



# Symmetric Key Services Markup Language (SKSML) Version 1.0

## Public Review Draft 03

5 August 2010

### Specification URIs:

#### This Version:

<http://docs.oasis-open.org/ekmi/sksml/v1.0/pr03/SKSML-1.0-Specification.html>  
<http://docs.oasis-open.org/ekmi/sksml/v1.0/pr03/SKSML-1.0-Specification.odt> (Authoritative)  
<http://docs.oasis-open.org/ekmi/sksml/v1.0/pr03/SKSML-1.0-Specification.pdf>

#### Previous 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>

#### Latest Version:

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

### Technical Committee:

OASIS Enterprise Key Management Infrastructure (EKMI) TC

### Chair(s):

Anil Saldhana, Red Hat Inc ([anil.saldhana@redhat.com](mailto:anil.saldhana@redhat.com))  
Timothy Bruce, CA Inc ([Timothy.Bruce@ca.com](mailto:Timothy.Bruce@ca.com))

### Editor(s):

Allen Schaaf ([netsecurity@sound-by-design.com](mailto:netsecurity@sound-by-design.com))  
Anil Saldhana, Red Hat, Inc. ([anil.saldhana@redhat.com](mailto:anil.saldhana@redhat.com))  
Tomas Gustavsson, PrimeKey Solutions AB ([tomas@primekey.se](mailto:tomas@primekey.se))

### 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 10 Jun 2008. <http://www.w3.org/TR/xmlsig-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 securely over standard HTTP using XML Security (XML Signature and XML Encryption).

**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](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>).

# Notices

Copyright © OASIS® 2010. 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

111	1 Introduction.....	6
112	1.1 Terminology.....	6
113	1.2 Glossary.....	6
114	1.3 Normative References.....	7
115	2 Background (non-normative).....	9
116	2.1 Requirements (non-normative).....	10
117	3 Examples of use of SKSML (non-normative).....	12
118	3.1 Request for a new symmetric key.....	12
119	3.2 Response with a new symmetric key.....	13
120	3.3 Request for an existing symmetric key.....	17
121	3.4 Response with an existing symmetric key.....	18
122	3.5 Request for a new symmetric key of a specific KeyClass.....	18
123	3.6 Response with a new symmetric key of a specific KeyClass.....	19
124	3.7 Request for multiple new symmetric keys.....	19
125	3.8 Response with multiple new symmetric keys.....	20
126	3.9 Request for a symmetric key with an Encryption Certificate.....	25
127	3.10 Response with an SKS error.....	26
128	3.11 Response with symmetric keys and errors.....	27
129	3.12 Response with a pending Request ID.....	30
130	3.13 Request for an update of a pending Request ID.....	30
131	3.14 Request for a symmetric key-caching policy.....	31
132	3.15 Response with a symmetric key-caching policy (1).....	31
133	3.16 Response with a symmetric key-caching policy (2).....	32
134	3.17 Response with multiple symmetric key-caching policies (3).....	34
135	4 Specification.....	39
136	4.1 Element <SymkeyRequest>.....	39
137	4.2 Element <GlobalKeyID>.....	43
138	4.3 Element <KeyClasses> and <KeyClass>.....	45
139	4.4 Element <X509EncryptionCertificate>.....	47
140	4.5 Element <SymkeyRequestID>.....	48
141	4.6 Element <SymkeyResponse>.....	49
142	4.7 Element <Symkey>.....	53
143	4.8 Element <SymkeyWorkInProgress>.....	55
144	4.9 Element <SymkeyError>.....	58
145	4.10 Element <KeyUsePolicy>.....	60
146	4.11 Type TwoPartIDType.....	63
147	4.12 Element <KeyAlgorithm>.....	64
148	4.13 Element <KeySize>.....	65
149	4.14 Element <Status>.....	66

150	4.15 Element <Permissions>.....	67
151	4.16 Element <PermittedApplications> and <PermittedApplication>.....	73
152	4.17 Element <PermittedDates> and <PermittedDate>.....	76
153	4.18 Element <PermittedDays> and <PermittedDay>.....	78
154	4.19 Element <PermittedDuration>.....	80
155	4.20 Element <PermittedLevels> and <PermittedLevel>.....	81
156	4.21 Element <PermittedLocations> and <PermittedLocation>.....	82
157	4.22 Element <PermittedNumberOfTransactions>.....	85
158	4.23 Element <PermittedTimes> and <PermittedTime>.....	86
159	4.24 Element <PermittedUses> and <PermittedUse>.....	88
160	4.25 Element <KeyCachePolicyRequest>.....	89
161	4.26 Element <KeyCachePolicyResponse>.....	89
162	4.27 Element <KeyCachePolicy>.....	90
163	4.28 Type KeyCacheDetailType.....	93
164	4.29 Use of Web Services Security (WSS).....	95
165	4.30 Use of SKMS Error Codes & Messages.....	96
166	5 Bindings.....	97
167	5.1 W3C Security Binding.....	97
168	5.2 Mutually Authenticated TLS Binding.....	97
169	5.3 SOAP-WSS Binding.....	97
170	6 Conformance.....	98
171	Appendix A.Acknowledgments.....	99
172	Appendix B.Revision History.....	100
173	Appendix C.SKMS Error Codes and Messages.....	101
174	Appendix D.Process for requesting a block of SKSML Error Codes for Vendor Use.....	108
175		

---

# 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

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

213 **SHA-256** – Secure Hashing Algorithm with a resultant size of 256-bits  
 214 **SHA-384** – Secure Hashing Algorithm with a resultant size of 384-bits  
 215 **SHA-512** – Secure Hashing Algorithm with a resultant size of 512-bits  
 216 **SID** or **Server ID** – Server Identifier; the unique integer assigned to every **SKS** server within an  
 217 enterprise's **SKMS**  
 218 **SKCL** – Symmetric Key Client Library; a software library that supports the **SKSML** protocol  
 219 **SKMS** – Symmetric Key Management System; a collection of hardware and software providing symmetric  
 220 encryption key-management services  
 221 **SKS** – Symmetric Key Services; a server that provides symmetric key management services over a  
 222 network or other mechanism selected by implementers  
 223 **SKSML** – Symmetric Key Services Markup Language; an XML-based protocol to request and receive  
 224 symmetric encryption key-management services  
 225 **SOAP** – Simple Object Access Protocol  
 226 **SOAP Body** – The content part of a SOAP message  
 227 **SOAP Envelope** – The SOAP message consisting of a SOAP Header and a SOAP Body, conforming to  
 228 the SOAP protocol standard.  
 229 **SOAP Error** – A SOAP error message response to a SOAP request  
 230 **SOAP Header** – The header part of a SOAP message containing meta-information about the message,  
 231 including security-related objects  
 232 **Symkey** – A symmetric encryption key  
 233 **unbounded** – A parameter used with the “maxOccurs” attribute to indicate an unlimited number  
 234 **X509 Digital Certificate** – a digital certificate that conforms to the Internet Engineering Task Force's PKI  
 235 X509 standard  
 236 **XMLEncryption** – Encrypted content represented in eXtensible Markup Language that conforms to the  
 237 World Wide Web Consortium's XML Encryption standard  
 238 **XMLSignature** – A digital signature represented in eXtensible Markup Language that conforms to the  
 239 World Wide Web Consortium's XML Signature standard

## 240 **1.3 Normative References**

241 **[AES]** Advanced Encryption Standard. NIST FIPS 197. [http://csrc.nist.gov/publications/fips/fips197/fips-](http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf)  
 242 [197.pdf](http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf)  
 243 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119,  
 244 March 1997.  
 245 **[SOAP]** SOAP v1.2 Specification. W3C Recommendation. 27 April 2007. <http://www.w3.org/TR/soap12>  
 246 **[XMLEncryption]** XML Encryption Syntax and Processing. W3C Recommendation. 10 Dec 2002.  
 247 <http://www.w3.org/TR/xmlenc-core/>  
 248 **[XMLSignature]** XML Signature Syntax and Processing. W3C Recommendation. 10 June 2008.  
 249 <http://www.w3.org/TR/xmldsig-core/>  
 250 **[WSS]** OASIS Standard, “Web Services Security – SOAP Message Security 1.0”, March 2004.  
 251 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>

252 **[RFC 2578]** K. McCloghrie, et al. *Structure of Management Information Version 2 (SMIv2)*. IETF RFC  
253 2578, April 1999. <http://www.rfc-editor.org/rfc/rfc2578.txt>

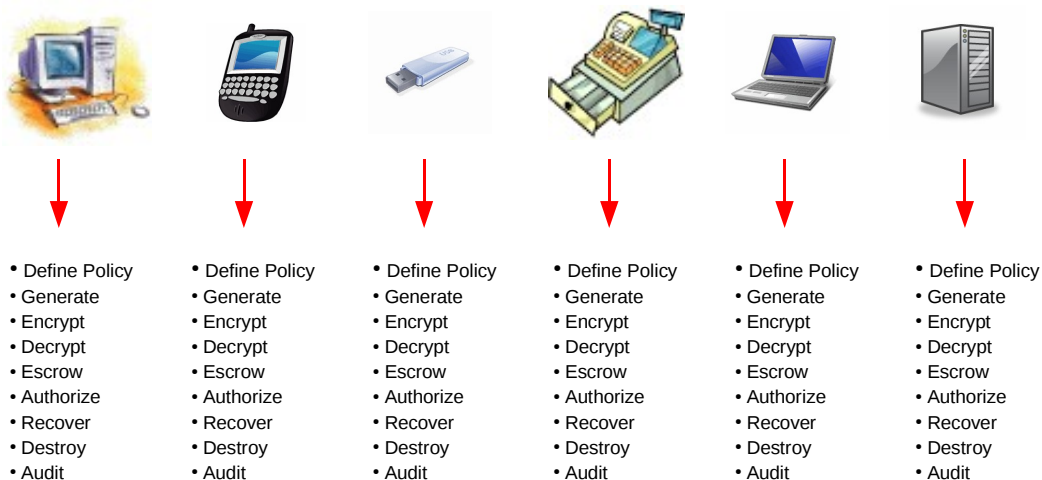


## 2 Background (non-normative)

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

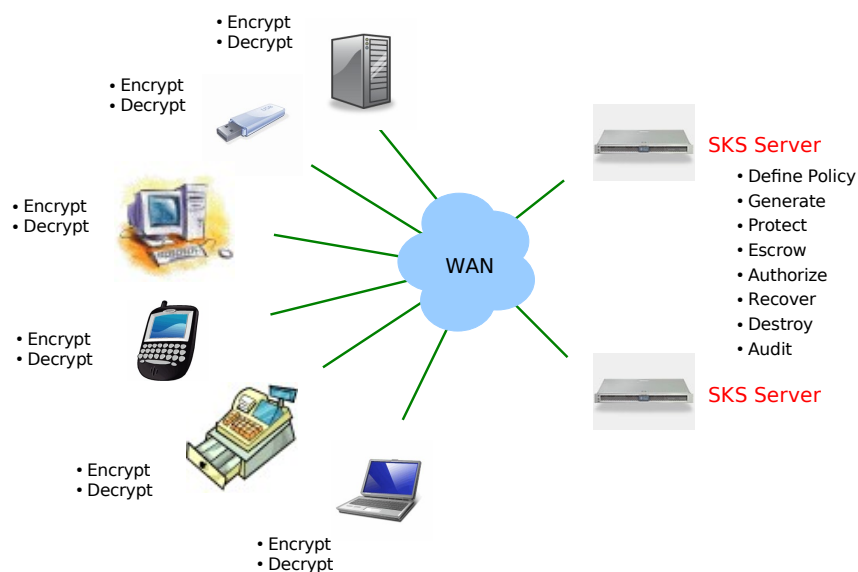
In a rush to provide solutions to the market, vendors have created many device-specific, platform-specific, database-specific and application-specific encryption and key-management tools. While these tools may be capable of performing their stated tasks adequately, a typical enterprise would have to 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;

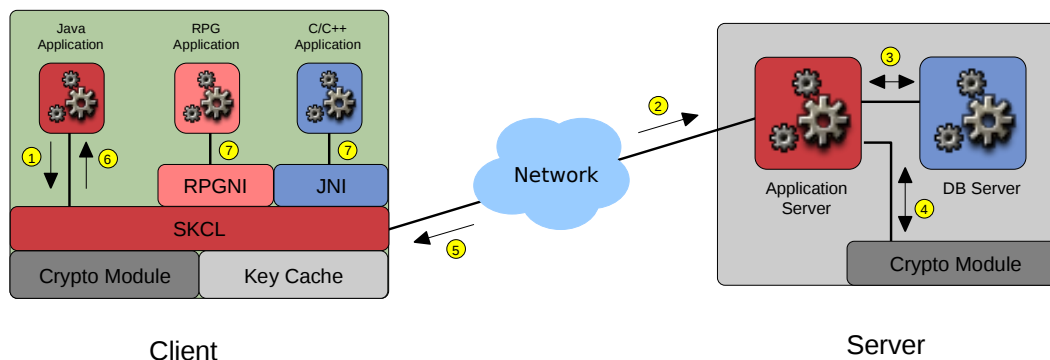
SKSML meets the above requirements in the following manner:

- SKSML uses XML for encapsulating its requests and responses and can thus, be used on any platform that supports XML;

- 307 • Using a scheme that concatenates unique Domain identifiers (Private Enterprise Numbers issued  
308 by the IANA), unique SKS Server identifiers within a domain and unique Key identifiers within an  
309 SKS server, SKSML creates Global Key Identifiers (GKID) that can uniquely identify symmetric  
310 keys across the internet;
  - 311 • SKSML relies on XML Signature and XML Encryption. Relying on RSA cryptographic key-pairs  
312 and digital certificates, SKSML uses the digital signatures for authenticity and message-integrity,  
313 while using RSA-encryption for confidentiality;
  - 314 • Using secure key-caching enabled through centrally-defined policies, SKSML supports the  
315 request and receipt of **KeyCachePolicy** elements by clients for the use of symmetric encryption  
316 keys even when the client is disconnected from the network and an SKS server;
  - 317 • SKSML provides significant flexibility for defining policies on how symmetric encryption keys may  
318 be used by client applications. The **KeyUsePolicy** element allows Security Officers to define  
319 which applications may use a specific key, days and times of use, location of use, purpose of  
320 use, key-sizes, encryption algorithms, etc.
- 321 SKSML is the first key-management protocol that will do for encryption key-management services what  
322 DNS did for name-service protocols: provide a single, standard means of requesting and receiving key-  
323 management services from centrally defined servers.

### 3 Examples of use of SKSML (non-normative)

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



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

#### 3.1 Request for a new symmetric key

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

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

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

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

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

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

344

## 345 3.2 Response with a new symmetric key

346 After an SKS server has performed its operations of authenticating the request, identifying the requester,  
 347 determining policies that apply to the requester, generating the symmetric encryption key in conformance  
 348 to the defined policy and finally escrowing a symmetric key securely, it assembles the following response  
 349 and returns it to the client.

350

351

```

352 [b01] <ekmi:SymkeyResponse
353 [b02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
354 [b03]     xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
355 [b04]     <ekmi:Symkey>
356 [b05]         <ekmi:SymkeyRequestID>10514-1-7476</ekmi:SymkeyRequestID>
357 [b06]         <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
358 [b07]         <ekmi:KeyUsePolicy>
359 [b08]             <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>
360 [b09]             <ekmi:PolicyName>DES-EDE KeyUsePolicy</ekmi:PolicyName>
361 [b10]             <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
362 [b11]             <ekmi:KeyAlgorithm>
363 [b12]                 http://www.w3.org/2001/04/xmlenc#tripledes-cbc
364 [b13]             </ekmi:KeyAlgorithm>
365 [b14]             <ekmi:KeySize>192</ekmi:KeySize>
366 [b15]             <ekmi:Status>Active</ekmi:Status>
367 [b16]         <ekmi:Permissions>
368 [b17]             <ekmi:PermittedApplications ekmi:any="false">
369 [b18]                 <ekmi:PermittedApplication>
370 [b19]                     <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
371 [b20]                     <ekmi:ApplicationName>
372 [b21]                         Payroll Application
373 [b22]                     </ekmi:ApplicationName>
374 [b23]                     <ekmi:ApplicationVersion>1.0</ekmi:ApplicationVersion>
375 [b24]                     <ekmi:ApplicationDigestAlgorithm>
376 [b25]                         http://www.w3.org/2000/09/xmldsig#sha1
377 [b26]                     </ekmi:ApplicationDigestAlgorithm>
378 [b27]                     <ekmi:ApplicationDigestValue>
379 [b28]                         NIG4bKkt4cziEqFFu0oBTM81efU=
380 [b29]                     </ekmi:ApplicationDigestValue>
381 [b30]                 </ekmi:PermittedApplication>
382 [b31]             </ekmi:PermittedApplications>
383 [b32]             <ekmi:PermittedDates ekmi:any="false">
384 [b33]                 <ekmi:PermittedDate>
385 [b34]                     <ekmi:StartDate>2008-01-01</ekmi:StartDate>
386 [b35]                     <ekmi:EndDate>2008-12-31</ekmi:EndDate>
387 [b36]                 </ekmi:PermittedDate>
388 [b37]             </ekmi:PermittedDates>
389 [b38]             <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
390 [b39]             <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
391 [b40]             <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
392 [b41]             <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
393 [b42]             <ekmi:PermittedNumberOfTransactions ekmi:any="true"
394 [b43]             xsi:nil="true"/>
395 [b43]             <ekmi:PermittedTimes ekmi:any="false">
396 [b44]                 <ekmi:PermittedTime>
397 [b45]                     <ekmi:StartTime>07:00:00</ekmi:StartTime>
398 [b46]                     <ekmi:EndTime>19:00:00</ekmi:EndTime>

```

```

399 [b47]         </ekmi:PermittedTime>
400 [b48]         </ekmi:PermittedTimes>
401 [b49]         <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
402 [b50]     </ekmi:Permissions>
403 [b51] </ekmi:KeyUsePolicy>
404 [b52]     <ekmi:EncryptionMethod
405 [b53]         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
406 [b54]     <xenc:CipherData>
407 [b55]         <xenc:CipherValue>
408 [b56]             E9zWB/y93hVSzeTLiDcQoDxmLNxTuxSffMNwCJmtldIqzQHBnpdQ81g6DKdkCFjJM
409 [b57]             hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfglpU8tGFbpWZcd/ATpJD/2fw
410 [b58]             UJow/qimxi8+huUYJMtaGHtXuLLWtx27STRcRpIsY=
411 [b59]         </xenc:CipherValue>
412 [b60]     </xenc:CipherData>
413 [b61] </ekmi:Symkey>
414 [b62] </ekmi:SymkeyResponse>

```

415

416

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

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

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

422 [b05] identifies the **SymkeyRequestID** assigned by the SKS server for this request. In this example, the  
423 concatenated values of the Domain ID, Server ID and Request ID indicate that the key belongs to the  
424 organization represented by the PEN, 10514; it was generated on an SKS server with a Server ID of 1  
425 and is the 7,476<sup>th</sup> unique request received on that SKS server. The client and the server use this value to  
426 associate asynchronous requests and responses for symmetric keys between themselves; however, the  
427 value is also returned for synchronous request/responses too.

428 [b06] identifies the **GlobalKeyID** (GKID) assigned by the SKS server for the new symmetric key being  
429 returned. In this example, the concatenated values of the Domain ID, Server ID and Key ID indicate that  
430 the key belongs to the organization represented by the PEN, 10514; it was generated on an SKS server  
431 with a Server ID of 1 and is the 235<sup>th</sup> unique key generated on that SKS server.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

473 [b24] is the start of the **ApplicationDigestAlgorithm** element. This element permits enterprises to  
474 ensure that only a cryptographically-verified application is authorized to use the symmetric encryption  
475 key. This assumes that the implementation has an infrastructure where the SKCL is capable of  
476 determining a cryptographic value to uniquely identify an application within the run-time environment.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

514 [b40] is an empty (null) **PermittedLevels** element. This element permits enterprises to ensure that the  
 515 symmetric encryption key can be used only by applications that are classified at a given level within the  
 516 Multi-Level Security (MLS) system as defined in the Bell-LaPadula model. In this specific instance, the  
 517 null element indicates that this permission does not apply to this symmetric key, and therefore, there are  
 518 no constraints on its use. However, the key is still subject to all non-null permission clauses.

