policy: {0}
SKMS-ERR-00108	Authorization failure – invalid role: {0}
SKMS-ERR-00109	Authorization failure – invalid application: {0}
SKMS-ERR-00110	Authorization failure – invalid date: {0}
SKMS-ERR-00111	Authorization failure – invalid day: {0}
SKMS-ERR-00112	Authorization failure – invalid duration: {0}
SKMS-ERR-00113	Authorization failure – invalid level: {0}
SKMS-ERR-00114	Authorization failure – invalid location: {0}
SKMS-ERR-00115	Authorization failure – invalid number of transactions: {0}
SKMS-ERR-00116	Authorization failure – invalid time: {0}
SKMS-ERR-00117	Authorization failure – invalid use: {0}
SKMS-ERR-00118	Authorization failure – unauthorized access: {0}
SKMS-ERR-00119	Authorization failure – unauthorized application: {0}
SKMS-ERR-00120	Authorization failure – unauthorized date: {0}
SKMS-ERR-00121	Authorization failure – unauthorized day: {0}
SKMS-ERR-00122	Authorization failure – unauthorized duration: {0}
SKMS-ERR-00123	Authorization failure – unauthorized level: {0}
SKMS-ERR-00124	Authorization failure – unauthorized location: {0}
SKMS-ERR-00125	Authorization failure – unauthorized number of transactions: {0}
SKMS-ERR-00126	Authorization failure – unauthorized time: {0}
SKMS-ERR-00127	Authorization failure – unauthorized use: {0}
SKMS-ERR-00118	Authorization failure – other authorization error: {0}

4431

4432 Cryptographic-module ERROR Codes (00200 - 00299)

Code	Message
SKMS-ERR-00201	Cryptographic-module failure – invalid signature: {0}
SKMS-ERR-00202	Cryptographic-module failure – invalid status: {0}

Code	Message
SKMS-ERR-00203	Cryptographic-module failure – invalid cryptographic provider: {0}
SKMS-ERR-00204	Cryptographic-module failure – invalid algorithm: {0}
SKMS-ERR-00205	Cryptographic-module failure – invalid initialization vector: {0}
SKMS-ERR-00206	Cryptographic-module failure – invalid padding: {0}
SKMS-ERR-00207	Cryptographic-module failure – invalid key-size: {0}
SKMS-ERR-00208	Cryptographic-module failure – invalid password or PIN: {0}
SKMS-ERR-00209	Cryptographic-module failure – missing cryptographic provider: {0}
SKMS-ERR-00210	Cryptographic-module failure – missing cryptographic module: {0}
SKMS-ERR-00211	Cryptographic-module failure – missing password or PIN: {0}
SKMS-ERR-00212	Cryptographic-module failure – missing private key: {0}
SKMS-ERR-00213	Cryptographic-module failure – missing digital certificate: {0}
SKMS-ERR-00214	Cryptographic-module failure – missing certificate chain: {0}
SKMS-ERR-00215	Cryptographic-module failure – failed to sign: {0}
SKMS-ERR-00216	Cryptographic-module failure – failed to verify: {0}
SKMS-ERR-00217	Cryptographic-module failure – failed to encrypt: {0}
SKMS-ERR-00218	Cryptographic-module failure – failed to decrypt: {0}
SKMS-ERR-00219	Cryptographic-module failure – failed to digest (hash): {0}
SKMS-ERR-00220	Cryptographic-module failure – failed to generate key: {0}
SKMS-ERR-00299	Cryptographic-module failure – other cryptographic-module error: {0}

4433

4434

Key-cache and KeyCachePolicy ERROR Codes (00300 - 00399)

Code	Message
SKMS-ERR-00301	Key-cache failure – invalid signature: {0}
SKMS-ERR-00302	Key-cache failure – invalid status: {0}
SKMS-ERR-00303	Key-cache failure – invalid cache: {0}
SKMS-ERR-00304	Key-cache failure – invalid policy: {0}
SKMS-ERR-00305	Key-cache failure – missing policy: {0}
SKMS-ERR-00306	Key-cache failure – expired policy: {0}
SKMS-ERR-00307	Key-cache failure – unauthorized create-access to cache: {0}
SKMS-ERR-00308	Key-cache failure – unauthorized read-access to cache: {0}