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

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

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

560 [b61] is the closing tag of the **Symkey** element.

561 [b62] is the closing tag of the **SymkeyResponse** element.

### 562 3.3 Request for an existing symmetric key

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

569 The client application (that has been linked to the SKCL) will call an API method within the SKCL for the  
570 appropriate symmetric key. After the SKCL has ensured that the application is authorized to make such a  
571 request, and assuming that the client application needs the key with the **GlobalKeyID** value of 10514-1-  
572 235, the SKCL assembles the following SKSML request.

573

```
574 [c01] <ekmi:SymkeyRequest
575 [c02]     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
576 [c03]     <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
577 [c04] </ekmi:SymkeyRequest>
```

578

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

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

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

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

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

## 588 3.4 Response with an existing symmetric key

589 After an SKS server has performed its operations of authenticating the request, identifying the requester  
590 and determining whether the requester is authorized to receive the symmetric key, the SKS server sends  
591 back the symmetric key using a **SymkeyResponse** message secured using XML Security. This  
592 **SymkeyResponse** is identical to the message described in Section 3.2 (since the **SymkeyRequest** was  
593 for the same symmetric key identified there) and is therefore, not repeated here for brevity.

## 594 3.5 Request for a new symmetric key of a specific KeyClass

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

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

603

```
604 [e01] <ekmi:SymkeyRequest  
605 [e02]   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
606 [e03]   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
607 [e04]   <ekmi:KeyClasses>  
608 [e05]     <ekmi:KeyClass>HR-Class</ekmi:KeyClass>  
609 [e06]   </ekmi:KeyClasses>  
610 [e07] </ekmi:SymkeyRequest>
```

611

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

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

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

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

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

619 [e06] is the closing tag of the **KeyClasses** element.  
620 [e07] is the closing tag of the **SymkeyRequest** element.

### 621 3.6 Response with a new symmetric key of a specific **KeyClass**

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

### 628 3.7 Request for multiple new symmetric keys

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

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

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

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

649

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

665

666

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 701 3.8 Response with multiple new symmetric keys

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

706 To reduce the verbosity of the response that includes nine symmetric encryption keys, the SKSML shows  
707 only the details of two symmetric keys and the encapsulating element tags for the remaining seven keys.

708 Regardless of what the KeyUsePolicy and/or Permissions elements state in those seven keys, the  
709 schema for each response conforms to the specification described in this document.

```
710 [h01] <ekmi:SymkeyResponse
711 [h02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
712 [h03]     xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
713 [h04]     <ekmi:Symkey>
714 [h05]         <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
715 [h06]         <ekmi:GlobalKeyID>10514-4-3792</ekmi:GlobalKeyID>
716 [h07]         <ekmi:KeyUsePolicy>
717 [h08]             <ekmi:KeyUsePolicyID>10514-9</ekmi:KeyUsePolicyID>
718 [h09]             <ekmi:PolicyName>DES-EDE KeyUsePolicy for EHR-CDC</ekmi:PolicyName>
719 [h10]             <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
720 [h11]             <ekmi:KeyAlgorithm>
721 [h12]                 http://www.w3.org/2001/04/xmlenc#tripledes-cbc
722 [h13]             </ekmi:KeyAlgorithm>
723 [h14]             <ekmi:KeySize>192</ekmi:KeySize>
724 [h15]             <ekmi:Status>Active</ekmi:Status>
725 [h16]             <ekmi:Permissions>
726 [h17]                 <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
727 [h18]                 <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
728 [h19]                 <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
729 [h20]                 <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
730 [h21]                 <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
731 [h22]                 <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
732 [h23]                 <ekmi:PermittedNumberOfTransactions
733 [h24]                     ekmi:any="true" xsi:nil="true"/>
734 [h25]                 <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
735 [h26]                 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
736 [h27]             </ekmi:Permissions>
737 [h28]         </ekmi:KeyUsePolicy>
738 [h29]         <ekmi:EncryptionMethod
739 [h30]             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
740 [h31]         <xenc:CipherData>
741 [h32]             <xenc:CipherValue>
742 [h33]                 E9zWB/y93hVSzeTLiDcQoDxmLnXtuxSffMNwCJmtIdIqzQHBnpdQ81g6DKdkCFjJMa1w
743 [h34]                 hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfglpU8tLWtx27STRcR/2fw2ava
744 [h35]                 ULWtx27STRcRJMtaGHtXuLLWtx27STRcRpIsY=
745 [h36]             </xenc:CipherValue>
746 [h37]         </xenc:CipherData>
747 [h38]     </ekmi:Symkey>
748 [h39] </ekmi:Symkey>
749 [h40]     <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
750 [h41]     <ekmi:GlobalKeyID>10514-4-3793</ekmi:GlobalKeyID>
751 [h42]     <ekmi:KeyUsePolicy>
752 [h43]         <ekmi:KeyUsePolicyID>10514-12</ekmi:KeyUsePolicyID>
753 [h44]         <ekmi:PolicyName>DES-EDE KeyUsePolicy for EHR-CRO</ekmi:PolicyName>
754 [h45]         <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
755 [h46]         <ekmi:KeyAlgorithm>
756 [h47]             http://www.w3.org/2001/04/xmlenc#tripledes-cbc
757 [h48]         </ekmi:KeyAlgorithm>
758 [h49]         <ekmi:KeySize>192</ekmi:KeySize>
759 [h50]         <ekmi:Status>Active</ekmi:Status>
760 [h51]         <ekmi:Permissions>
761 [h52]             <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
762 [h53]             <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
763 [h54]                 <ekmi:PermittedDate>
764 [h55]                     <ekmi:StartDate>2008-01-01</ekmi:StartDate>
765 [h56]                     <ekmi:EndDate>2009-12-31</ekmi:EndDate>
766 [h57]                 </ekmi:PermittedDate>
```

```

767 [h58]         </ekmi:PermittedDates>
768 [h59]         <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
769 [h60]         <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
770 [h61]         <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
771 [h62]         <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
772 [h63]         <ekmi:PermittedNumberOfTransactions
773 [h64]             ekmi:any="true" xsi:nil="true"/>
774 [h65]         <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
775 [h66]         <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
776 [h67]     </ekmi:Permissions>
777 [h68] </ekmi:KeyUsePolicy>
778 [h69] <ekmi:EncryptionMethod
779 [h70]     Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
780 [h71] <xenc:CipherData>
781 [h72]     <xenc:CipherValue>
782 [h73]         qUiQXG0ca8EU871zBoXBjDoDxmLNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJM1
783 [h74]         hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/2fw1
784 [h75]         UJow/qimxi8+ huUYJMtGhtXuLWtx27STRcRpIsY=
785 [h76]     </xenc:CipherValue>
786 [h77] </xenc:CipherData>
787 [h78] </ekmi:Symkey>
788 [h79] <ekmi:Symkey>
789 [h80]     <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
790 [h81]     <ekmi:GlobalKeyID>10514-4-3795</ekmi:GlobalKeyID>
791     ...
792 [h82]     <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
793     ...
794 [h83] </ekmi:Symkey>
795 [h84] <ekmi:Symkey>
796 [h85]     <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
797 [h86]     <ekmi:GlobalKeyID>10514-4-3797</ekmi:GlobalKeyID>
798     ...
799 [h87]     <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
800     ...
801 [h88] </ekmi:Symkey>
802 [h89] <ekmi:Symkey>
803 [h90]     <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
804 [h91]     <ekmi:GlobalKeyID>10514-4-3798</ekmi:GlobalKeyID>
805     ...
806 [h92]     <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
807     ...
808 [h93] </ekmi:Symkey>
809 [h94] <ekmi:Symkey>
810 [h95]     <ekmi:GlobalKeyID>10514-4-3799</ekmi:GlobalKeyID>
811     ...
812 [h96]     <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
813     ...
814 [h97] </ekmi:Symkey>
815 [h98] <ekmi:Symkey>
816 [h99]     <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
817 [h100]    <ekmi:GlobalKeyID>10514-4-3801</ekmi:GlobalKeyID>
818     ...
819 [h101]    <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
820     ...
821 [h102] </ekmi:Symkey>
822 [h103] <ekmi:Symkey>
823 [h104]     <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
824 [h105]     <ekmi:GlobalKeyID>10514-4-3803</ekmi:GlobalKeyID>
825     ...
826 [h106]    <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>

```



```

827      ...
828 [h107]    </ekmi:Symkey>
829 [h108]    <ekmi:Symkey>
830 [h109]      <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
831 [h110]      <ekmi:GlobalKeyID>10514-4-3805</ekmi:GlobalKeyID>
832      ...
833 [h111]      <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>
834      ...
835 [h112]    </ekmi:Symkey>
836 [h113]    </ekmi:SymkeyResponse>

```

837

838

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

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

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

844 [h05] 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 4 and is the 78,122<sup>nd</sup> 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

910 [h31] is the start of the **CipherData** element. This element is from the W3C XML Encryption namespace  
911 (as identified by the "xenc" qualifier in the element name).

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

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

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

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



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

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

- 920 • [h41] identifies a different **GlobalKeyID** (10514-4-3793) for this symmetric key;
- 921 • [h43] identifies a different **KeyUsePolicy** (10514-12) for this symmetric key;
- 922 • [h45] identifies a different **KeyClass** (EHR-CRO) for this symmetric key;
- 923 • [h51] - [h67] defines a different **Permissions** element for this symmetric key;
- 924 • [h73] - [h75] contains a different **CipherValue** for this symmetric key;

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

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

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

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

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

940

### 941 3.9 Request for a symmetric key with an Encryption Certificate

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

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

```

949 [i01] <ekmi:SymkeyRequest
950 [i02]   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
951 [i03]   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
952 [i04]   <ekmi:X509EncryptionCertificate>
953     MIIDfDCCAmSgAwIBAgIIAe/AvLiGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdu3Ryb25nab
954     S2V5IERFTU8gU3Vib3JkaW5hdGUGuQ0ExJDAiBgNVBAsTG0ZvcjBTdHJvbmdLZXkgREVNTyBVc2Ugld
955     T25seTEXMBUGA1UEChMOU3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTcxMDMwWhcNMDcwNzIa64dd3k
956     A1UECxMbRm9yIFN0cm9uZ0tleSBERU1PIFVzZSBPbm5MRcwFQYDVQQKEw5TdHJvbmdBdXRoIEl2da
957     S2V5IERFTU8gU3Vib3JkaW5hdGUGuQ0ExJDAiBgNVBAsTG0ZvcjBTdHJvbmdLZXkgREVNTyBVc2Ugia
958     T25seTEXMBUGA1UEChMOU3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTY0NjEwWhcNMDcwNzI1s34wdd
959     NjEwWjBpMREwDwYKZCIiZPyLGQBARMB0TEVMBGA1UEAxMMU0tTIFNlcnZlci0xMSQwIgYDVQs2dw2
960     ExtGb3Igu3Ryb25nS2V5IERFTU8gVXNlIE9ubHkxZzAVBgNVBAoTDlN0cm9uZ0F1dGggSW5jMIIBd2
  
```

```

961     NBgqhkiG9w0BAQEFAAAQCAQ8AMIIBCgKCAQEAztppqRoU5A8plx1Rz1QEUnlAAM1D5g9+isIr3wxa
962     hbwjtfSMYilnY4iV77xU/nsM0nMZ7RxsLYKdCzQ10DVYqQwqmAvaJ5Z6SVy34gZ51YG+rSWE3NjFsd
963     b0XW8RJYA/Tn6Lmht/qngrcaqmtP0cAAiMRZ0WtCTmC2K/LEqDabXSyU6Hh8ySNE3njybvMwPresf
964     zsYokTdvWQqT6tKo10wJsdJ1+hxM7DrnMLvMNq5reINfsKhDdX17wzhrBUx+hiYA/qo8tMXkL6wsd
965     4PN5dYugtZpSzIdU05tIg58Avhzo7hy5oofBlKFY22CeljQ36u0bMjuyGj6UYHs3rdfsds32rda
966     YzCBnzANBgqhkiG9w0BAQEFAA0BjQAwgYkCgYEAyAmxMZhYA8wHJ4UE4b61s51JVWe4Fygj4MCf3a
967     hvcNAQELBQADggEBACK05PtvZD4WPgl0e=
968 [i05]     </ekmi:X509EncryptionCertificate>
969 [i06]     </ekmi:SymkeyRequest>

```

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

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

973 [i03] 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.

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

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

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

### 980 3.10 Response with an SKS error

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

```

985 [j01]     <ekmi:SymkeyResponse
986 [j02]         xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
987 [j03]         xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
988 [j04]         <ekmi:SymkeyError>
989 [j05]             <ekmi:SymkeyRequestID>10514-2-1044</ekmi:SymkeyRequestID>
990 [j06]             <ekmi:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>
991 [j07]             <ekmi:RequestedKeyClass>Payroll</ekmi:RequestedKeyClass>
992 [j08]             <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
993 [j09]             <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
994 [j10]         </ekmi:SymkeyError>
995 [j11]     </ekmi:SymkeyResponse>

```

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

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

999 [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.

1001 [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 1,044<sup>th</sup> unique request received on that SKS server.

1005 [j06] indicates the **RequestedGlobalKeyID** the client requested. Returning the GKID in the error  
1006 response allows the SKCL to associate the error message with the requesting application on the client  
1007 machine.

1008 [j07] indicates the **RequestedKeyClass** the client requested. Returning the key-class in the error  
1009 response allows the SKCL to associate the error message with the requesting application on the client  
1010 machine.

1011 [j08] is an **ErrorCode** returned by the SKS server. The code may be one of the standard error codes  
1012 defined in this specification, or may be a vendor-specific error code.

1013 [j09] is the text of the **ErrorMessage**, localized to the region and language of the requesting client  
1014 application.

1015 [j10] is the closing tag of the **SymkeyError** tag.

1016 [j11] is the closing tag of the **SymkeyResponse** tag.

### 1017 3.11 Response with symmetric keys and errors

1018 When a client application requests multiple symmetric keys, the SKS server may respond in one of three  
1019 ways. The SKS server may:

- 1020 i. Return all symmetric keys as requested;
- 1021 ii. Return no symmetric keys – i.e. it returns all errors;
- 1022 iii. Return some symmetric keys and some errors.

1023 The following SKSML shows the third case, where the server returns some symmetric keys and errors in  
1024 response to a request for multiple keys (such as the one shown in **Section 3.7 Request for multiple**  
1025 **new symmetric keys**).

1026 In a response that contains a mix of symmetric keys and errors, all symmetric keys precede all errors –  
1027 i.e. the **SymkeyResponse** element will not consist of **Symkeys** interspersed with **SymkeyErrors** in  
1028 between; all **Symkeys** (if any) will start from the top of the response till the first **SymkeyError** element.

```
1029 [k01]      <ekmi:SymkeyResponse
1030 [k02]          xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1031 [k03]          xmlns:xenc='http://www.w3.org/2001/04/xmenc#'>
1032 [k04]      <ekmi:Symkey>
1033 [k05]          <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1034 [k06]          <ekmi:GlobalKeyID>10514-4-3792</ekmi:GlobalKeyID>
1035 [k07]      <ekmi:KeyUsePolicy>
1036 [k08]          <ekmi:KeyUsePolicyID>10514-9</ekmi:KeyUsePolicyID>
1037 [k09]          <ekmi:PolicyName>DES-EDE Policy for EHR-CDC</ekmi:PolicyName>
1038 [k10]          <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
1039 [k11]          <ekmi:KeyAlgorithm>
1040 [k12]              http://www.w3.org/2001/04/xmenc#tripledes-cbc
1041 [k13]          </ekmi:KeyAlgorithm>
1042 [k14]          <ekmi:KeySize>192</ekmi:KeySize>
1043 [k15]          <ekmi:Status>Active</ekmi:Status>
1044 [k16]      <ekmi:Permissions>
1045 [k17]          <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
1046 [k18]          <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
1047 [k19]          <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
1048 [k20]          <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
1049 [k21]          <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
1050 [k22]          <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
1051 [k23]          <ekmi:PermittedNumberOfTransactions
```

```

1052 [k24]          ekmi:any="true" xsi:nil="true"/>
1053 [k25]          <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
1054 [k26]          <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
1055 [k27]          </ekmi:Permissions>
1056 [k28]        </ekmi:KeyUsePolicy>
1057 [k29]        <ekmi:EncryptionMethod
1058 [k30]          Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
1059 [k31]        <xenc:CipherData>
1060 [k32]          <xenc:CipherValue>
1061 [k33]            E9zWB/y93hVSzeTLiDcQoDxmLNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJ
1062 [k34]            hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfg1pU8tLWtx27STRcR/2fw
1063 [k35]            ULWtx27STRcRJMtaGHtXuLLWtx27STRcRpIsY=
1064 [k36]          </xenc:CipherValue>
1065 [k37]        </xenc:CipherData>
1066 [k38]      </ekmi:Symkey>
1067 [k39]    <ekmi:Symkey>
1068 [k40]      <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1069 [k41]      <ekmi:GlobalKeyID>10514-4-3793</ekmi:GlobalKeyID>
1070 [k42]      <ekmi:KeyUsePolicy>
1071 [k43]        <ekmi:KeyUsePolicyID>10514-12</ekmi:KeyUsePolicyID>
1072 [k44]        <ekmi:PolicyName>DES-EDE Policy for EHR-CRO</ekmi:PolicyName>
1073 [k45]        <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
1074 [k46]        <ekmi:KeyAlgorithm>
1075 [k47]          http://www.w3.org/2001/04/xmlenc#tripleDES-cbc
1076 [k48]        </ekmi:KeyAlgorithm>
1077 [k49]        <ekmi:KeySize>192</ekmi:KeySize>
1078 [k50]        <ekmi:Status>Active</ekmi:Status>
1079 [k51]        <ekmi:Permissions>
1080 [k52]          <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
1081 [k53]          <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
1082 [k54]            <ekmi:PermittedDate>
1083 [k55]              <ekmi:StartDate>2008-01-01</ekmi:StartDate>
1084 [k56]              <ekmi:EndDate>2009-12-31</ekmi:EndDate>
1085 [k57]            </ekmi:PermittedDate>
1086 [k58]          </ekmi:PermittedDates>
1087 [k59]          <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
1088 [k60]          <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
1089 [k61]          <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
1090 [k62]          <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
1091 [k63]          <ekmi:PermittedNumberOfTransactions
1092 [k64]            ekmi:any="true" xsi:nil="true"/>
1093 [k65]          <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
1094 [k66]          <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
1095 [k67]        </ekmi:Permissions>
1096 [k68]      </ekmi:KeyUsePolicy>
1097 [k69]    <ekmi:EncryptionMethod
1098 [k70]      Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
1099 [k71]    <xenc:CipherData>
1100 [k72]      <xenc:CipherValue>
1101 [k73]        qUiQXG0ca8EU871zBoXBjDoDxmLNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCF
1102 [k74]        hQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/
1103 [k75]        UJow/qimxi8+ huUYJMtaGHtXuLLWtx27STRcRpIsY=
1104 [k76]      </xenc:CipherValue>
1105 [k77]    </xenc:CipherData>
1106 [k78]  </ekmi:Symkey>
1107 [k79] <ekmi:Symkey>
1108 [k80]   <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1109 [k81]   <ekmi:GlobalKeyID>10514-4-3795</ekmi:GlobalKeyID>
1110       ...
1111 [k82]   <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>

```

```

1112      ...
1113 [k83]      </ekmi:Symkey>
1114 [k84]      <ekmi:Symkey>
1115 [k85]      <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1116 [k86]      <ekmi:GlobalKeyID>10514-4-3797</ekmi:GlobalKeyID>
1117      ...
1118 [k87]      <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
1119      ...
1120 [k88]      </ekmi:Symkey>
1121 [k89]      <ekmi:Symkey>
1122 [k90]      <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1123 [k91]      <ekmi:GlobalKeyID>10514-4-3798</ekmi:GlobalKeyID>
1124      ...
1125 [k92]      <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
1126      ...
1127 [k93]      </ekmi:Symkey>
1128 [k94]      <ekmi:Symkey>
1129 [k95]      <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1130 [k96]      <ekmi:GlobalKeyID>10514-4-3799</ekmi:GlobalKeyID>
1131      ...
1132 [k97]      <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
1133      ...
1134 [k98]      </ekmi:Symkey>
1135 [k99]      <ekmi:Symkey>
1136 [k100]     <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1137 [k101]     <ekmi:GlobalKeyID>10514-4-3801</ekmi:GlobalKeyID>
1138      ...
1139 [k102]     <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
1140      ...
1141 [k103]     </ekmi:Symkey>
1142 [k104]     <ekmi:SymkeyError>
1143 [k105]     <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1144 [k106]     <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
1145 [k107]     <ekmi:RequestedKeyClass>EHR-PAT</ekmi:RequestedKeyClass>
1146 [k108]     <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
1147 [k109]     <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
1148 [k110]     </ekmi:SymkeyError>
1149 [k111]     <ekmi:SymkeyError>
1150 [k112]     <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
1151 [k113]     <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
1152 [k114]     <ekmi:RequestedKeyClass>EHR-PHY</ekmi:RequestedKeyClass>
1153 [k115]     <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
1154 [k116]     <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
1155 [k117]     </ekmi:SymkeyError>
1156 [k118]     </ekmi:SymkeyResponse>

```

1157 **[k01] – [k103]** is no different from the response shown in **Section 3.8 Response with multiple new**  
1158 **symmetric keys.**

1159 **[k104] - [k110]** identifies the first **SymkeyError** returned by the SKS server. It is not unlike the error  
1160 described in **Section 3.9 Response with an SKS error.**

1161 **[k111] - [k117]** identifies the second **SymkeyError** returned by the SKS server.

1162 **[k118]** is the closing tag of the **SymkeyResponse** tag.

## 1163 3.12 Response with a pending Request ID

1164 Some use-cases call for sending a request to the SKS server and getting a response, asynchronously. In  
1165 these situations, SKSML allows for returning a response with a **SymkeyRequestID** in it. This indicates



1166 that the request is being processed and its status is currently pending. The client application may use this  
1167 request identifier to poll the SKS server for an update on the request status.

1168 The format of the **SymkeyResponse** is as follows.

```
1169 [l01]      <ekmi:SymkeyResponse  
1170 [l02]          xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'  
1171 [l03]          xmlns:xenc='http://www.w3.org/2001/04/xmenc#'>  
1172 [l04]      <ekmi:SymkeyWorkInProgress>  
1173 [l05]          <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>  
1174 [l06]          <ekmi:SymkeyRequestID>10514-4-7235</ekmi:SymkeyRequestID>  
1175 [l07]      </ekmi:SymkeyWorkInProgress>  
1176 [l08]      </ekmi:SymkeyResponse>
```

1177 [l01] – [l03] is the standard preamble to a **SymkeyResponse** element.

1178 [l04] identifies the start of the second **SymkeyWorkInProgress** element. This indicates that the server  
1179 was unable to return a response at this time and as a result, has provided a request identifier that the  
1180 client may use to query the server at a later time for a response.

1181 [l05] contains the **RequestedGlobalKeyID** element. This element contains either a request for a new  
1182 symmetric key, or a request for an existing symmetric key. In this example, the request is for a new  
1183 symmetric key.

1184 [l06] contains the **SymkeyRequestID** element. This element contains the unique request identifier  
1185 provided by the SKS server for the request it received. The **SymkeyRequestID** appears identical to the  
1186 **GlobalKeyID**; however, their meanings are very distinct.

1187 [l07] contains the closing tag of the **SymkeyWorkInProgress** element.

1188 [l08] is the closing tag of the **SymkeyResponse** element.

### 1189 3.13 Request for an update of a pending Request ID

1190 Some use-cases call for sending a request and getting a response, to and from the SKS server,  
1191 asynchronously. In these situations, SKSML allows for returning a response with a **SymkeyRequestID** in  
1192 it that indicates that the status of the request is currently pending. The client application may use this  
1193 request identifier to poll the SKS server for an update on the request status.

1194 When a client makes a **SymkeyRequest** with a **SymkeyRequestID** in it, the SKS server may send back  
1195 one of three responses: i) a **SymkeyWorkInProgress** with the same **SymkeyRequestID** in it, indicating  
1196 that the request is still pending; ii) a **SymkeyError**, indicating that the request was processed, but the  
1197 processing resulted in an error; and iii) a **Symkey** with a symmetric key in it, indicating a successful  
1198 response.

1199 The format of the **SymkeyRequest** is as follows.

```
1200 [m01]      <ekmi:SymkeyRequest  
1201 [m02]          xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'  
1202 [m03]          <ekmi:SymkeyRequestID>10514-4-7235</ekmi:SymkeyRequestID>  
1203 [m04]      </ekmi:SymkeyRequest>
```

1204 [m01] – [m02] is the standard preamble to a **SymkeyRequest** element.

1205 [m03] contains the **SymkeyRequestID** element. This element contains the unique request identifier  
1206 provided by the SKS server when the client made the request for a symmetric key earlier. The  
1207 **SymkeyRequestID** appears identical to the **GlobalKeyID**; however, their meanings are very distinct when  
1208 they are wrapped in the appropriate element tags.

1209 [m04] contains the closing tag of the **SymkeyRequest** element.

### 3.14 Request for a symmetric key-caching policy

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:

```
[n01] <ekmi:KeyCachePolicyRequest
[n02]   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
[n03]   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
[n04]     ...
[n05]   </ds:Signature>
[n06] </ekmi:KeyCachePolicyRequest>
```

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

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

[n03] to [n05] will identify the XML Digital Signature of the sender.

[n06] is the end of the **KeyCachePolicyRequest** element.

The server uses the digital signature in XML format to establish the identity of the requester, as well as to ensure message integrity in the request.

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

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

After an SKS server has performed its operations of authenticating a **KeyCachePolicyRequest**, identifying the requester, determining policies that apply to the requester, it assembles the following response and returns it to the client.

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

1254 **[o01]** is the start of the **KeyCachePolicyResponse** element.

1255 **[o02]** and **[o03]** identify the namespaces to which this XML conforms, and the location of their XML  
 1256 Schema Definitions (XSD).

1257 **[o04]** is the start of the first – and only - **KeyCachePolicy** element.

1258 **[o05]** identifies the **KeyCachePolicyID** (KCPID) assigned to this policy by the SKS server. In this  
 1259 example, the concatenated values of the Domain ID and Policy ID indicate that the key belongs to the  
 1260 organization represented by the PEN, 10514; and is the first key-caching policy within the SKMS.

1261 **[o06]** provides a descriptive name for this key-cache policy through the **PolicyName** element, which is  
 1262 helpful to human readers when identifying this policy.

1263 **[o07]** is the start tag of the **Description** element.

1264 **[o08]** - **[o10]** provides a human-readable description about this key-cache policy.

1265 **[o11]** is the closing tag of the **Description** element.

1266 **[o12]** specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are  
 1267 subject to this caching policy.

1268 **[o13]** specifies the date and time that this **KeyCachePolicy** is effective. This is accomplished through a  
 1269 **StartDate** element. In this example, the policy is effective as of January 01, 2008.

1270 **[o14]** specifies the date and time that this **KeyCachePolicy** becomes invalid. This is accomplished  
 1271 through a **EndDate** element. In this example, the use of the UNIX “epoch” date (January 01, 1969)  
 1272 indicates that this policy never expires.

1273 **[o15]** specifies the frequency at which this client must check with the SKS server for updates to the key-  
 1274 caching policy. This is specified in seconds in the **PolicyCheckInterval** element; in this example it is a  
 1275 monthly interval.

1276 **[o16]** indicates the **Status** of this **KeyCachePolicy** and whether it is an active policy or not.

1277 **[o17]** is the closing tag of the **KeyCachePolicy** element.

1278 **[o18]** is the closing tag of the **KeyCachePolicyResponse** element.

### 1279 3.16 Response with a symmetric key-caching policy (2)

1280 This is a second example of a key-caching policy response that has additional elements in the policy  
 1281 permitting caching and specify the number of unused and used (for encryption) symmetric keys that may  
 1282 be cached by the client machine.

```

1283 [p01] <ekmi:KeyCachePolicyResponse
1284 [p02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1285 [p03]     xmlns:xenc='http://www.w3.org/2001/04/xmenc#'>
1286 [p04]     <ekmi:KeyCachePolicy>
1287 [p05]         <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
1288 [p06]         <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
1289 [p07]         <ekmi:Description>
1290 [p08]             This policy defines how company-issued laptops will manage symmetric
1291 [p09]             keys used for file/disk encryption in their local cache. This
1292 [p10]             policy must be used by all laptops that use the company EKMI.
1293 [p11]         </ekmi:Description>
1294 [p12]         <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
1295 [p13]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1296 [p14]         <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
1297 [p15]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
  
```



```

1298 [p16]      <ekmi:Status>Active</ekmi:Status>
1299 [p17]      <ekmi:NewKeysCacheDetail>
1300 [p18]      <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1301 [p19]      <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1302 [p20]      </ekmi:NewKeysCacheDetail>
1303 [p21]      <ekmi:UsedKeysCacheDetail>
1304 [p22]      <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1305 [p23]      <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1306 [p24]      </ekmi:UsedKeysCacheDetail>
1307 [p25]      </ekmi:KeyCachePolicy>
1308 [p26] </ekmi:KeyCachePolicyResponse>

```

1309 [p01] is the start of the **KeyCachePolicyResponse** element.

1310 [p02] and [p03] identify the namespaces to which this XML conforms, and the location of their XML Schema Definitions (XSD).

1312 [p04] is the start of the first – and only - **KeyCachePolicy** element.

1313 [p05] identifies the **KeyCachePolicyID** (KCPID) assigned by the SKS server for the key-caching policy being returned.

1315 [p06] provides a descriptive name for this key-cache policy through the **PolicyName** element, which is helpful to human readers when identifying this policy.

1317 [p07] is the start tag of the **Description** element.

1318 [p08] - [p10] provides a human-readable description about this key-cache policy.

1319 [p11] is the closing tag of the **Description** element.

1320 [p12] specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are subject to this caching policy.

1322 [p13] specifies the date and time that this **KeyCachePolicy** is effective. This is accomplished through a **StartDate** element. In this example, the policy is effective as of January 01, 2008.

1324 [p14] specifies the date and time that this **KeyCachePolicy** becomes invalid. This is accomplished through a **EndDate** element. In this example, the policy expires on December 31, 2008.

1326 [p15] specifies the frequency at which this client must check with the SKS server for updates to the key-caching policy. This is specified in seconds in the **PolicyCheckInterval** element; in this example it is a monthly interval.

1329 [p16] indicates the **Status** of this **KeyCachePolicy** and whether it is an active policy or not.

1330 [p17] is the start of the **NewKeysCacheDetail** element, which provides details about how many new symmetric keys – that haven't been used for any encryption transactions – may be cached by the client and for how long.

1333 [p18] indicates the maximum number of new (unused) symmetric keys that may be cached by the client. This is specified through the **MaximumKeys** element. When the client uses a symmetric key, this reduces the number of new symmetric keys. In this case, the SKCL connects to the SKS server (if it is on the network) and requests a new symmetric key to add to its new-key cache.

1337 [p19] indicates the maximum duration that new (unused) symmetric keys may be cached by the client. This is specified through the **MaximumDuration** element in seconds. If there are any new keys that exceed this duration limit, the SKCL deletes the specific symmetric key and replaces it with a new symmetric key from the SKS server.

1341 [p20] is the closing tag of the **NewKeysCacheDetail** element.

1342 [p21] is the start of the **UsedKeysCacheDetail** element, which provides details about how many used  
 1343 symmetric keys – those that HAVE been used for any encryption transactions – may be cached by the  
 1344 client and for how long.

1345 [p22] indicates the maximum number of used symmetric keys that may be cached by the client through  
 1346 the **MaximumKeys** element. If the client already has the maximum number of used-keys in its cache,  
 1347 using the First-In-First-Out (FIFO) method, it deletes the oldest symmetric key in the cache to replace  
 1348 with the key that transitioned from the “new” to “used” status.

1349 [p23] indicates the maximum duration that used symmetric keys may be cached by the client through the  
 1350 **MaximumDuration** element in seconds. If there are any used keys that exceed this duration limit, the  
 1351 SKCL deletes the specific symmetric key. While this may temporarily reduce the number of used  
 1352 symmetric keys in the cache, the SKCL takes the most conservative position when making this decision.

1353 [p24] is the closing tag of the **UsedKeysCacheDetail** element.

1354 [p25] is the closing tag of the **KeyCachePolicy** element.

1355 [p26] is the closing tag of the **KeyCachePolicyResponse** element.

### 1356 3.17 Response with multiple symmetric key-caching policies (3)

1357 This is a third example of a key-caching policy response that has multiple key-cache policies that apply to  
 1358 different classes of symmetric keys.

```

1359 [q01] <ekmi:KeyCachePolicyResponse
1360 [q02]     xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
1361 [q03]     xmlns:xenc='http://www.w3.org/2001/04/xmenc#'>
1362 [q04]     <ekmi:KeyCachePolicy>
1363 [q05]         <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
1364 [q06]         <ekmi:PolicyName>No Caching Policy</ekmi:PolicyName>
1365 [q07]         <ekmi:Description>
1366 [q08]             This policy is for high-risk, always-connected machines on the
1367 [q09]             network, which will never cache symmetric keys locally. This
1368 [q10]             policy never expires (but checks monthly for any updates).
1369 [q11]         </ekmi:Description>
1370 [q12]         <ekmi:KeyClass>NoCachingClass</ekmi:KeyClass>
1371 [q13]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1372 [q14]         <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
1373 [q15]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1374 [q16]         <ekmi>Status>Active</ekmi>Status>
1375 [q17]     </ekmi:KeyCachePolicy>
1376 [q18]     <ekmi:KeyCachePolicy>
1377 [q19]         <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
1378 [q20]         <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
1379 [q21]         <ekmi:Description>
1380 [q22]             This policy defines how company-issued laptops will manage symmetric
1381 [q23]             keys used for file/disk encryption in their local cache. This
1382 [q24]             policy must be used by all laptops that use the company EKMI.
1383 [q25]         </ekmi:Description>
1384 [q26]         <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
1385 [q27]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1386 [q28]         <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
1387 [q29]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1388 [q30]         <ekmi>Status>Active</ekmi>Status>
1389 [q31]         <ekmi:NewKeysCacheDetail>
1390 [q32]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1391 [q33]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1392 [q34]         </ekmi:NewKeysCacheDetail>
1393 [q35]     </ekmi:UsedKeysCacheDetail>
  
```

```

1394 [q36]         <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1395 [q37]         <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1396 [q38]     </ekmi:UsedKeysCacheDetail>
1397 [q39] </ekmi:KeyCachePolicy>
1398 [q40] <ekmi:KeyCachePolicy>
1399 [q41]     <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
1400 [q42]     <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
1401 [q43]     <ekmi:Description>
1402 [q44]         This policy defines how company-issued laptops will manage
1403 [q45]         symmetric keys used for file/disk encryption in their local
1404 [q46]         cache. This policy must be used by all laptops.
1405 [q47]     </ekmi:Description>
1406 [q48]     <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
1407 [q49]     <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
1408 [q50]     <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
1409 [q51]     <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
1410 [q52]     <ekmi>Status>Active</ekmi>Status>
1411 [q53]     <ekmi:NewKeysCacheDetail>
1412 [q54]         <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1413 [q55]         <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1414 [q56]     </ekmi:NewKeysCacheDetail>
1415 [q57]     <ekmi:UsedKeysCacheDetail>
1416 [q58]         <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
1417 [q59]         <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
1418 [q60]     </ekmi:UsedKeysCacheDetail>
1419 [q61] <ekmi:KeyCachePolicy>
1420 [q62] </ekmi:KeyCachePolicyResponse>

```

1421

1422 [q01] is the start of the **KeyCachePolicyResponse** element.

1423 [q02] and [q03] identify the namespaces to which this XML conforms, and the location of their XML  
1424 Schema Definitions (XSD).

1425 [q04] is the start of the first of three **KeyCachePolicy** elements in this response.

1426 [q05] identifies the **KeyCachePolicyID** (KCPID) assigned to this policy by the SKS server. In this  
1427 example, the concatenated values of the Domain ID and Policy ID indicate that the key belongs to the  
1428 organization represented by the PEN, 10514; and is the first key-caching policy within the SKMS.

1429 [q06] provides a descriptive name for this key-cache policy through the **PolicyName** element, which is  
1430 helpful to human readers when identifying this policy.

1431 [q07] is the start tag of the **Description** element.

1432 [q08] - [q10] provides a human-readable description about this key-cache policy.

1433 [q11] is the closing tag of the **Description** element.

1434 [q12] specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are  
1435 subject to this caching policy.

1436 [q13] specifies the date and time that this **KeyCachePolicy** is effective. This is accomplished through a  
1437 **StartDate** element. In this example, the policy is effective as of January 01, 2008.

1438 [q14] specifies the date and time that this **KeyCachePolicy** becomes invalid. This is accomplished  
1439 through a **EndDate** element. In this example, the use of the UNIX "epoch" date (January 01, 1969)  
1440 indicates that this policy never expires.

1441 [q15] specifies the frequency at which this client must check with the SKS server for updates to the key-  
 1442 caching policy. This is specified in seconds in the **PolicyCheckInterval** element; in this example it is a  
 1443 monthly interval.

1444 [q16] indicates the **Status** of this **KeyCachePolicy** and whether it is an active policy or not.

1445 [q17] is the closing tag of the first **KeyCachePolicy** element.

1446 [q18] is the start of the second of three **KeyCachePolicy** elements in this response.

1447 [q19] identifies the **KeyCachePolicyID** (KCPID) assigned by the SKS server for the key-caching policy  
 1448 being returned.

1449 [q20] provides the descriptive name for this key-cache policy through the **PolicyName** element.

1450 [q21] is the start tag of the **Description** element.

1451 [q22] - [q24] provides a human-readable description about this key-cache policy.

1452 [q25] is the closing tag of the **Description** element.

1453 [q26] specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are  
 1454 subject to this caching policy.

1455 [q27] specifies the date and time that this **KeyCachePolicy** is effective. This is accomplished through a  
 1456 **StartDate** element. In this example, the policy is effective as of January 01, 2008.

1457 [q28] specifies the date and time that this **KeyCachePolicy** becomes invalid. This is accomplished  
 1458 through a **EndDate** element. In this example, the policy expires on December 31, 2008.

1459 [q29] specifies the frequency at which this client must check with the SKS server for updates to the key-  
 1460 caching policy. This is specified in seconds in the **PolicyCheckInterval** element; in this example it is a  
 1461 monthly interval.

1462 [q30] indicates the **Status** of this **KeyCachePolicy** and whether it is an active policy or not.

1463 [q31] is the start of the **NewKeysCacheDetail** element, which provides details about how many new  
 1464 symmetric keys – that haven't been used for any encryption transactions – may be cached by the client  
 1465 and for how long.

1466 [q32] indicates the maximum number of new (unused) symmetric keys that may be cached by the client.  
 1467 This is specified through the **MaximumKeys** element. When the client uses a symmetric key, this  
 1468 reduces the number of new symmetric keys. In this case, the SKCL connects to the SKS server (if it is on  
 1469 the network) and requests a new symmetric key to add to its new-key cache.

1470 [q33] indicates the maximum duration that new (unused) symmetric keys may be cached by the client.  
 1471 This is specified through the **MaximumDuration** element in seconds. If there are any new keys that  
 1472 exceed this duration limit, the SKCL deletes the specific symmetric key and replaces it with a new  
 1473 symmetric key from the SKS server.