Code	Message
SKMS-ERR-00309	Key-cache failure – unauthorized write-access to cache: {0}
SKMS-ERR-00399	Key-cache failure – other key-cache error: {0}

4435

4436 Key-usage and KeyUsePolicy ERROR Codes (00400 - 00499)

Code	Message
SKMS-ERR-00401	Key-usage failure – invalid signature: {0}
SKMS-ERR-00402	Key-usage failure – invalid status: {0}
SKMS-ERR-00403	Key-usage failure – invalid policy: {0}
SKMS-ERR-00404	Key-usage failure – invalid application: {0}
SKMS-ERR-00405	Key-usage failure - expired policy: {0}
SKMS-ERR-00406	Key-usage failure – missing policy: {0}
SKMS-ERR-00407	Key-usage failure - missing library: {0}
SKMS-ERR-00499	Key-usage failure – other key-usage error: {0}

4437

4438 Symmetric Key Client Library ERROR Codes (00500 - 00599)

Code	Message
SKMS-ERR-00501	SKCL error - invalid signature: {0}
SKMS-ERR-00502	SKCL error - invalid status: {0}
SKMS-ERR-00503	SKCL error - invalid parameter: {0}
SKMS-ERR-00504	SKCL error - invalid domain ID: {0}
SKMS-ERR-00505	SKCL error - invalid server ID: {0}
SKMS-ERR-00506	SKCL error - invalid key ID: {0}
SKMS-ERR-00507	SKCL error - invalid request ID: {0}
SKMS-ERR-00508	SKCL error - invalid key-class: {0}
SKMS-ERR-00509	SKCL error - invalid server URL: {0}
SKMS-ERR-00510	SKCL error - invalid plaintext: {0}
SKMS-ERR-00511	SKCL error - invalid ciphertext: {0}
SKMS-ERR-00512	SKCL error - missing parameter: {0}
SKMS-ERR-00513	SKCL error - missing domain ID: {0}
SKMS-ERR-00514	SKCL error - missing server ID: {0}

Code	Message
SKMS-ERR-00515	SKCL error - missing key ID: {0}
SKMS-ERR-00516	SKCL error - missing request ID: {0}
SKMS-ERR-00517	SKCL error - missing key-class: {0}
SKMS-ERR-00518	SKCL error - missing server URL: {0}
SKMS-ERR-00519	SKCL error - missing plaintext: {0}
SKMS-ERR-00520	SKCL error - missing ciphertext: {0}
SKMS-ERR-00521	SKCL error – SOAP layer error: {0}
SKMS-ERR-00522	SKCL error - network layer error: {0}
SKMS-ERR-00523	SKCL error – database layer error: {0}
SKMS-ERR-00599	SKCL error - other SKCL error: {0}

4439

4440 Symmetric Key Services server ERROR Codes (00600 - 00699)

Code	Message
SKMS-ERR-00601	SKS error - invalid signature: {0}
SKMS-ERR-00602	SKS error - invalid status: {0}
SKMS-ERR-00603	SKS error - invalid parameter: {0}
SKMS-ERR-00604	SKS error - invalid domain ID: {0}
SKMS-ERR-00605	SKS error - invalid server ID: {0}
SKMS-ERR-00606	SKS error - invalid key ID: {0}
SKMS-ERR-00607	SKS error - invalid request ID: {0}
SKMS-ERR-00608	SKS error - invalid key-class: {0}
SKMS-ERR-00609	SKS error - invalid server URL: {0}
SKMS-ERR-00610	SKS error - invalid plaintext: {0}
SKMS-ERR-00611	SKS error - invalid ciphertext: {0}
SKMS-ERR-00612	SKS error - missing parameter: {0}
SKMS-ERR-00613	SKS error - missing domain ID: {0}
SKMS-ERR-00614	SKS error - missing server ID: {0}
SKMS-ERR-00615	SKS error - missing key ID: {0}
SKMS-ERR-00616	SKS error - missing request ID: {0}
SKMS-ERR-00617	SKS error - missing key-class: {0}
SKMS-ERR-00618	SKS error - missing server URL: {0}

Code	Message
SKMS-ERR-00619	SKS error - missing default key-cache policy: {0}
SKMS-ERR-00620	SKS error - missing default key-use policy: {0}
SKMS-ERR-00621	SKS error – SOAP layer error: {0}
SKMS-ERR-00622	SKS error - network layer error: {0}
SKMS-ERR-00623	SKS error – database layer error: {0}
SKMS-ERR-00699	SKS error - other SKS error: {0}

4441

4442

Request checking ERROR Codes (00700 - 00799)

Code	Message
SKMS-ERR-00701	Request-check error - invalid signature: {0}
SKMS-ERR-00702	Request-check error - invalid status: {0}
SKMS-ERR-00703	Request-check error - invalid parameter: {0}
SKMS-ERR-00704	Request-check error - invalid domain ID: {0}
SKMS-ERR-00705	Request-check error - invalid server ID: {0}
SKMS-ERR-00706	Request-check error - invalid polling frequency: {0}
SKMS-ERR-00707	Request-check error - invalid request ID: {0}
SKMS-ERR-00799	Request-check error - other request-check error: {0}

4443

4444

Miscellaneous ERROR Codes (00800 - 00899)

Code	Message
SKMS-ERR-00801	Miscellaneous error - invalid signature: {0}
SKMS-ERR-00802	Miscellaneous error - invalid status: {0}
SKMS-ERR-00899	Miscellaneous error - other miscellaneous error: {0}

Appendix D. Process for requesting a block of SKSML Error Codes for Vendor Use

Vendors who choose to implement SKSML will be able to apply for a unique block of 1,000 code numbers to be assigned for their exclusive use within their implementation of the SKSML protocol. They may choose to use these numbers for messages related to their implementation, in addition to the Standard SKMS Codes & Messages.

In order to receive this unique block of code-numbers, the Vendor must use the following process:

1. An authorized representative of the Vendor must send an e-mail to the OASIS EKMI TC (using the guidelines in this URL: http://www.oasis-open.org/committees/comments/index.php?wg_abbrev=ekmi) asserting the following:
 - a) That they intend to implement the SKSML 1.0 specification within 6-12 months of their dated e-mail;
 - b) That they will implement ALL the Standard Codes & Messages as described in this document, in their implementation;
 - c) That they will not duplicate ANY Standard Code-message within their assigned private-block of numbers;
 - d) That if the TC later chooses to standardize a specific message within the Standard Codes, that may overlap with a Member's private-block message, the Member will use the Standard Code in implementations created subsequent to the standardization of the code/message;
 - e) That they will notify this TC of the release date of their product, with the relevant section of their documentation pointing to the use of the Standard Codes in their product
2. The OASIS EKMI TC Chair or Secretary will verify that the e-mail contains all assertions;
3. The OASIS EKMI TC Chair or Secretary will setup a ballot for the TC to vote to issue the next available block of 1,000 code-numbers to the requesting Vendor. (The first OASIS Member to apply will receive the numbers 10001 – 11000; the next Vendor request will receive 11001 – 12000 and so on).
4. Upon the conclusion of the vote, the TC Chair or Secretary will notify the Vendor of the assigned block of code-numbers (copying the TC);
5. The TC Chair or Secretary will update a web-page on the TC's home-page with the following information:
 - a) The name of the Vendor
 - b) A link to the e-mail request from the Vendor
 - c) A link to the TC ballot authorizing the assignment
 - d) The date of assignment and
 - e) The assigned block of code-numbers
6. When the Vendor releases the product, they will notify the OASIS TC of the product, a link to the product's web URL and a link to the specific section of their documentation high-lighting the Standard Codes & Messages, as well as the Private Block Codes & Messages;
7. The TC Chair or Secretary will then update the above-mentioned web-page to now include this additional information:
 - a) The web-link to the Vendor's implementation
 - b) The web-link to the documentation highlighting the Codes and Messages

4487 8. In the event that a Vendor does not implement the protocol within the 6-12 month period, the TC
4488 will vote and reclaim the private-block assigned to the original requester. Any use of the private-
4489 block of assigned numbers, after the TC has voted to reclaim them, will be a violation of the TC's
4490 guidelines for the SKSML Specification. OASIS Administration will then be directed by this TC to
4491 take whatever action it is legally permitted to address this violation.