1474 [q34] is the closing tag of the **NewKeysCacheDetail** element.

1475 [q35] is the start of the **UsedKeysCacheDetail** element, which provides details about how many used  
 1476 symmetric keys – those that HAVE been used for any encryption transactions – may be cached by the  
 1477 client and for how long.

1478 [q36] indicates the maximum number of used symmetric keys that may be cached by the client through  
 1479 the **MaximumKeys** element. If the client already has the maximum number of used-keys in its cache,  
 1480 using the First-In-First-Out (FIFO) method, it deletes the oldest symmetric key in the cache to replace  
 1481 with the key that transitioned from the “new” to “used” status.

1482 [q37] indicates the maximum duration that used symmetric keys may be cached by the client through the  
1483 **MaximumDuration** element in seconds. If there are any used keys that exceed this duration limit, the  
1484 SKCL deletes the specific symmetric key. While this may temporarily reduce the number of used  
1485 symmetric keys in the cache, the SKCL takes the most conservative position when making this decision.

1486 [q38] is the closing tag of the **UsedKeysCacheDetail** element.

1487 [q39] is the closing tag of the second **KeyCachePolicy** element.

1488 [q40] is the start of the third **KeyCachePolicy** element in this response.

1489 [q41] identifies the **KeyCachePolicyID** (KCPID) assigned by the SKS server for the key-caching policy  
1490 being returned.

1491 [q42] provides the descriptive name for this key-cache policy through the **PolicyName** element.

1492 [q43] is the start tag of the **Description** element.

1493 [q44] - [q46] provides a human-readable description about this key-cache policy.

1494 [q47] is the closing tag of the **Description** element.

1495 [q48] specifies the **KeyClass** to which this policy applies. Only keys that belong to this key-class are  
1496 subject to this caching policy.

1497 [q49] specifies the date and time that this **KeyCachePolicy** is effective.

1498 [q50] specifies the date and time that this **KeyCachePolicy** becomes invalid.

1499 [q51] specifies the frequency at which this client must check with the SKS server for updates to the key-  
1500 caching policy.

1501 [q52] indicates the **Status** of this **KeyCachePolicy** and whether it is an active policy or not.

1502 [q53] is the start of the **NewKeysCacheDetail** element, which provides details about how many new  
1503 symmetric keys may be cached by the client and for how long.

1504 [q54] indicates the maximum number of new (unused) symmetric keys that may be cached by the client.  
1505 This is specified through the **MaximumKeys** element. When the client uses a symmetric key, this  
1506 reduces the number of new symmetric keys. In this case, the SKCL connects to the SKS server (if it is on  
1507 the network) and requests a new symmetric key to add to its new-key cache.

1508 [q55] indicates the maximum duration that new (unused) symmetric keys may be cached by the client.  
1509 This is specified through the **MaximumDuration** element in seconds. If there are any new keys that  
1510 exceed this duration limit, the SKCL deletes the specific symmetric key and replaces it with a new  
1511 symmetric key from the SKS server.

1512 [q56] is the closing tag of the **NewKeysCacheDetail** element.

1513 [q57] is the start of the **UsedKeysCacheDetail** element, which provides details about how many used  
1514 symmetric keys – those that HAVE been used for any encryption transactions – may be cached by the  
1515 client and for how long.

1516 [q58] indicates the maximum number of used symmetric keys that may be cached by the client through  
1517 the **MaximumKeys** element. If the client already has the maximum number of used-keys in its cache,  
1518 using the First-In-First-Out (FIFO) method, it deletes the oldest symmetric key in the cache to replace  
1519 with the key that transitioned from the “new” to “used” status.

1520 [q59] indicates the maximum duration that used symmetric keys may be cached by the client through the  
1521 **MaximumDuration** element in seconds. If there are any used keys that exceed this duration limit, the  
1522 SKCL deletes the specific symmetric key. While this may temporarily reduce the number of used  
1523 symmetric keys in the cache, the SKCL takes the most conservative position when making this decision.

- 1524    **[q60]** is the closing tag of the ***UsedKeysCacheDetail*** element.
- 1525    **[q61]** is the closing tag of the third and final ***KeyCachePolicy*** element in this response.
- 1526    **[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.

#### 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"
```

```

1561         type="ekmi:SymkeyRequestIDType"
1562
1563         minOccurs="1"
1564         maxOccurs="unbounded">
1565     </xsd:element>
1566 </xsd:sequence>
1567 </xsd:choice>
1568 </xsd:complexType>
1569 </xsd:element>

```

1570

1571 The <SymkeyRequest> element consists of a choice of two sequences of elements; one or the other  
 1572 sequence MUST be present in a <SymkeyRequest>.

1573 1. <GlobalKeyID> [Required]

1574

1575 The first sequence choice consists of the <GlobalKeyID> element of type **GlobalKeyIDType**. It  
 1576 identifies the unique global key identifier of the requested symmetric key within the target  
 1577 Symmetric Key Management System (SKMS). If this sequence is chosen, there MUST be at  
 1578 least one <GlobalKeyID> element in a <SymkeyRequest>, but there may be an unbounded  
 1579 (unlimited) number of <GlobalKeyID> elements specified.

1580

1581 The <GlobalKeyID> element and **GlobalKeyIDType** is specified in Section 4.2.

1582 <KeyClasses> [Optional]

1583

1584 This element of type **KeyClassesType**, when specified, identifies at least one <KeyClass>  
 1585 element, but may specify an unbounded (unlimited) number of <KeyClass> elements within the  
 1586 <KeyClasses> set. Client applications may request one or more symmetric keys conforming to  
 1587 one or more key classes required by the application. If the client application is authorized to  
 1588 receive keys conforming to such key classes, the **SKS** server will generate and supply them.

1589

1590 When more than one <GlobalKeyID> for a new symmetric key is specified in the request, there  
 1591 MAY be only one <KeyClass> element within the <KeyClasses> set.

1592

1593 When the client requires more than one new symmetric key, and each key is required to be of a  
 1594 different key class, there MUST be only one <GlobalKeyID> element followed by as many  
 1595 <KeyClass> elements inside the <KeyClasses> set, as needed by the client application.

1596

1597 When a client requires multiple symmetric keys of two or more key classes, the client MUST send  
 1598 multiple requests to the **SKS** server. See examples 4 and 5 below in this section.

1599

1600 The <KeyClasses> and <KeyClass> elements and their respective types are specified in Section  
 1601 4.3.

1602 <X509EncryptionCertificate> [Optional]

1603

1604 This element of type **X509EncryptionCertificateType**, when specified, identifies a PKI X509-  
 1605 compliant digital certificate whose corresponding private-key is owned/authorized for use by the  
 1606 requesting client. The <X509EncryptionCertificate> MUST meet the following requirements:



- 1607 a) It MUST be a valid X509 v3 digital certificate whose expiry is not earlier than the date  
1608 and time the symmetric-key request is received by the **SKS** server;
- 1609 b) It MUST not be revoked by the Certificate Authority (CA) who issued the encryption  
1610 certificate at the date and time the symmetric-key request is received by the **SKS** server;
- 1611 c) It MUST have its *keyEncipherment* bit set in the *keyUsage* extension of the  
1612 certificate.

1613 Note: The **SKS** server will use the specified **X509EncryptionCertificate** if the security policy on  
1614 the **SKS** server permits it. The security policy may have specified that only encryption certificates  
1615 stored on the **SKS** server database or in an LDAP Directory known to the server be used; in  
1616 which case the SKS server will ignore the encryption certificate in the  
1617 **X509EncryptionCertificate** element and use what is stored on the **SKS** server or other location  
1618 known to the server. In the event the **SKS** server uses an encryption certificate stored on the  
1619 server-side, it is assumed that the requesting client has the corresponding private-key to decrypt  
1620 the payload when extracted from the response.

1621  
1622 The <X509EncryptionCertificate> and its respective type is specified in Section 4.4.

## 1623 2. <SymkeyRequestID> [Required]

1624  
1625 The second sequence choice consists of the <SymkeyRequestID> element of type  
1626 **SymkeyRequestIDType**. It identifies the unique symmetric-key request identifier within the target  
1627 Symmetric Key Management System (SKMS). If this sequence is chosen, there MUST be at  
1628 least one <SymkeyRequestID> element in a <SymkeyRequest>, but there may be an unbounded  
1629 (unlimited) number of <SymkeyRequestID> elements specified.

1630  
1631 The presence of the <SymkeyRequestID> element in the <SymkeyRequest> implies that the  
1632 client had previously made a <SymkeyRequest> asynchronously, and instead of receiving a  
1633 <Symkey> or a <SymkeyError>, it received a <SymkeyWorkInProgress> response with a  
1634 <SymkeyRequestID> in it. The client is now following-up with the SKS Server to get an update on  
1635 its earlier request with the <SymkeyRequestID>.

1636  
1637 The <SymkeyRequestID> element and the **SymkeyRequestIDType** is specified in Section 4.5.

1638 Some examples of the <SymkeyRequest> element are as follows:

### 1639 Example 1 – A single new symmetric key request of a default key class:

```
1640 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1641   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1642 </ekmi:SymkeyRequest>
```

### 1643 Example 2 – A request for three new symmetric keys of a default key class for each symmetric 1644 key:

```
1645 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1646   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1647   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1648   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1649 </ekmi:SymkeyRequest>
```

### 1650 Example 3 – A request for a single new symmetric key of a specific key class:

```
1651 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
1652   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
1653   <ekmi:KeyClasses>  
1654     <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
```



```

1707 hvcNAQELBQADggEBACK05PtvZD4WPgl0e+EHUiApzFyCdRzf0pFZtxRwG9lR1PZUWUjmwTNfGFsL
1708 S6kyoHgUfVa5fpT1EU1mXUB/Lmo3hFGyprZjfmD7DwuBcYgmZhv7yHrmGOMIOXjFTACvHpM0v0ce
1709 hVx2e4VE0yhBLu/LdH9awGGDp6Bk2XzxqQcs8y6Zz0XZAnPgKQZdjbfKERSsy/d1D8pk5baBk4bd
1710 Zh5680caUrbm9ZReRVTVaY5qi0pk0U+tDrBSj/HIL6GAqegYllkz6KYCy6RV0y6iVVSjHocDqdJr
1711 EVOR+ds6xn8mmodlERILmuxiLpibPp609SfnDIXNlzLwe5g7ep3lc=
1712 </ekmi:X509EncryptionCertificate>
1713 </ekmi:SymkeyRequest>

```

#### 1714 **Example 9 – A new symmetric key request with a specific key class and the** 1715 **X509EncryptionCertificate:**

```

1716 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1717   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
1718   <ekmi:KeyClasses>
1719     <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
1720   </ekmi:KeyClasses>
1721   <ekmi:X509EncryptionCertificate>
1722     MIIDfDCCAmSgAwIBAgIIAe/AvliGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdu3Ryb25n
1723     S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvciBTdHJvbmdLZXkgREVNTyBVc2Ug
1724     T25seTEwMDU3Ryb25nQXV0aCBjbWwHcNMDYwNzI1MTcxMDMwWhcNMDcwNzI1MTcy
1725     MDMwWjBtMREwDwYKZCIzImZyPzYKQBARMBMjE2MjcGA1UEAxMQUE9TIFJlZ2lzdGVyIDYmYjE2MjcGA1
1726     A1UECxMmRm9yIFN0cm9uZ0tleSBERU1PIFVzZSBPbm5MRcwFQYDVQQKEw5TdHJvbmdBdXR0IElu
1727     YzCBnzANBgkqhkiG9w0BAQEFAA0BJQAwYkCgYEAyAmzMZHA8wHJ4UE4b61s51JVWe4Fygj4MCf
1728     U7LA3JhpUS4TlX0XFwqrcmltL0iVG7YBFarJFluBFJW2X6q8FuvUprv4V9nJrgiwAPtkiRyIx96n
1729     qKXIXkUlQ4idlEgIAZI9dEdf4Y5cqBBCygPyNBoTudglM7R47AjR4nr4ks8CAwEAAaOBqTCBpjAO
1730     BgNVHQ8BAf8EBAMCBLAwHQYDVR00BBYEF0IOrrWrZo0LdBRLLVncRAwLBqVZpCMB8GA1UdIwQYMBAA
1731     FPTYwEHoJG4iFVHRnt2EWxGluAQVMBGGA1UdIAQRMA8wDQYLLKwYEAAdISg30BBAEwOgYDVR0fBDMw
1732     MTAvC2Gk4YpaHR0cDovL2RlbW8uc3Ryb25na2V5Lm9yZy9kZW1vLXN1Yi1jYS5jcmmwDQYJKoZI
1733     hvcNAQELBQADggEBACK05PtvZD4WPgl0e+EHUiApzFyCdRzf0pFZtxRwG9lR1PZUWUjmwTNfGFsL
1734     S6kyoHgUfVa5fpT1EU1mXUB/Lmo3hFGyprZjfmD7DwuBcYgmZhv7yHrmGOMIOXjFTACvHpM0v0ce
1735     hVx2e4VE0yhBLu/LdH9awGGDp6Bk2XzxqQcs8y6Zz0XZAnPgKQZdjbfKERSsy/d1D8pk5baBk4bd
1736     Zh5680caUrbm9ZReRVTVaY5qi0pk0U+tDrBSj/HIL6GAqegYllkz6KYCy6RV0y6iVVSjHocDqdJr
1737     EVOR+ds6xn8mmodlERILmuxiLpibPp609SfnDIXNlzLwe5g7ep3lc=
1738   </ekmi:X509EncryptionCertificate>
1739 </ekmi:SymkeyRequest>

```

## 1740 **4.2 Element <GlobalKeyID>**

1741 The <GlobalKeyID> element is the unique identifier of a symmetric encryption key within an SKMS.  
1742 Every symmetric key generated by the **SKS** server MUST be assigned a unique <GlobalKeyID> as  
1743 specified in this section.

### 1744 **Schema Definition:**

```

1745 <xsd:simpleType name="GlobalKeyIDType">
1746   <xsd:restriction base="xsd:string">
1747     <xsd:minLength value="5"/>
1748     <xsd:maxLength value="62"/>
1749     <xsd:pattern value="[0-9]{1,20}-[0-9]{1,20}-[0-9]{1,20}"/>
1750     <xsd:whiteSpace value="collapse"/>
1751   </xsd:restriction>
1752 </xsd:simpleType>

```

1753 The <GlobalKeyID> element is of the **GlobalKeyIDType**, and is a string identifier of a symmetric key  
1754 consisting of five parts concatenated together:

- 1755 1. A positive integer identifying the **Domain ID**. The **DomainID** identifies the IANA-issued Private  
1756 Enterprise Number (PEN) as published at <http://www.iana.org/assignments/enterprise-numbers>  
1757 and is used by the **SKS** server to constrain the ownership of objects within the SKMS:

1758        2. A literal hyphen ("-") without surrounding spaces;

1759        3. A positive integer identifying the Server ID of the server;

1760        4. Another literal hyphen ("-") without surrounding spaces;

1761        5. A positive integer identifying the unique Key ID;

1762        Combined, the five components of this element make up a unique identifier for a symmetric key within the

1763        SKMS. Since all enterprise are expected to use only the PENs assigned to them, and assuming they do,

1764        the <GlobalKeyID> is unique across the internet.

1765        The **DomainID** part of the <GlobalKeyID> element MUST be a positive integer in the range of 0 (zero) to

1766        18446744073709551615 (20-byte ASCII decimal).

1767        When an SKMS manages the symmetric keys for a single enterprise, the **DomainID** part of the

1768        <GlobalKeyID> element in a <SymkeyRequest> MAY be zero ("0"). When an SKMS manages symmetric

1769        keys for multiple enterprises, the **DomainID** in the <GlobalKeyID> of a <SymkeyRequest> MUST be

1770        positive and non-zero. In such a situation, the client application will request a symmetric key for the

1771        domain in which it is authorized to request and receive keys.

1772        The **DomainID** in the <GlobalKeyID> element of a <SymkeyResponse> MUST always be positive and

1773        non-zero. It will typically contain the PEN of the domain to which the symmetric key belongs.

1774        The **ServerID** part of the <GlobalKeyID> element MUST be a positive integer in the range of 0 (zero) to

1775        18446744073709551615 (20-byte ASCII decimal).

1776        The **ServerID** part of the <GlobalKeyID> element of a <SymkeyRequest> MUST always be zero ("0").

1777        The **ServerID** part of the <GlobalKeyID> element of a <SymkeyResponse> MUST always be positive and

1778        non-zero. It will typically contain the unique server identifier of the **SKS** server where the symmetric key

1779        was generated.

1780        The **KeyID** part of the <GlobalKeyID> element MUST be a positive integer in the range of 0 (zero) to

1781        18446744073709551615 (20-byte ASCII decimal).

1782        The **KeyID** part of the <GlobalKeyID> element of a <SymkeyRequest> MUST always be zero ("0").

1783        The **KeyID** part of the <GlobalKeyID> element of a <SymkeyResponse> MUST always be positive and

1784        non-zero. It will typically contain the unique key identifier of the symmetric key within the **SKS** server

1785        where the key was generated.

1786        **Example 1 – A <GlobalKeyID> value for a new symmetric key from an SKMS that serves a single**

1787        **domain:**

1788               <ekmi:GlobalKeyID>**0-0-0**</ekmi:GlobalKeyID>

1789

1790        **Example 2 – A <GlobalKeyID> value for a new symmetric key for the domain with the PEN 10514,**

1791        **from an SKMS that serves multiple domains:**

1792               <ekmi:GlobalKeyID>**10514-0-0**</ekmi:GlobalKeyID>

1793

1794        **Example 3 – A <GlobalKeyID> value for the 16,777,215th symmetric key generated on 2<sup>nd</sup> SKS**

1795        **server for an enterprise with the PEN 10514, in either a <SymkeyRequest> or a <SymkeyResponse>:**

1796

1797               <ekmi:GlobalKeyID>**10514-2-16777215**</ekmi:GlobalKeyID>

1798

1799 **Example 4 – The maximum <GlobalKeyID> value possible (a 62-byte ASCII decimal), in a**  
1800 **<SymkeyRequest> or <SymkeyResponse>:**

1801     <ekmi:GlobalKeyID>

1802

1803     **18446744073709551615-18446744073709551615-18446744073709551615**

1805     </ekmi:GlobalKeyID>

1806

## 1807 **4.3 Element <KeyClasses> and <KeyClass>**

1808 The <KeyClasses> element of type **KeyClassesType**, when specified, identifies at least one  
1809 <KeyClass> element, but may specify an unbounded (unlimited) number of <KeyClass> elements within  
1810 the <KeyClasses> set.

1811

### 1812 **Schema Definition:**

```
1813 <xsd:complexType name="KeyClassesType">
```

```
1814 <xsd:sequence>
```

```
1815 <xsd:element
```

```
1816     name="KeyClass"
```

```
1817     type="ekmi:KeyClassType"
```

```
1818     minOccurs="1"
```

```
1819     maxOccurs="unbounded"/>
```

```
1820 </xsd:sequence>
```

```
1821 </xsd:complexType>
```

```
1822 <xsd:simpleType name="KeyClassType">
```

```
1823 <xsd:restriction base="xsd:string">
```

```
1824 <xsd:maxLength value="255"/>
```

```
1825 </xsd:restriction>
```

```
1826 </xsd:simpleType>
```

1827 Client applications may request one or more symmetric keys conforming to one or more key classes  
1828 required by the application. If the client application is authorized to receive keys conforming to such key  
1829 classes, the SKS server will generate and supply them.

1830

1831 The <KeyClasses> element is useful only when requesting new symmetric keys, i.e. symmetric  
1832 encryption keys that have previously NOT been used for encrypting data. There is little reason for a  
1833 client application to specify the <KeyClasses> element when requesting an existing (escrowed)  
1834 symmetric key. This is because the SKS server will return the requested key to authorized clients with  
1835 whatever key class is associated with the key, regardless of what key class is specified in the request.  
1836 The key class will have been associated with the symmetric key at the time of its generation and cannot  
1837 be changed once associated with a key.

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

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

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

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

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

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

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

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

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

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

```
<ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
  <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
  <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
  <ekmi:KeyClasses>  
    <ekmi:KeyClass>FIN-FX</ekmi:KeyClass>  
  </ekmi:KeyClasses>  
</ekmi:SymkeyRequest>
```

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

```

1885     <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1886       <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
1887       <ekmi:KeyClasses>
1888         <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
1889         <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
1890         <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
1891         <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
1892       </ekmi:KeyClasses>
1893     </ekmi:SymkeyRequest>
1894

```

#### 1895 4.4 Element <X509EncryptionCertificate>

1896 The <X509EncryptionCertificate> element of type *X509EncryptionCertificateType*, when specified,  
 1897 contains a PKI X509-compliant digital certificate.

1898

##### 1899 Schema Definition:

```

1900     <xsd:simpleType name="X509EncryptionCertificateType">
1901       <xsd:restriction base="xsd:base64Binary"/>
1902     </xsd:simpleType>
1903

```

1903

1904 The <X509EncryptionCertificate> element appears only within a <SymkeyRequest> element.  
 1905 There SHALL be only one <X509EncryptionCertificate> element in a <SymkeyRequest>. The  
 1906 content of the <X509EncryptionCertificate> element is restricted to being **base64Binary** from the  
 1907 XML Schema Definition types.

1908

##### 1909 Example 1 – A symmetric key request with the X509EncryptionCertificate sent by the client to the 1910 server:

```

1911     <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
1912       <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
1913       <ekmi:X509EncryptionCertificate>
1914         MIIIDfDCCAmSgAwIBAgIIAe/AvliGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdU3Ryb25n
1915         S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvciBTdHJvbmdLZXkgREVNTyBVc2Ug
1916         T25seTEwMBUGA1UEChM0U3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTcxMDMwWhcNMDcwNzI1MTcy
1917         MDMwWjBtMREwDwYKCCZImiZPyLGQBARMBMjEzBzBpbnM5MRcwFQYDVQQKEw5TdHJvbmdBdXR0IElu
1918         A1UECXMbRm9yIFN0cm9uZ0tleSBERU1PIFVzZSBPbnM5MRcwFQYDVQQKEw5TdHJvbmdBdXR0IElu
1919         YzCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYKCGYEAyAmxMZhYA8wHJ4UE4b61s51JVWe4Fygj4MCf
1920         U7LA3JhpUS4TLX0XFwqrcmltL0iVG7YBFarJFluBFJW2X6q8FuvUprv4V9nJrgiWAPtkiRyIx96n
1921         qKXIxkU1Q4idlEg1AZI9dEdf4Y5cqBBCygPyNBoTudgLM7R47AjR4nr4ks8CAwEAAa0BqTCBpjAO
1922         BgNVHQ8BAf8EBAMCBLAwHQYDVR00BBYEF0IOwrZo0LdBRLLVncRAwLBqVZpCMB8GA1UdIwQYMBAA
1923         FPTYWEHoJG4iFVHRnt2EWxGluAQVMBGGA1UdIAQRMA8wDQYLKwYEAAdISg30BBAEW0gYDVR0fBDMw
1924         MTAvoC2gK4YpaHR0cDovL2RlbW8uc3Ryb25na2V5Lm9yZy9kZW1vLXN1Yi1jYS5jcmwwDQYJKoZI
1925         hvcNAQELBQADggEBAK05PtvZD4WPgl0e+EHUiApzFyCdRzf0pFZtxRwG9lR1PZUWUjmwTNfGFsL
1926         S6kyoHgUfVa5fpT1EU1mXUB/Lmo3hFGyprZjfmD7DwuBcYgmZhv7yHrmGOMIOXjFTACvHpM0vOce
1927         hVx2e4VE0yhBLu/LdH9awGGDp6Bk2XzxqQcs8y6Zz0XZAnPgKQZdjbfKERSsy/d1D8pk5baBk4bd
1928         Zh5680caUrbm9ZReRVTVaY5qiQpk0U+tdrBsJ/HIL6GAqegYllkz6KYCy6RV0y6iVVSjHocDqdJr
1929         EVOR+ds6xn8mmodlERrILmuxiLpibPp609SfnDIxNlLwe5g7ep3lc=
1930       </ekmi:X509EncryptionCertificate>
1931     </ekmi:SymkeyRequest>

```



## 4.5 Element <SymkeyRequestID>

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

### Schema Definition:

```
<xsd:simpleType name="SymkeyRequestIDType">
  <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 <SymkeyRequestID> element is of the **SymkeyRequestIDType**, and is a string identifier 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 unique Server ID of the server within the SKMS of the above-mentioned domain, that originally received the request;
4. Another literal hyphen ("-") without surrounding spaces;
5. A positive integer identifying the unique Request ID on the server that received the request;

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

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

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

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

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

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

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

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

```
<ekmi:SymkeyRequestID>  
18446744073709551615-18446744073709551615-18446744073709551615  
</ekmi:SymkeyRequestID>
```

## 4.6 Element <SymkeyResponse>

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

### Schema Definition:

```
<xsd:element name="SymkeyResponse">  
  <xsd:complexType>  
    <xsd:choice>  
      <xsd:sequence>  
        <xsd:element  
          name="Symkey"  
          type="ekmi:SymkeyType"  
          minOccurs="1"  
          maxOccurs="unbounded"/>  
        <xsd:element  
          name="SymkeyWorkInProgress"  
          type="ekmi:SymkeyWorkInProgressType"  
          minOccurs="0"  
          maxOccurs="unbounded"/>  
        <xsd:element  
          name="SymkeyError"  
          type="ekmi:SymkeyErrorType"  
          minOccurs="0"  
          maxOccurs="unbounded"/>  
      </xsd:sequence>  
    </xsd:complexType>  
  </xsd:element>
```

```

2006         </xsd:sequence>
2008
2009         <xsd:sequence>
2010             <xsd:element
2011                 name="SymkeyWorkInProgress"
2012                 type="ekmi:SymkeyWorkInProgressType"
2013                 minOccurs="1"
2014                 maxOccurs="unbounded"/>
2015             <xsd:element
2016                 name="SymkeyError"
2017                 type="ekmi:SymkeyErrorType"
2018                 minOccurs="0"
2019                 maxOccurs="unbounded"/>
2020         </xsd:sequence>
2022
2023         <xsd:sequence>
2024             <xsd:element
2025                 name="SymkeyError"
2026                 type="ekmi:SymkeyErrorType"
2027                 minOccurs="1"
2028                 maxOccurs="unbounded"/>
2029         </xsd:sequence>
2030     </xsd:choice>
2031 </xsd:complexType>
2032
2031 </xsd:element>

```

2033 The <SymkeyResponse> element consists of a choice of one of three sequences of children elements:

- 2034 • A sequence of a required <Symkey> element followed with an optional <SymkeyWorkInProgress>  
2035 and/or an optional <SymkeyError> element;
- 2036 • A sequence of a required <SymkeyWorkInProgress> element followed by an optional  
2037 <SymkeyError> element; or
- 2038 • A required <SymkeyError> element .

2039 In any <SymkeyResponse> element that consists of the first two choices, all <Symkey> elements MUST  
2040 precede the first <SymkeyWorkInProgress> element and all <SymkeyWorkInProgress> elements MUST  
2041 precede the first <SymkeyError> element..

#### 2042 1. <Symkey> [Required]

2043  
2044 This element of type **SymkeyType**, is returned by the **SKS** server in response to a successful  
2045 processing of a <SymkeyRequest>. There MAY be more than one <Symkey> element in the  
2046 <SymkeyResponse> if the client application made a request for multiple symmetric keys.

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

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

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

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

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

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

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

When the <SymkeyResponse> element contains at least one <Symkey> or <SymkeyWorkInProgress> element, the <SymkeyError> is an optional element; otherwise the <SymkeyError> is required.

The <SymkeyError> element and **SymkeyErrorType** are specified in Section 4.9.

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

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

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

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

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

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

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

```
2094     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2095 </ekmi:SymkeyResponse>
```

2096

2097 **Example 4 – A response with multiple work-in-progress elements:**

```
2098 <ekmi:SymkeyResponse
2099     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2100     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2101     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2102     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2103 </ekmi:SymkeyResponse>
```

2104

2105 **Example 5 – A response with multiple symmetric-key and work-in-progress elements:**

```
2106 <ekmi:SymkeyResponse
2107     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2108     <ekmi:Symkey>.....</ekmi:Symkey>
2109     <ekmi:Symkey>.....</ekmi:Symkey>
2110     <ekmi:Symkey>.....</ekmi:Symkey>
2111     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2112     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2113     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2114 </ekmi:SymkeyResponse>
```

2115

2116 **Example 6 – A response with an error:**

```
2117 <ekmi:SymkeyResponse
2118     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2119     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2120 </ekmi:SymkeyResponse>
```

2121

2122 **Example 7 – A response with multiple errors:**

```
2123 <ekmi:SymkeyResponse
2124     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2125     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2126     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2127 </ekmi:SymkeyResponse>
```

2128

2129 **Example 8 – A response with one symmetric key and one error:**

```
2130 <ekmi:SymkeyResponse
2131     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2132     <ekmi:Symkey>.....</ekmi:Symkey>
2133     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2134 </ekmi:SymkeyResponse>
```

2135

2136 **Example 9 – A response with multiple symmetric keys and multiple errors:**

```

2137     <ekmi:SymkeyResponse
2138         xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2139         <ekmi:Symkey>.....</ekmi:Symkey>
2140         <ekmi:Symkey>.....</ekmi:Symkey>
2141         <ekmi:Symkey>.....</ekmi:Symkey>
2142         <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2143         <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2144     </ekmi:SymkeyResponse>

```

2145

2146 **Example 10 – A response with multiple symmetric keys, multiple work-in-progress and multiple**  
2147 **error elements:**

```

2148     <ekmi:SymkeyResponse
2149         xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2150         <ekmi:Symkey>.....</ekmi:Symkey>
2151         <ekmi:Symkey>.....</ekmi:Symkey>
2152         <ekmi:Symkey>.....</ekmi:Symkey>
2153         <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2154         <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2155         <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
2156         <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2157         <ekmi:SymkeyError>.....</ekmi:SymkeyError>
2158     </ekmi:SymkeyResponse>

```

2159

## 2160 4.7 Element <Symkey>

2161 The <Symkey> element is the *raison d'être* of the **SKSML** protocol. The element of type **SymkeyType**,  
2162 contains the symmetric key returned by the **SKS** server, in response to a successful processing of a  
2163 <SymkeyRequest> from a client application.

### 2164 Schema Definition:

```

2165     <xsd:complexType name="SymkeyType">
2166         <xsd:sequence>
2167             <xsd:element name="SymkeyRequestID" type="ekmi:SymkeyRequestID"/>
2168             <xsd:element name="GlobalKeyID" type="ekmi:GlobalKeyIDType"/>
2169             <xsd:element name="KeyUsePolicy" type="ekmi:KeyUsePolicyType"/>
2170             <xsd:element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>
2171             <xsd:element ref="xenc:CipherData"/>
2172         </xsd:sequence>
2173     </xsd:complexType>

```

2174

2175 When a request for a symmetric key is successful, there **MUST** be at least one <Symkey> element in a  
2176 <SymkeyResponse> element. There **MAY** be more than one <Symkey> element in the response if the  
2177 client application made a request for multiple symmetric keys and the **SKS** server processed the request  
2178 successfully.

2179 In the event an **SKS** server receives the request asynchronously, the response **SHALL** contain a  
2180 <SymkeyWorkInProgress> element implying that the request is being processed and is pending  
2181 completion. The <SymkeyError> element is specified in Section 4.7.

2182 In the event of an error in processing the request, there **SHALL** be no <Symkey> element in the response;  
2183 there **SHALL** be a <SymkeyError> element, instead. The <SymkeyError> element is specified in Section  
2184 4.9.

2185 The <Symkey> element consists of a sequence of the following child elements:

2186 1. <SymkeyRequestID> [Required]

2187

2188 This element of type **SymkeyRequestIDType** identifies the unique identifier of the request made  
2189 by the client within an SKMS. There SHALL be only one <SymkeyRequestID> within a  
2190 <Symkey> element.

2191

2192 The <SymkeyRequestID> element and the **SymkeyRequestIDType** are specified in Section 4.5.

2193 2. <GlobalKeyID> [Required]

2194

2195 This element of type **GlobalKeyIDType** identifies the unique identifier of the symmetric key within  
2196 an SKMS. There SHALL be only one <GlobalKeyID> within a <Symkey> element.

2197

2198 The <GlobalKeyID> element and the **GlobalKeyIDType** are specified in Section 4.2.

2199 3. <KeyUsePolicy> [Required]

2200

2201 This element of type **KeyUsePolicyType**, defines how the symmetric key in this <Symkey>  
2202 element may be used by applications. There SHALL be only one <KeyUsePolicy> element  
2203 within a <Symkey> element.

2204

2205 The <KeyUsePolicy> element and **KeyUsePolicyType** are specified in Section 4.10.

2206 4. <EncryptionMethod> [Required]

2207

2208 This element of type **EncryptionMethodType** from [XMLEncryption] describes how the  
2209 symmetric key in this <Symkey> element is encrypted for transport between the SKS Server and  
2210 the client application.

2211

2212 The <EncryptionMethod> MUST specify one of the following two transport algorithms in the  
2213 **Algorithm** attribute of the element:

2214

2215 - [http://www.w3.org/2001/04/xmlenc#rsa-1\\_5](http://www.w3.org/2001/04/xmlenc#rsa-1_5)

2216 - <http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p>

2217 5. <CipherData> [Required]

2218

2219 This element of **CipherDataType** from [XMLEncryption] contains the encrypted symmetric-key.  
2220 As specified in [XMLEncryption], the content of this element is Base-64 encoded and is of the  
2221 XML Schema **base64Binary** type.

2222 Some high-level examples of the <Symkey> element are as follows. Details about the <KeyUsePolicy>  
2223 element have been elided for brevity:

2224 **Example 1 – A response with a symmetric key:**

```
2225 <ekmi:SymkeyResponse
2226   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2227   <ekmi:Symkey>
2228     <ekmi:SymkeyRequestID>10514-1-7455</ekmi:SymkeyRequestID>
2229     <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
2230     <ekmi:KeyUsePolicy>....</ekmi:KeyUsePolicy>
2231     <ekmi:EncryptionMethod
2232       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
2233     <xenc:CipherData>
2234       <xenc:CipherValue>
2235         E9zWB/y93hVSzeTLiDcQoDxm1NXTux+SffMNwCJmt1dIqzQHBnpdQ8
```



```

2236         1g6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxj
2237         fg1pU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
2238     </xenc:CipherValue>
2239 </xenc:CipherData>
2240 </ekmi:Symkey>
2241 </ekmi:SymkeyResponse>

```

#### 2242 Example 2 – A response with multiple symmetric keys:

```

2243 <ekmi:SymkeyResponse
2244   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
2245   <ekmi:Symkey>
2246     <ekmi:SymkeyRequestID>10514-1-12455</ekmi:SymkeyRequestID>
2247     <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
2248     <ekmi:KeyUsePolicy>....</ekmi:KeyUsePolicy>
2249     <ekmi:EncryptionMethod
2250       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgflp"/>
2251   <xenc:CipherData>
2252     <xenc:CipherValue>
2253       E9zWB/y93hVSzeTLiDcQoDxm1NxTux+SffMNwCJmt1dIqzQHBnpgQ8
2254       1g6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxj
2255       fg1pU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
2256     </xenc:CipherValue>
2257   </xenc:CipherData>
2258 </ekmi:Symkey>
2259 <ekmi:Symkey>
2260   <ekmi:SymkeyRequestID>10514-1-12467</ekmi:SymkeyRequestID>
2261   <ekmi:GlobalKeyID>10514-1-236</ekmi:GlobalKeyID>
2262   <ekmi:KeyUsePolicy>....</ekmi:KeyUsePolicy>
2263   <ekmi:EncryptionMethod
2264     Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgflp"/>
2265   <xenc:CipherData>
2266     <xenc:CipherValue>
2267       Qbg65cy93hVSzeTLiDcQoDxm1NxTux+SffMNwCJmt1dIqzQHBnpgQ8
2268       7k6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxj
2269       uyecU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
2270     </xenc:CipherValue>
2271   </xenc:CipherData>
2272 </ekmi:Symkey>
2273 </ekmi:SymkeyResponse>

```

## 2274 4.8 Element <SymkeyWorkInProgress>

2275 The <SymkeyWorkInProgress> element is returned in a <SymkeyResponse> when a client either makes  
 2276 an asynchronous request for a symmetric-key from an SKS server.

2277 The <SymkeyWorkInProgress> element of type **SymkeyWorkInProgressType**, contains a unique  
 2278 request ID returned by the **SKS** server, in response to receiving a <SymkeyRequest> from a client  
 2279 application.

#### 2280 Schema Definition:

```

2281   <xsd:complexType name="SymkeyWorkInProgressType">
2282     <xsd:sequence>
2283       <xsd:element
2284         name="RequestedGlobalKeyID"
2285         type="ekmi:GlobalKeyIDType"

```

```

2286         minOccurs="1"
2287         maxOccurs="1">
2288     </xsd:element>
2289     <xsd:element
2290         name="RequestedKeyClass"
2291         type="ekmi:KeyClassType"
2292         minOccurs="0"
2293         maxOccurs="1">
2294     </xsd:element>
2295     <xsd:element
2296         name="SymkeyRequestID"
2297         type="ekmi:SymkeyRequestIDType"
2298         minOccurs="1"
2299
2300         maxOccurs="1">
2301     </xsd:element>
2302     <xsd:element
2303         name="RequestCheckInterval"
2304         type="ekmi:RequestCheckIntervalType"
2305         minOccurs="1"
2306         maxOccurs="1">
2307     </xsd:element>
2308 </xsd:sequence>
2309 </xsd:complexType>

```

2310 **Schema Definition:**

```

2311     <xsd:simpleType name="RequestCheckIntervalType">
2312         <xsd:restriction base="xsd:positiveInteger">
2313             <xsd:minInclusive value="60"/>
2314             <xsd:maxInclusive value="86400"/>
2315         </xsd:restriction>
2316     </xsd:simpleType>

```

2317 The <SymkeyWorkInProgress> element consists of a sequence of the following child elements:

- 2318 1. <RequestedGlobalKeyID> [Required]

2319

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

2322 <SymkeyWorkInProgress> element.

2323

2324 The **GlobalKeyIDType** is specified in Section 4.2.

2325 2. <RequestedKeyClass> [Optional]

2326

2327 This element of type **KeyClassType**, identifies the class of the symmetric-key that was  
2328 requested, if any. If present, there SHALL be only one <RequestedKeyClass> element within a  
2329 <SymkeyWorkInProgress> element.

2330

2331 The **KeyClassType** is specified in Section 4.3.

2332 3. <SymkeyRequestID> [Required]

2333

2334 This element of type **SymkeyRequestIDType** identifies the unique request identifier returned by  
2335 the **SKS** server. This request identifier may be used by the client to query the **SKS** server for an  
2336 update on the request at a future time.

2337

2338 The <SymkeyRequestID> element and **SymkeyRequestIDType** are specified in Section 4.5.

2339 4. <RequestCheckInterval> [Required]

2340

2341 This element of **RequestCheckIntervalType** defines the number of seconds a client MUST wait  
2342 after it has received a <SymkeyWorkInProgress> response before it may query the **SKS** server  
2343 for an update on the request. The value also indicates that the requesting client MUST wait the  
2344 same interval between update queries in the event it continues to get a  
2345 <SymkeyWorkInProgress> response from the server.

2346 The **RequestCheckIntervalType** is a positiveInteger type from the XML Schema Definition  
2347 and restricts the minimum value to be 60 seconds, and the maximum value to be 604,800  
2348 seconds (1-week).

2349 Some high-level examples of the <SymkeyWorkInProgress> element are as follows.:

2350 **Example 1 – A response with a work-in-progress element and a check-request frequency interval**  
2351 **of 5-minutes:**

```
2352 <ekmi:SymkeyResponse xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
2353   <ekmi:SymkeyWorkInProgress>  
2354     <ekmi:RequestedGlobalKeyID>10514-1-235</ekmi:RequestedGlobalKeyID>  
2355     <ekmi:SymkeyRequestID>10514-1-17964</ekmi:SymkeyRequestID>  
2356     <ekmi:RequestCheckInterval>300</ekmi:RequestCheckInterval>  
2357   </ekmi:SymkeyWorkInProgress>  
2358 </ekmi:SymkeyResponse>
```

2359 **Example 2 – A response with a work-in-progress element and a check-request frequency interval**  
2360 **of 30-minutes:**

```
2361 <ekmi:SymkeyResponse xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
2362   <ekmi:SymkeyWorkInProgress>  
2363     <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>  
2364     <ekmi:RequestedKeyClass>HR-Class</ekmi:RequestedKeyClass>  
2365     <ekmi:SymkeyRequestID>10514-3-4955</ekmi:SymkeyRequestID>  
2366     <ekmi:RequestCheckInterval>1800</ekmi:RequestCheckInterval>  
2367   </ekmi:SymkeyWorkInProgress>  
2368 </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:

```
<xsd:complexType name="SymkeyErrorType">
  <xsd:sequence>
    <xsd:element
      name="SymkeyRequestID" type="ekmi:SymkeyRequestIDType">
    <xsd:element
      name="RequestedGlobalKeyID" type="ekmi:GlobalKeyIDType"/>
    <xsd:element
      name="RequestedKeyClass"
      type="ekmi:KeyClassType"
      minOccurs="0"/>
    <xsd:element name="ErrorCode">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="255"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="ErrorMessage">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="1024"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

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

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

1. <SymkeyRequestID> [Required]

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

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

2. <RequestedGlobalKeyID> [Required]

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

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

3. <RequestedKeyClass> [Optional]

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

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

4. <ErrorCode> [Required]

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

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

5. <ErrorMessage> [Required]

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

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

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

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

```
<ekmi:SymkeyResponse
  xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
  <ekmi:SymkeyError>
    <ekmi:SymkeyRequestID>10514-2-34345</ekmi:SymkeyRequestID>
    <ekmi:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>
    <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
    <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
  </ekmi:SymkeyError>
</ekmi:SymkeyResponse>
```

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

```

<ekmi:SymkeyResponse
  xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
  <ekmi:SymkeyError>
    <ekmi:SymkeyRequestID>10514-1-4564345</ekmi:SymkeyRequestID>
    <ekmi:RequestedGlobalKeyID>10514-1-0</ekmi:RequestedGlobalKeyID>
    <ekmi:ErrorCode>SKS-100001</ekmi:ErrorCode>
    <ekmi:ErrorMessage>Invalid GlobalKeyID</ekmi:ErrorMessage>
  </ekmi:SymkeyError>
  <ekmi:SymkeyError>
    <ekmi:SymkeyRequestID>10514-1-4564349</ekmi:SymkeyRequestID>
    <ekmi:RequestedGlobalKeyID>10514-1-7522</ekmi:RequestedGlobalKeyID>
    <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
    <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
  </ekmi:SymkeyError>
</ekmi:SymkeyResponse>

```

2446

## 2447 4.10 Element <KeyUsePolicy>

2448 The <KeyUsePolicy> element defines rules that conforming implementations of the **SKCL** MUST adhere  
 2449 to when using the symmetric key sent by the **SKS** Server. It is an integral part of the <Symkey> element.

2450

### 2451 Schema Definition:

```

2452 <xsd:complexType name="KeyUsePolicyType" mixed="true">
2453   <xsd:sequence>
2454     <xsd:element name="KeyUsePolicyID" type="ekmi:TwoPartIDType"/>
2455     <xsd:element name="PolicyName">
2456       <xsd:simpleType>
2457         <xsd:restriction base="xsd:string">
2458           <xsd:maxLength value="255"/>
2459         </xsd:restriction>
2460       </xsd:simpleType>
2461     </xsd:element>
2462     <xsd:element name="KeyClass" type="ekmi:KeyClassType"/>
2463     <xsd:element name="KeyAlgorithm" type="ekmi:EncryptionAlgorithmType"/>
2464     <xsd:element name="KeySize" type="ekmi:KeySizeType"/>
2465     <xsd:element name="Status" type="ekmi:StatusType"/>
2466     <xsd:element name="Permissions" type="ekmi:PermissionsType"/>
2467   </xsd:sequence>
2468 </xsd:complexType>

```

2469 The <KeyUsePolicy> element is of the **KeyUsePolicyType** and consists of the following child elements:

2470 1. <KeyUsePolicyID> [Required]

2471  
2472 The <KeyUsePolicyID> element, of type **TwoPartIDType**, identifies the unique policy object  
2473 within the SKMS. There SHALL be only one <KeyUsePolicyID> element within a  
2474 <KeyUsePolicy> element.

2475  
2476 The **TwoPartIDType** is specified in Section 4.11.

2477 2. <PolicyName> [Required]

2478  
2479 The <PolicyName> element, of type XSD **String**, with a maximum length of 255 characters,  
2480 identifies a unique name given to this <KeyUsePolicy>. There SHALL be only one  
2481 <PolicyName> element within a <KeyUsePolicy> element.

2482 3. <KeyClass> [Required]

2483  
2484 The <KeyClass> element, of type **KeyClassType**, identifies a key-class assigned to this  
2485 <KeyUsePolicy>. There SHALL be only one <KeyUsePolicyID> element within a  
2486 <KeyUsePolicy> element.

2487  
2488 The **KeyClassType** is specified in Section 4.3.

2489 4. <KeyAlgorithm> [Required]

2490  
2491 The <KeyAlgorithm> element, of type **EncryptionAlgorithmType**, identifies encryption  
2492 algorithm to be used by applications when using this symmetric key. There SHALL be only one  
2493 <KeyAlgorithm> element within a <KeyUsePolicy> element.

2494  
2495 The <KeyAlgorithm> element is specified in Section 4.12.

2496 5. <KeySize> [Required]

2497  
2498 The <KeySize> element, of type **KeySizeType**, defines the size of the symmetric key, in bits  
2499 (binary digits). There SHALL be only one <KeySize> element within a <KeyUsePolicy> element.

2500  
2501 Note: It is possible to determine the size of a symmetric key in an **SKCL** implementation without  
2502 having to send the size in the response. So, why include it? It is our belief that while network  
2503 bandwidth and compute performance of devices are increasing steadily, encryption is desired in  
2504 many small and portable devices. Consequently, it will speed up applications in cryptographic  
2505 processing if they do not have to determine the size of each key they use. While "protocol purity"  
2506 demands that implementation issues do not show up in protocol design, we believe it is justified  
2507 in this case.

2508  
2509 The **KeySizeType** is specified in Section 4.13.

2510 6. <Status> [Required]

2511  
2512 The <Status> element, of type **StatusType**, identifies the current status of the symmetric key.  
2513 There SHALL be only one <Status> element within a <KeyUsePolicy> element.

2514  
2515 The **StatusType** is specified in Section 4.14.

2516 7. <Permissions> [Required]

2517  
2518 The <Permissions> element, of type **PermissionsType**, defines what is permissible to client  
2519 applications with the symmetric key this element is associated with. It is the responsibility of the  
2520 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>
        <ekmi:StartDate>2008-01-01</ekmi:StartDate>
        <ekmi:EndDate>2008-12-31</ekmi:EndDate>
      </ekmi:PermittedDate>
    </ekmi:PermittedDates>
    <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
    <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
    <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
    <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
    <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
    <ekmi:PermittedTimes ekmi:any="false">
      <ekmi:PermittedTime>
        <ekmi:StartTime>07:00:00</ekmi:StartTime>
        <ekmi:EndTime>19:00:00</ekmi:EndTime>
      </ekmi:PermittedTime>
    </ekmi:PermittedTimes>
    <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
  </ekmi:Permissions>
</ekmi:KeyUsePolicy>
```

2534  
2535

2536 **Example 2 – A <KeyUsePolicy> with no restrictions on the key:**

```
2537     <ekmi:KeyUsePolicy>
2538         <ekmi:KeyUsePolicyID>10514-2</ekmi:KeyUsePolicyID>
2539         <ekmi:PolicyName>Laptop KeyUsePolicy</ekmi:PolicyName>
2540         <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
2541         <ekmi:KeyAlgorithm>
2542             http://www.w3.org/2001/04/xmlenc#aes256-cbc
2543         </ekmi:KeyAlgorithm>
2544         <ekmi:KeySize>256</ekmi:KeySize>
2545         <ekmi:Status>Active</ekmi:Status>
2546         <ekmi:Permissions>
2547             <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
2548             <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
2549             <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2550             <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2551             <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2552             <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2553             <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
2554             <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
2555             <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2556         </ekmi:Permissions>
2557     </ekmi:KeyUsePolicy>
```

2558

2559

## 2560 4.11 Type *TwoPartIDType*

2561 The ***TwoPartIDType*** is used to create identifiers for many elements within the **SKSML**. It is a simple  
2562 concatenation of two integers with a hyphen between them ("-") to create an XML Schema ***String*** type.

2563 The ***TwoPartIDType*** has a minimum length of three (3) characters, and a maximum length of 41  
2564 characters.

### 2565 Schema Definition:

```
2566     <xsd:simpleType name="TwoPartIDType">
2567         <xsd:restriction base="xsd:string">
2568             <xsd:minLength value="3"/>
2569             <xsd:maxLength value="41"/>
2570             <xsd:pattern value="[1-9][0-9]{0,19}-[1-9][0-9]{0,19}"/>
2571             <xsd:whiteSpace value="collapse"/>
2572         </xsd:restriction>
2573     </xsd:simpleType>
```

2574 The ***TwoPartIDType*** is used in the <ApplicationID>, the <KeyCachePolicyID> and the  
2575 <KeyUsePolicyID> elements within the **SKSML**.

2576 Some examples of the <KeyUsePolicy> element are as follows.

2577 **Example 1 – A *TwoPartIDType* used to identify an ApplicationID:**

2578       <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>

2579   **Example 2 – A *TwoPartIDType* used to identify a *KeyUsePolicyID*:**

2580       <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>

2581   **Example 3 – A minimum-length *TwoPartIDType* :**

2582       <ekmi:KeyCachePolicyID>5-4</ekmi:KeyCachePolicyID>

2583   **Example 4 – A maximum-length *TwoPartIDType* :**

2584       <ekmi:ApplicationID>  
2585           18446744073709551615-18446744073709551615  
2586       </ekmi:ApplicationID>

## 2587   4.12 Element <KeyAlgorithm>

2588   The element <KeyAlgorithm> , of type ***EncryptionAlgorithmType***, is used to identify the cryptographic  
2589   algorithm to be used with the symmetric keys in the <SymkeyResponse>.

### 2590   Schema Definition:

```
2591   <xsd:simpleType name="EncryptionAlgorithmType">  
2592     <xsd:restriction base="xsd:anyURI">  
2593       <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>  
2594       <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>  
2595       <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes192-cbc"/>  
2596       <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>  
2597     </xsd:restriction>  
2598   </xsd:simpleType>
```

2599   The algorithms currently supported by this specification are the algorithms defined in **[XMLEncryption]**.  
2600   As new algorithms are added to **[XMLEncryption]**, they will be added to the enumerated list in this  
2601   element. Currently, the following four algorithms are supported:

#### 2602   1. Triple Data Encryption Standard (3DES)

2603

2604       Within the context of this specification, and as specified in **[XMLEncryption]**, the form of 3DES  
2605       supported within **SKSML** is a 192-bit key with a 64-bit Initialization Vector. Of the key bits, the  
2606       first 64 are used in the first DES operation, the second 64 bits in the second (middle) DES  
2607       operation, and the third 64 bits in the third (last) DES operation. Each of these 64 bits of key  
2608       contain 56 effective bits and 8 parity bits.

2609

2610       The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#tripledes-cbc>.

#### 2611   2. Advanced Encryption Standard (AES) – 128-bit

2612

2613       Within the context of this specification, and as specified in **[AES]**, this is a 128-bit symmetric key  
2614       used in the Cipher Block Chaining (CBC) mode.

2615

2616       The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#aes128-cbc>.

#### 2617   3. Advanced Encryption Standard (AES) – 192-bit

2618

2619       Within the context of this specification, and as specified in **[AES]**, this is a 192-bit symmetric key  
2620       used in the Cipher Block Chaining (CBC) mode.

2621

2622       The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#aes192-cbc>.

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

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

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

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

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

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

```
<ekmi:KeyUsePolicy>
...
<ekmi:KeyAlgorithm>
http://www.w3.org/2001/04/xmlenc#tripledes-cbc
</ekmi:KeyAlgorithm>
...
</ekmi:KeyUsePolicy>
```

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

```
<ekmi:KeyUsePolicy>
...
<ekmi:KeyAlgorithm>
http://www.w3.org/2001/04/xmlenc#aes128-cbc
</ekmi:KeyAlgorithm>
...
</ekmi:KeyUsePolicy>
```

## 4.13 Element <KeySize>

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

##### Schema Definition:

```
<xsd:simpleType name="KeySizeType">
  <xsd:restriction base="xsd:unsignedShort">
    <xsd:totalDigits value="3"/>
    <xsd:fractionDigits value="0"/>
    <xsd:enumeration value="128"/>
    <xsd:enumeration value="192"/>
    <xsd:enumeration value="256"/>
  </xsd:restriction>
</xsd:simpleType>
```

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

Currently, the following three key-sizes are supported:

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

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

2668 **Example 1 – An example using a 128-bit key size:**

```
2669     <ekmi:KeyUsePolicy>
2670         ...
2671         <ekmi:KeySize>128</ekmi:KeySize>
2672         ...
2673     </ekmi:KeyUsePolicy>
```

2674 **Example 2 – An example using a 192-bit key size:**

```
2675     <ekmi:KeyUsePolicy>
2676         ...
2677         <ekmi:KeySize>192</ekmi:KeySize>
2678         ...
2679     </ekmi:KeyUsePolicy>
```

2680 **Example 3 – An example using a 256-bit key size:**

```
2681     <ekmi:KeyUsePolicy>
2682         ...
2683         <ekmi:KeySize>256</ekmi:KeySize>
2684         ...
2685     </ekmi:KeyUsePolicy>
```

## 2686 **4.14 Element <Status>**

2687 The element <Status>, of type **StatusType**, is used to identify the current status of an object . It is  
2688 used in almost every element within the SKMS.

### 2689 **Schema Definition:**

```
2690     <xsd:simpleType name="StatusType">
2691         <xsd:restriction base="xsd:string">
2692             <xsd:enumeration value="Active"/>
2693             <xsd:enumeration value="Default"/>
2694             <xsd:enumeration value="Inactive"/>
2695             <xsd:enumeration value="Other"/>
2696         </xsd:restriction>
2697     </xsd:simpleType>
```

2698 Where it exists within an element, there SHALL be only one <Status> element within the enclosing  
2699 element.

2700 The <Status> element can contain one of four **String** type values:

- 2701 1. The **Active** value indicates that the element that makes up the document-root is currently active  
2702 in the SKMS and conforming **SKCL** implementations may use it within applications.
- 2703 2. The **Default** value indicates that the element that makes up the document root is the default  
2704 element in the SKMS, is also active, and conforming **SKCL** implementations may use it within  
2705 applications.
- 2706 3. The **Inactive** value indicates that the element that makes up the document root is not active in  
2707 the SKMS, and conforming **SKCL** implementations may NOT use it within applications.
- 2708 4. The **Other** value indicates that the element that makes up the document root has a meaning that  
2709 is application-specific. However, conforming **SKCL** implementations may NOT use it within  
2710 applications.

2711 Some examples of the <Status> element are shown below; other parts of their enclosing elements are  
2712 not shown for brevity:

2713 **Example 1 – An example with an Active status within a <KeyUsePolicy> element:**

```
2714     <ekmi:KeyUsePolicy>
2715         ...
2716         <ekmi:Status>Active</ekmi:Status>
2717         ...
2718     </ekmi:KeyUsePolicy>
```

2719 **Example 2 – An example with an Inactive status within a <KeyUsePolicy> element:**

```
2720     <ekmi:KeyUsePolicy>
2721         ...
2722         <ekmi:Status>Inactive</ekmi:Status>
2723         ...
2724     </ekmi:KeyUsePolicy>
```

2725 **Example 3 – An example with a Default status within a <KeyUsePolicy> element:**

```
2726     <ekmi:KeyUsePolicy>
2727         ...
2728         <ekmi:Status>Default</ekmi:Status>
2729         ...
2730     </ekmi:KeyUsePolicy>
```

## 2731 4.15 Element <Permissions>

2732 The <Permissions> element, of the type **PermissionsType** is at the heart of the <KeyUsePolicy>  
2733 element. It provides guidance to conforming **SKCL** implementations on who may use the symmetric key,  
2734 when they may use it, for what purposes, for how long and in which locations. For applications that  
2735 conform to the Multi-Level Security (MLS) model, there is a provision for specifying which levels are  
2736 permitted use of the key. There is also an element that allows for extending the <Permissions> element  
2737 to accommodate rules that have not been envisioned in the current specification.

2738 There SHALL be only one <Permissions> element within a <KeyUsePolicy> element.

### 2739 Schema Definition:

```
2740     <xsd:complexType name="PermissionsType">
2741         <xsd:sequence>
2742             <xsd:element
2743                 name="PermittedApplications"
2744                 type="ekmi:PermittedApplicationsType"
2745                 minOccurs="1"
2746                 nillable="true"/>
2747             <xsd:element
2748                 name="PermittedDates"
2749                 type="ekmi:PermittedDatesType"
2750                 minOccurs="1"
2751                 nillable="true"/>
2752             <xsd:element
2753                 name="PermittedDays"
2754                 type="ekmi:PermittedDaysType"
2755                 minOccurs="1"
2756                 nillable="true"/>
2757             <xsd:element name="PermittedDuration"
2758                 type="ekmi:PermittedDurationType"
2759                 minOccurs="1"
2760                 nillable="true"/>
2761             <xsd:element
2762                 name="PermittedLevels"
2763                 type="ekmi:PermittedLevelsType"
```

```

2764         minOccurs="1"
2765         nillable="true"/>
2766     <xsd:element
2767         name="PermittedLocations"
2768         type="ekmi:PermittedLocationsType"
2769         minOccurs="1"
2770         nillable="true"/>
2771     <xsd:element
2772         name="PermittedNumberOfTransactions"
2773         type="ekmi:PermittedNumberOfTransactionsType"
2774         minOccurs="1"
2775         nillable="true"/>
2776     <xsd:element
2777         name="PermittedTimes"
2778         type="ekmi:PermittedTimesType"
2779         minOccurs="1"
2780         nillable="true"/>
2781     <xsd:element
2782         name="PermittedUses"
2783         type="ekmi:PermittedUsesType"
2784         minOccurs="1"
2785         nillable="true"/>
2786     <xsd:element
2787         name="Other"
2788         type="xsd:anyType"
2789         minOccurs="0"/>
2790 </xsd:sequence>
2791 </xsd:complexType>

```

2792 The <Permissions> element consists of the following sub-elements:

- 2793 1. A required <PermittedApplications> element which identifies applications that are permitted  
2794 use of the symmetric key in question. While the <PermittedApplications> element is  
2795 required, it may be empty (NULL). The absence of sub-elements in the  
2796 <PermittedApplications> element implies that all applications are permitted to use the key.  
2797 Identifying a specific application restricts the use of the key to only the identified applications.

2798  
2799 The <PermittedApplications> element is specified in Section 4.16.

- 2800 2. A required <PermittedDates> element which identifies the date ranges during which applications  
2801 are permitted use of the symmetric key in question. While the <PermittedDates> element is  
2802 required, it may be empty (NULL). The absence of sub-elements in the <PermittedDates>  
2803 element implies that applications are permitted to use the key on any date. Identifying specific  
2804 date ranges restricts the use of the key to only the duration between the identified dates.

2805  
2806 The <PermittedDates> element is specified in Section 4.17.

- 2807 3. A required <PermittedDays> element which identifies the days of week during which applications  
2808 are permitted use of the symmetric key in question. While the <PermittedDays> element is  
2809 required, it may be empty (NULL). The absence of sub-elements in the <PermittedDays>  
2810 element implies that applications are permitted to use the key on any day of the week. Identifying  
2811 specific days restricts the use of the key to only the identified days.

2812  
2813 The <PermittedDays> element is specified in Section 4.18.

- 2814 4. A required <PermittedDuration> element which identifies the duration (in seconds) in which  
2815 applications are permitted use of the symmetric key in question *once the SKCL starts using the*  
2816 *symmetric key*. While the <PermittedDuration> element is required, it may be empty (NULL).  
2817 The absence of any content – the duration time - in the <PermittedDuration> element implies  
2818 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 element.

The <PermittedNumberOfTransactions> element is specified in Section 4.22.

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

The <PermittedTimes> element is specified in Section 4.23.

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

The <PermittedUses> element is specified in Section 4.24.

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

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

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

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

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

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

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

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

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

```
<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:Permissions>
```

```

2924         </ekmi:DigestValue>
2925     </ekmi:PermittedApplication>
2926 </ekmi:PermittedApplications>
2927 <ekmi:PermittedDates ekmi:any="false">
2928     <ekmi:PermittedDate>
2929         <ekmi:StartDate>2008-01-01</ekmi:StartDate>
2930         <ekmi:EndDate>2008-12-31</ekmi:EndDate>
2931     </ekmi:PermittedDate>
2932 </ekmi:PermittedDates>
2933 <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2934 <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2935 <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2936 <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2937 <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
2938 <ekmi:PermittedTimes ekmi:any="false">
2939     <ekmi:PermittedTime>
2940         <ekmi:StartTime>07:00:00</ekmi:StartTime>
2941         <ekmi:EndTime>19:00:00</ekmi:EndTime>
2942     </ekmi:PermittedTime>
2943 </ekmi:PermittedTimes>
2944 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2945 </ekmi:Permissions>

```

2946 **Example 2 – A <Permissions> element that permits two specific applications the use of the**  
2947 **symmetric key in question, between January 01, 2009 and January 31, 2009.**

```

2948 <ekmi:Permissions>
2949     <ekmi:PermittedApplications ekmi:any="false">
2950         <ekmi:PermittedApplication>
2951             <ekmi:ApplicationID>10514-24</ekmi:ApplicationID>
2952             <ekmi:ApplicationName>
2953                 Employee Tax Reporting Application
2954             </ekmi:ApplicationName>
2955             <ekmi:Version>3.3</ekmi:Version>
2956             <ekmi:DigestAlgorithm>
2957                 http://www.w3.org/2000/09/xmldsig#sha1
2958             </ekmi:DigestAlgorithm>
2959             <ekmi:DigestValue>
2960                 af96d65a7a2415239c8eb8be1347f704322957a4
2961             </ekmi:DigestValue>
2962         </ekmi:PermittedApplication>
2963         <ekmi:PermittedApplication>
2964             <ekmi:ApplicationID>10514-25</ekmi:ApplicationID>
2965             <ekmi:ApplicationName>
2966                 IRS Tax Reporting Application
2967             </ekmi:ApplicationName>
2968             <ekmi:Version>2.1</ekmi:Version>
2969             <ekmi:DigestAlgorithm>
2970                 http://www.w3.org/2000/09/xmldsig#sha1
2971             </ekmi:DigestAlgorithm>
2972             <ekmi:DigestValue>
2973                 a4f5925185ffe12c1a91ea3de90fc086b34b34b2
2974             </ekmi:DigestValue>
2975         </ekmi:PermittedApplication>
2976     </ekmi:PermittedApplications>
2977     <ekmi:PermittedDates ekmi:any="false">
2978         <ekmi:PermittedDate>
2979             <ekmi:StartDate>2009-01-01</ekmi:StartDate>
2980             <ekmi:EndDate>2009-12-31</ekmi:EndDate>
2981         </ekmi:PermittedDate>
2982     </ekmi:PermittedDates>

```

```

2983     <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2984     <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2985     <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2986     <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
2987     <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
2988     <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
2989     <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
2990 </ekmi:Permissions>

```

2991 **Example 3 – A <Permissions> element that permits all applications the use of the symmetric key**  
2992 **in question, for 100 transactions for encrypting credit card numbers.**

```

2993 <ekmi:Permissions>
2994   <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
2995   <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
2996   <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
2997   <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
2998   <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
2999   <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
3000   <ekmi:PermittedNumberOfTransactions ekmi:any="false">
3001     100
3002   </ekmi:PermittedNumberOfTransactions>
3003   <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
3004   <ekmi:PermittedUses ekmi:any="false">
3005     <ekmi:PermittedUse>CCN</ekmi:PermittedUse>
3006   </ekmi:PermittedUses>
3007 </ekmi:Permissions>

```

3008 **Example 4 – A <Permissions> element that permits all applications the use of the symmetric key**  
3009 **in question, for 600 seconds once the SKCL starts using the key.**

```

3010 <ekmi:Permissions>
3011   <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
3012   <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
3013   <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
3014   <ekmi:PermittedDuration ekmi:any="false">600</ekmi:PermittedDuration>
3015   <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
3016   <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
3017   <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
3018   <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
3019   <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
3020 </ekmi:Permissions>

```

3021 **Example 5 – A <Permissions> element that permits a specific application the use of the**  
3022 **symmetric key in question, at specific geographic locations only on weekdays between the hours**  
3023 **of 0800 and 1700, and only when the application is operating at the Secret security level within an**  
3024 **MLS system.**

```

3025 <ekmi:Permissions>
3026   <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
3027   <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
3028   <ekmi:PermittedDays ekmi:any="false">
3029     <ekmi:PermittedDay>Weekday</ekmi:PermittedDay>
3030   </ekmi:PermittedDays>
3031   <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
3032   <ekmi:PermittedLevels ekmi:any="false">
3033     <ekmi:PermittedLevel>Secret</ekmi:PermittedLevel>
3034   </ekmi:PermittedLevels>
3035   <ekmi:PermittedLocations ekmi:any="false">
3036     <ekmi:PermittedLocation>
3037       <ekmi:LocationName>Facility A51</ekmi:LocationName>

```

```

3038         <ekmi:Latitude>37.385562</ekmi:Latitude>
3039         <ekmi:Longitude>-121.993387</ekmi:Latitude>
3040     </ekmi:PermittedLocation>
3041     <ekmi:PermittedLocation>
3042         <ekmi:LocationName>Facility DC-VA01</ekmi:LocationName>
3043         <ekmi:Latitude>88.485362</ekmi:Latitude>
3044         <ekmi:Longitude>-21.453648</ekmi:Latitude>
3045     </ekmi:PermittedLocation>
3046 </ekmi:PermittedLocations>
3047 <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
3048 <ekmi:PermittedTimes ekmi:any="false">
3049     <ekmi:PermittedTime>
3050         <ekmi:StartTime>08:00:00</ekmi:StartTime>
3051         <ekmi:EndTime>17:00:00</ekmi:EndTime>
3052     </ekmi:PermittedTime>
3053 </ekmi:PermittedTimes>
3054 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
3055 </ekmi:Permissions>

```

## 3056 4.16 Element <PermittedApplications> and <PermittedApplication>

3057 The element <PermittedApplications>, of type **PermittedApplicationsType** and its only child-  
3058 element <PermittedApplication> of type **ApplicationsType** are used to define the list of applications  
3059 that are permitted to use a symmetric key within a specific <Symkey> element.

### 3060 Schema Definition:

```

3061 <xsd:complexType name="PermittedApplicationsType">
3062     <xsd:sequence>
3063         <xsd:element
3064             name="PermittedApplication"
3065             type="ekmi:ApplicationsType"
3066             minOccurs="0"
3067             maxOccurs="unbounded"/>
3068     </xsd:sequence>
3069     <xsd:attribute ref="ekmi:any" use="required"/>
3070 </xsd:complexType>

```

### 3071 Schema Definition:

```

3072 <xsd:complexType name="ApplicationsType">
3073     <xsd:sequence>
3074         <xsd:element name="ApplicationID" type="ekmi:TwoPartIDType"/>
3075         <xsd:element name="ApplicationName">
3076             <xsd:simpleType>
3077                 <xsd:restriction base="xsd:string">
3078                     <xsd:maxLength value="256"/>
3079                     <xsd:whiteSpace value="preserve"/>
3080                 </xsd:restriction>
3081             </xsd:simpleType>
3082         </xsd:element>
3083         <xsd:element name="Version" minOccurs="0">
3084             <xsd:simpleType>
3085                 <xsd:restriction base="xsd:string">
3086                     <xsd:maxLength value="32"/>
3087                     <xsd:whiteSpace value="preserve"/>
3088                 </xsd:restriction>
3089             </xsd:simpleType>
3090         </xsd:element>
3091         <xsd:group ref="ekmi:MessageDigestGroup" minOccurs="0"/>
3092     </xsd:sequence>

```

```

3093         <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
3094     </xsd:sequence>
3095 </xsd:complexType>

```

3096 **Schema Definition:**

```

3097
3098     <xsd:group name="MessageDigestGroup">
3099         <xsd:sequence>
3100             <xsd:element name="DigestAlgorithm">
3101                 <xsd:simpleType>
3102                     <xsd:restriction base="xsd:anyURI">
3103                         <xsd:enumeration value="http://www.w3.org/2000/09/xmldsig#sha1"/>
3104                         <xsd:enumeration value="http://www.w3.org/2001/04/xmenc#sha256"/>
3105                         <xsd:enumeration value="http://www.w3.org/2001/04/xmenc#sha512"/>
3106                     </xsd:restriction>
3107                 </xsd:simpleType>
3108             </xsd:element>
3109             <xsd:element name="DigestValue">
3110                 <xsd:simpleType>
3111                     <xsd:restriction base="xsd:base64Binary">
3112                         <xsd:maxLength value="1024"/>
3113                     </xsd:restriction>
3114                 </xsd:simpleType>
3115             </xsd:element>
3116         </xsd:sequence>
3117     </xsd:group>

```

3118 **Schema Definition:**

```

3119     <xsd:attribute name="any">
3120         <xsd:simpleType>
3121             <xsd:restriction base="xsd:string">
3122                 <xsd:enumeration value="false"/>
3123                 <xsd:enumeration value="true"/>
3124             </xsd:restriction>
3125         </xsd:simpleType>
3126     </xsd:attribute>

```

3127 There SHALL be only one <PermittedApplications> element within the <Permissions> element.  
3128 However, there MAY be an unbounded (unlimited) number of <PermittedApplication> elements within  
3129 a <PermittedApplications> element.

3130 The <PermittedApplications> element SHALL have one attribute named “any”, that will have a  
3131 “false” or “true” value, based on the following:

- 3132 • When the <PermittedApplications> element is null (i.e. it does not have a single  
3133 <PermittedApplication> sub-element in it), the value of the “any” attribute SHALL be set to  
3134 “true” AND the XML Schema Instance (XSI) “nil” attribute SHALL be set to “true”.
- 3135 • When the <PermittedApplications> element is not-null (i.e. it has at least one  
3136 <PermittedApplication> sub-element in it), the value of the “any” attribute SHALL be set to  
3137 “false” AND the XML Schema Instance (XSI) “nil” attribute SHALL NOT be present.

3138 A null <PermittedApplications> element specifies that ALL applications are permitted use of the  
3139 symmetric key, subject to complying with all other permission clauses in the <KeyUsePolicy> element.

3140 The <PermittedApplication> sub-element of type **ApplicationsType**, provides details of the  
3141 application which is permitted use of the symmetric key in question. The <PermittedApplication>  
3142 element consists of the following sub-elements:

1. The <ApplicationID> element identifies the unique identifier assigned to this application within the SKMS. It is a **TwoPartIDType** as specified in Section 4.10. There SHALL be only one <ApplicationID> element within a <PermittedApplication> element.
  2. The <ApplicationName> element identifies the name assigned to this application within the SKMS. It is an XSD **String** with a maximum length of 256 characters. There SHALL be only one <ApplicationName> element within a <PermittedApplication> element.
  3. An optional <Version> element identifying the version number of this application within the **SKMS**. It is an XSD **String** with a maximum length of 32 characters. There MAY be only one <Version> element within a <PermittedApplication> element.
  4. The <MessageDigestGroup> group which identifies the message digest value of the application's binary image, along with the message digest algorithm used to calculate the digest value. The <MessageDigestGroup> consists of the following elements:
    - a) The <DigestAlgorithm> element of the XSD type **anyURI**, which supports one of the following three digest algorithms:
      - i. <http://www.w3.org/2000/09/xmldsig#sha1>
      - ii. <http://www.w3.org/2001/04/xmlenc#sha256>
      - iii. <http://www.w3.org/2001/04/xmlenc#sha512>
    - b) The <DigestValue> element of the XSD type **base64Binary**.
- There SHALL be only one <MessageDigestGroup> group within a <PermittedApplication> element.
5. An optional <Other> element that provides implementers the ability to carry other information about the application that may be relevant to their **SKMS**. Implementers are cautioned that the use of the <Other> element may not be supported by other **SKCL** implementations, and may break interoperability between two **SKMS** implementations. Should there be a strong need for additional features in the <PermittedApplication> element, implementers are encouraged to contact the OASIS EKMI TC with their requirements.

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

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

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

```
<ekmi:PermittedApplications ekmi:any="false">
  <ekmi:PermittedApplication>
    <ekmi:ApplicationID>10514-24</ekmi:ApplicationID>
    <ekmi:ApplicationName>Employee Tax
Reporting</ekmi:ApplicationName>
    <ekmi:Version>3.3</ekmi:Version>
    <ekmi:DigestAlgorithm>
http://www.w3.org/2000/09/xmldsig#sha1
    </ekmi:DigestAlgorithm>
    <ekmi:DigestValue>G4bsdfKkt4cziEqFFu0oBTM81efU=</ekmi:DigestValue>
  </ekmi:PermittedApplication>
```



```

3189     <ekmi:PermittedApplication>
3190         <ekmi:ApplicationID>10514-25</ekmi:ApplicationID>
3191         <ekmi:ApplicationName>IRS Tax Reporting
3192 Application</ekmi:ApplicationName>
3193         <ekmi:Version>2.1</ekmi:Version>
3194         <ekmi:DigestAlgorithm>
3195             http://www.w3.org/2001/04/xmlenc#sha256
3196         </ekmi:DigestAlgorithm>
3197         <ekmi:DigestValue>
3198
3199             ab7b85c9410d48c54fc7939c391be4028e7305085191c56e7b3740f2cbdbbc79
3200         </ekmi:DigestValue>
3201     </ekmi:PermittedApplication>
3202 </ekmi:PermittedApplications>

```

3203 **Example 2 – An example of a <PermittedApplications> element with one child**  
3204 **<PermittedApplication> element that applies to all versions of the application; the message**  
3205 **digest value and algorithm are not used in this example:**

```

3206     <ekmi:PermittedApplications ekmi:any="false">
3207         <ekmi:PermittedApplication>
3208             <ekmi:ApplicationID>10514-14</ekmi:ApplicationID>
3209             <ekmi:ApplicationName>E-Commerce Payment</ekmi:ApplicationName>
3210         </ekmi:PermittedApplication>
3211     </ekmi:PermittedApplications>

```

3212 **Example 3 – An example of a null <PermittedApplications> element specifying that ALL**  
3213 **applications are permitted the use of the symmetric key:**

```

3214     <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>

```

## 3215 4.17 Element <PermittedDates> and <PermittedDate>

3216 The element <PermittedDates>, of type *PermittedDatesType* and its only child-element  
3217 <PermittedDate>, which is an anonymous XSD *ComplexType*, are used to define ranges of dates  
3218 between which applications are permitted to use the symmetric key within a specific <Symkey> element.

### 3219 Schema Definition:

```

3220     <xsd:complexType name="PermittedDatesType">
3221         <xsd:sequence>
3222             <xsd:element name="PermittedDate" minOccurs="0" maxOccurs="unbounded">
3223                 <xsd:complexType>
3224                     <xsd:sequence>
3225                         <xsd:element name="StartDate">
3226                             <xsd:simpleType>
3227                                 <xsd:restriction base="xsd:date">
3228                                     <xsd:pattern value="\{Nd\}{4}-\{Nd\}{2}-\{Nd\}{2}" />
3229                                 </xsd:restriction>
3230                             </xsd:simpleType>
3231                         </xsd:element>
3232                         <xsd:element name="EndDate">
3233                             <xsd:simpleType>
3234                                 <xsd:restriction base="xsd:date">
3235                                     <xsd:pattern value="\{Nd\}{4}-\{Nd\}{2}-\{Nd\}{2}" />
3236                                 </xsd:restriction>
3237                             </xsd:simpleType>
3238                         </xsd:element>

```



```

3239         </xsd:sequence>
3240     </xsd:complexType>
3241 </xsd:element>
3242 </xsd:sequence>
3243 <xsd:attribute ref="ekmi:any" use="required"/>
3244 </xsd:complexType>

```

3245 There SHALL be only one <PermittedDates> element within the <Permissions> element. However,  
3246 there MAY be an unbounded number of <PermittedDate> elements within a <PermittedDates>  
3247 element.

3248 The <PermittedDates> element SHALL have one attribute named “any”, that will have a “false” or “true”  
3249 value, based on the following:

- 3250 • When the <PermittedDates> element is null (i.e. it does not have a single <PermittedDate>  
3251 sub-element in it), the value of the “any” attribute SHALL be set to “true” AND the XML Schema  
3252 Instance (XSI) “nil” attribute SHALL be set to “true”.
- 3253 • When the <PermittedDates> element is not-null (i.e. it has at least one <PermittedDate> sub-  
3254 element in it), the value of the “any” attribute SHALL be set to “false” AND the XML Schema  
3255 Instance (XSI) “nil” attribute SHALL NOT be present.

3256 A null <PermittedDates> element specifies that applications are permitted use of the symmetric key on  
3257 any calendar date of the year, subject to complying with all other permission clauses in the  
3258 <Permissions> element.

3259 The <PermittedDate> sub-element identifies an individual set of dates between which application are  
3260 permitted use of the symmetric key in question. The <PermittedDate> element consists of the following  
3261 sub-elements:

- 3262 1. The <StartDate> element identifies the date from which applications MAY start using the  
3263 symmetric key in question. It is an XSD **Date** type that MUST be specified in a specific pattern  
3264 (see examples) where the first four digits specify the year, the second two digits specify the  
3265 calendar month in the year, and the last two digits specify the calendar date of the month.

3266 There SHALL be only one <StartDate> element within a <PermittedDate> element.

3267 Conforming **SKCL** implementations SHALL NOT start using the symmetric before the onset of  
3268 the <StartDate> on the client machine. Unless further constrained by the <PermittedTimes>  
3269 element, the onset of the <StartDate> is specified to be the first second of the day – 00:00:01  
3270 Hours – on the client machine.

- 3273 2. The <EndDate> element identifies the date until which applications may use the symmetric key in  
3274 question. It is an XSD **Date** type that MUST be specified in a specific pattern (see examples)  
3275 where the first four digits specify the year, the second two digits specify the calendar month in the  
3276 year, and the last two digits specify the calendar date of the month.

3277 There SHALL be only one <EndDate> element within a <PermittedDate> element.

3278 Conforming **SKCL** implementations SHALL NOT use the symmetric after the end of the  
3279 <EndDate> on the client machine. Unless further constrained by the <PermittedTimes>  
3280 element, the end of the <EndDate> is specified to be the last second of the day – 23:59:59 Hours  
3281 – on the client machine.

3284 Examples of the <PermittedDates> element are shown below; other required parts of their enclosing  
3285 elements are not shown for brevity:

3286 **Example 1 – An example of a <PermittedDates> element with a single<PermittedDate> element.**  
3287 **The <StartDate> specifies January 01, 2009 while the <EndDate> specifies January 31, 2009:**

```

3288     <ekmi:PermittedDates ekmi:any="false">
3289         <ekmi:PermittedDate>
3290             <ekmi:StartDate>2009-01-01</ekmi:StartDate>
3291             <ekmi:EndDate>2009-01-31</ekmi:EndDate>
3292         </ekmi:PermittedDate>
3293     </ekmi:PermittedDates>

```

3294 **Example 2 – An example of a <PermittedDates> element with two <PermittedDate> elements. For**  
3295 **the first <PermittedDate> element , the <StartDate> element specifies July 01, 2008 while the**  
3296 **<EndDate> element specifies July 03, 2008. For the second <PermittedDate> element, the**  
3297 **<StartDate> element specifies July 07, 2008 while the <EndDate> element specifies July 12, 2008.**  
3298 **This policy would restrict a symmetric key with this <PermittedDates> element so it cannot be**  
3299 **used on the July 4<sup>th</sup> weekend of 2008:**

```

3300     <ekmi:PermittedDates ekmi:any="false">
3301         <ekmi:PermittedDate>
3302             <ekmi:StartDate>2008-07-01</ekmi:StartDate>
3303             <ekmi:EndDate>2008-07-03</ekmi:EndDate>
3304         </ekmi:PermittedDate>
3305         <ekmi:PermittedDate>
3306             <ekmi:StartDate>2008-07-07</ekmi:StartDate>
3307             <ekmi:EndDate>2009-07-12</ekmi:EndDate>
3308         </ekmi:PermittedDate>
3309     </ekmi:PermittedDates>

```

3310 **Example 3 – An example of a null <PermittedDates> element, specifying that applications are**  
3311 **permitted use of the symmetric key on any date:**

```

3312     <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>

```

## 3313 4.18 Element <PermittedDays> and <PermittedDay>

3314 The element <PermittedDays> , of the type *PermittedDaysType* and its only child-element  
3315 <PermittedDay> of the *PermittedDayType*, are used to define days of the week on which applications  
3316 are permitted to use a symmetric key within a specific <Symkey> element.

### 3317 Schema Definition:

```

3318     <xsd:complexType name="PermittedDaysType">
3319         <xsd:sequence>
3320             <xsd:element
3321                 name="PermittedDay"
3322                 type="ekmi:PermittedDayType"
3323                 minOccurs="0"
3324                 maxOccurs="unbounded">
3325             </xsd:element>
3326         </xsd:sequence>
3327         <xsd:attribute ref="ekmi:any" use="required"/>
3328     </xsd:complexType>

```

### 3329 Schema Definition:

```

3330     <xsd:simpleType name="PermittedDayType">
3331         <xsd:restriction base="xsd:string">
3332             <xsd:enumeration value="Sunday"/>
3333             <xsd:enumeration value="Monday"/>
3334             <xsd:enumeration value="Tuesday"/>
3335             <xsd:enumeration value="Wednesday"/>
3336             <xsd:enumeration value="Thursday"/>

```

```

3337     <xsd:enumeration value="Friday"/>
3338     <xsd:enumeration value="Saturday"/>
3339     <xsd:enumeration value="Weekday"/>
3340     <xsd:enumeration value="Weekend"/>
3341     </xsd:restriction>
3342 </xsd:simpleType>

```

3343 There SHALL be only one <PermittedDays> element within the <Permissions> element. However,  
 3344 there MAY be an unbounded (unlimited) number of <PermittedDay> elements within a  
 3345 <PermittedDays> element.

3346 The <PermittedDays> element SHALL have one attribute named “any”, that will have a “false” or “true”  
 3347 value, based on the following:

- 3348 • When the <PermittedDays> element is null (i.e. it does not have a single <PermittedDay> sub-  
 3349 element in it), the value of the “any” attribute SHALL be set to “true” AND the XML Schema  
 3350 Instance (XSI) “nil” attribute SHALL be set to “true”.
- 3351 • When the <PermittedDays> element is not-null (i.e. it has at least one <PermittedDay> sub-  
 3352 element in it), the value of the “any” attribute SHALL be set to “false” AND the XML Schema  
 3353 Instance (XSI) “nil” attribute SHALL NOT be present.

3354 A null <PermittedDays> element specifies that applications are permitted use of the symmetric key on all  
 3355 days of the week, subject to complying with all other permission clauses in the <Permissions> element.

3356 The <PermittedDay> element, of the XSD **String** type, identifies individual days of the week from an  
 3357 enumerated list on which application are permitted to use the symmetric key in question.

3358 Examples of the <PermittedDays> element are shown below; other parts of their enclosing elements are  
 3359 not shown for brevity:

3360 **Example 1 – An example of a <PermittedDays> element with a single<PermittedDay> child**  
 3361 **element, specifying that the symmetric key may be used only on weekdays:**

```

3362     <ekmi:PermittedDays ekmi:any="false">
3363       <ekmi:PermittedDay>Weekday </ekmi:PermittedDay>
3364     </ekmi:PermittedDays>

```

3365 **Example 2 – An example of a <PermittedDays> element with three <PermittedDay> child**  
 3366 **elements, specifying that the symmetric key may be used only on Mondays, Wednesdays and**  
 3367 **Fridays:**

```

3368     <ekmi:PermittedDays ekmi:any="false">
3369       <ekmi:PermittedDay>Monday</ekmi:PermittedDay>
3370       <ekmi:PermittedDay>Wednesday</ekmi:PermittedDay>
3371       <ekmi:PermittedDay>Friday</ekmi:PermittedDay>
3372     </ekmi:PermittedDays>

```

3373 **Example 3 – An example of a null <PermittedDays> element, specifying that the symmetric key**  
 3374 **may be used on any day of the week:**

```

3375     <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>

```

## 3376 4.19 Element <PermittedDuration>

3377 The element <PermittedDuration>, of the type **PermittedDurationType** is used to define the number  
 3378 of seconds, applications are permitted to use a symmetric key, once the **SKCL** has started using the  
 3379 symmetric key in question.

### 3380 Schema Definition:

```

3381     <xsd:complexType name="PermittedDurationType">
3382         <xsd:simpleContent>
3383             <xsd:extension base="ekmi:DurationType">
3384                 <xsd:attribute ref="ekmi:any" use="required"/>
3385             </xsd:extension>
3386         </xsd:simpleContent>
3387     </xsd:complexType>

```

#### 3388 Schema Definition:

```

3389     <xsd:simpleType name="DurationType">
3390         <xsd:restriction base="xsd:positiveInteger">
3391             <xsd:minInclusive value="1"/>
3392             <xsd:maxInclusive value="18446744073709551615"/>
3393         </xsd:restriction>
3394     </xsd:simpleType>

```

3395 There SHALL be only one <PermittedDuration> element within the <Permissions> element.

3396 The <PermittedDuration> element SHALL have one attribute, named “any” that will have a “false” or  
3397 “true” value, based on the following:

- 3398 • When the <PermittedDuration> element is null (i.e. it does not have any content in it), the value  
3399 of the “any” attribute SHALL be set to “true” AND the XML Schema Instance (XSI) “nil” attribute  
3400 SHALL be set to “true”.
- 3401 • When the <PermittedDuration> element is not-null (i.e. it has content in it), the value of the  
3402 “any” attribute SHALL be set to “false” AND the XML Schema Instance (XSI) “nil” attribute SHALL  
3403 NOT be present.

3404 A null <PermittedDuration> element specifies that applications are permitted use of the symmetric key  
3405 indefinitely, subject to complying with all other permission clauses in the <Permissions> element.

3406 The <PermittedDuration> element, of the XSD *positiveInteger* type, identifies the number of seconds  
3407 for which the symmetric key in question may be used, ONCE the key has been used by conforming **SKCL**  
3408 implementations for the first time. The values for <PermittedDuration> may range between 1 and  
3409 18446744073709551615.

3410 As long as the symmetric has not been used by an **SKCL** on a client device (it might be cached for many  
3411 days/weeks/months depending on the <KeyCachePolicy> in effect within the SKMS for that device) the  
3412 effective lifetime of the symmetric key may be well past the number of seconds specified in  
3413 <PermittedDuration> when calculated from the time of the key's generation time on the **SKS** server. It  
3414 is the responsibility of the **SKCL** implementation, when presented with a <PermittedDuration> element  
3415 in a <KeyUsePolicy> of a symmetric key, to keep track of the date/time when the symmetric key in  
3416 question was first used on the client device, and how long the key will last after that.

3417 Examples of the <PermittedDuration> element are shown below; other parts of their enclosing  
3418 elements are not shown for brevity:

3419 **Example 1 – An example of a <PermittedDuration> element specifying that the symmetric key**  
3420 **may be used only for a single 24-hour period from the moment it is first used by an SKCL:**

```
3421     <ekmi:PermittedDuration ekmi:any="false">86400</ekmi:PermittedDuration>
```

3422 **Example 2 – An example of a <PermittedDuration> element specifying that the symmetric key**  
3423 **may be used only for week from the moment it is first used by an SKCL:**

```
3424     <ekmi:PermittedDuration ekmi:any="false">604800</ekmi:PermittedDuration>
```

3425 **Example 3 – An example of a <PermittedDuration> element specifying that the symmetric key**  
3426 **may be used only 5 minutes from the moment it is first used by an SKCL:**

3427       <ekmi:PermittedDuration ekmi:any="false">300</ekmi:PermittedDuration>

3428 **Example 4 – An example of a null <PermittedDuration> element specifying that the symmetric**  
3429 **key may be used indefinitely by an SKCL:**

3430       <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>

## 3431 **4.20 Element <PermittedLevels> and <PermittedLevel>**

3432 The element <PermittedLevels>, of the type **LevelClassificationType**, is used to define the security  
3433 level at which applications are permitted use of a symmetric key. This element is useful only to  
3434 applications and systems that conform to the multi-level security (MLS) system as defined in the Bell-  
3435 LaPadula model.

### 3436 **Schema Definition:**

```
3437 <xsd:complexType name="PermittedLevelsType">
3438   <xsd:sequence>
3439     <xsd:element
3440       name="PermittedLevel"
3441       type="ekmi:LevelClassificationType"
3442       minOccurs="0"
3443       maxOccurs="unbounded">
3444     </xsd:element>
3445     <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
3446   </xsd:sequence>
3447   <xsd:attribute ref="ekmi:any" use="required"/>
3448 </xsd:complexType>
```

### 3449 **Schema Definition:**

```
3450 <xsd:simpleType name="LevelClassificationType">
3451   <xsd:restriction base="xsd:string">
3452     <xsd:enumeration value="Unclassified"/>
3453     <xsd:enumeration value="Confidential"/>
3454     <xsd:enumeration value="Secret"/>
3455     <xsd:enumeration value="Top-Secret"/>
3456   </xsd:restriction>
3457 </xsd:simpleType>
```

3458 There SHALL be only one <PermittedLevels> element within the <Permissions> element. However,  
3459 there MAY be an unbounded (unlimited) number of <PermittedLevel> elements within the  
3460 <PermittedLevels> element. (Practically, it makes no sense to have more than the known levels;  
3461 however, this specification leaves itself open to the possibility that other levels may be defined).

3462 The <PermittedLevels> element SHALL have one attribute named "any", that will have a "false" or  
3463 "true" value, based on the following:

- 3464       • When the <PermittedLevels> element is null (i.e. it does not have a single <PermittedLevel>  
3465       sub-element in it), the value of the "any" attribute SHALL be set to "true" AND the XML Schema  
3466       Instance (XSI) "nil" attribute SHALL be set to "true".
- 3467       • When the <PermittedLevels> element is not-null (i.e. it has at least one <PermittedLevel>  
3468       sub-element in it), the value of the "any" attribute SHALL be set to "false" AND the XML Schema  
3469       Instance (XSI) "nil" attribute SHALL NOT be present.

3470 A null <PermittedLevels> element specifies that applications at ANY level are permitted use of the  
3471 symmetric key, subject to complying with all other permission clauses in the <Permissions> element.

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

1. Unclassified
2. Confidential
3. Secret
4. Top-Secret

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

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

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

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

```
<ekmi:PermittedLevels ekmi:any="false">
  <ekmi:PermittedLevel>Secret</ekmi:PermittedLevel>
  <ekmi:PermittedLevel>Top-Secret</ekmi:PermittedLevel>
</ekmi:PermittedLevels>
```

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

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

## 4.21 Element <PermittedLocations> and <PermittedLocation>

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

### Schema Definition:

```
<xsd:complexType name="PermittedLocationsType">
  <xsd:sequence>
    <xsd:element name="PermittedLocation" minOccurs="1" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="LocationName">
            <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                <xsd:maxLength value="256"/>
                <xsd:whiteSpace value="preserve"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:group
            ref="ekmi:LocationCoordinateGroup"
            minOccurs="0"/>
```



```

3517         maxOccurs="unbounded"/>
3518         <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
3519     </xsd:sequence>
3520 </xsd:complexType>
3521 </xsd:element>
3522 </xsd:sequence>
3523 <xsd:attribute ref="ekmi:any" use="required"/>
3524 </xsd:complexType>

```

#### 3525 Schema Definition:

```

3526 <xsd:group name="LocationCoordinateGroup">
3527     <xsd:sequence>
3528         <xsd:element name="Latitude">
3529             <xsd:simpleType>
3530                 <xsd:restriction base="xsd:decimal">
3531                     <xsd:totalDigits value="10"/>
3532                     <xsd:fractionDigits value="7"/>
3533                 </xsd:restriction>
3534             </xsd:simpleType>
3535         </xsd:element>
3536         <xsd:element name="Longitude">
3537             <xsd:simpleType>
3538                 <xsd:restriction base="xsd:decimal">
3539                     <xsd:totalDigits value="10"/>
3540                     <xsd:fractionDigits value="7"/>
3541                 </xsd:restriction>
3542             </xsd:simpleType>
3543         </xsd:element>
3544     </xsd:sequence>
3545 </xsd:group>

```

3546 There SHALL be only one <PermittedLocations> element within the <Permissions> element.  
3547 However, there MAY be an unbounded (unlimited) number of <PermittedLocation> sub-elements within  
3548 the <PermittedLocations> element.

3549 The <PermittedLocations> element SHALL have one attribute named "any", that will have a "false" or  
3550 "true" value, based on the following:

- 3551 • When the <PermittedLocations> element is null (i.e. it does not have a single  
3552 <PermittedLocation> sub-element in it), the value of the "any" attribute SHALL be set to "true"  
3553 AND the XML Schema Instance (XSI) "nil" attribute SHALL be set to "true".
- 3554 • When the <PermittedLocations> element is not-null (i.e. it has at least one  
3555 <PermittedLocation> sub-element in it), the value of the "any" attribute SHALL be set to  
3556 "false" AND the XML Schema Instance (XSI) "nil" attribute SHALL NOT be present.

3557 A null <PermittedLocations> element specifies that applications are permitted use of the symmetric key  
3558 at ANY physical location, subject to complying with all other permission clauses in the <Permissions>  
3559 element.

3560 The <PermittedLocation> element, of the **PermittedLocationType**, identifies the precise geographical  
3561 location where the symmetric key in question may be used. The <PermittedLocation> SHALL contain  
3562 the following elements:

- 3563 1. The <LocationName> element identifies a human-readable name of the physical location. It is  
3564 an XSD **String** type element, with a maximum length of 256 characters.

3565 There SHALL be only one <LocationName> element within a <PermittedLocation> element.  
3566

3567 2. An optional **LocationCoordinateGroup** which, when present, SHALL contain the following two  
3568 elements:

3569 a) The <Latitude> element of XSD **Decimal** type, that identifies the horizontal coordinate  
3570 location of the client device on the Earth, measured in *degrees* and expressed as a  
3571 decimal with the *minutes* and *seconds* part of the measurement expressed as a single  
3572 fraction.

3573  
3574 When used, there SHALL be only one <Latitude> element within the  
3575 <PermittedLocation> element.

3576 b) The <Longitude> element of XSD **Decimal** type, that identifies the vertical coordinate  
3577 location of the client device on the Earth, measured in *degrees* and expressed as a  
3578 decimal with the *minutes* and *seconds* part of the measurement expressed as a single  
3579 fraction.

3580  
3581 When used, there SHALL be only one <Longitude> element within the  
3582 <PermittedLocation> element.

3583 Some examples of the <PermittedLocations> element are shown below; other parts of their enclosing  
3584 elements are not shown for brevity:

3585 **Example 1 – An example of a <PermittedLocations> element specifying that the symmetric key**  
3586 **may be used only by applications at a single named location:**

```
3587 <ekmi:PermittedLocations ekmi:any="false">  
3588   <ekmi:PermittedLocation>  
3589     <ekmi:LocationName>StrongAuth Server Room</ekmi:LocationName>  
3590   </ekmi:PermittedLocation>  
3591 </ekmi:PermittedLocations>
```

3592 **Example 2 – An example of a <PermittedLocations> element specifying that the symmetric key**  
3593 **may be used only by applications at a single location at the given GPS coordinates:**

```
3594 <ekmi:PermittedLocations ekmi:any="false">  
3595   <ekmi:PermittedLocation>  
3596     <ekmi:LocationName>StrongAuth Server Room</ekmi:LocationName>  
3597     <ekmi:Latitude>37.385653 </ekmi:Latitude>  
3598     <ekmi:Longitude>-121.993192 </ekmi:Longitude>  
3599   </ekmi:PermittedLocation>  
3600 </ekmi:PermittedLocations>
```

3601 **Example 3 – An example of a <PermittedLocations> element specifying that the symmetric key**  
3602 **may be used only by applications at multiple locations:**

```
3603 <ekmi:PermittedLocations ekmi:any="false">  
3604   <ekmi:PermittedLocation>  
3605     <ekmi:LocationName>Humongous Headquarters</ekmi:LocationName>  
3606   </ekmi:PermittedLocation>  
3607   <ekmi:PermittedLocation>  
3608     <ekmi:LocationName> Humongous Primary Data  
3609 Center</ekmi:LocationName>  
3610     <ekmi:Latitude>37.385653 </ekmi:Latitude>  
3611     <ekmi:Longitude>-121.993192 </ekmi:Longitude>  
3612   </ekmi:PermittedLocation>  
3613   <ekmi:PermittedLocation>  
3614     <ekmi:LocationName>Humongous DR Data Center</ekmi:LocationName>  
3615     <ekmi:Latitude>68.845901 </ekmi:Latitude>
```



```

3616         <ekmi:Longitude>11.393385 </ekmi:Longitude>
3617     </ekmi:PermittedLocation>
3618 </ekmi:PermittedLocations>

```

3619 **Example 4 – An example of a null <PermittedLocations> element specifying that the symmetric**  
3620 **key may be used at any location on the planet:**

```

3621     <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>

```

## 3622 4.22 Element <PermittedNumberOfTransactions>

3623 The element <PermittedNumberOfTransactions>, of type *PermittedNumberOfTransactionsType* is  
3624 used to define the number of **encryption** transactions that applications are permitted with a symmetric  
3625 key within a specific <Symkey> element, once the **SKCL** has started using the symmetric key in question.  
3626 It does not limit the number of **decryption** transactions with the same symmetric key.

### 3627 Schema Definition:

```

3628     <xsd:complexType name="PermittedNumberOfTransactionsType">
3629         <xsd:simpleContent>
3630             <xsd:extension base="ekmi:NumberOfTransactionsType">
3631                 <xsd:attribute ref="ekmi:any" use="required"/>
3632             </xsd:extension>
3633         </xsd:simpleContent>
3634     </xsd:complexType>

```

### 3635 Schema Definition:

```

3636     <xsd:simpleType name="NumberOfTransactionsType">
3637         <xsd:restriction base="xsd:positiveInteger">
3638             <xsd:minInclusive value="1"/>
3639             <xsd:maxInclusive value="18446744073709551615"/>
3640         </xsd:restriction>
3641     </xsd:simpleType>

```

3642 There SHALL be only one <PermittedNumberOfTransactions> element within the <Permissions>  
3643 element.

3644 The <PermittedNumberOfTransactions> element SHALL have one attribute named “any”, that will have  
3645 a “false” or “true” value, based on the following:

- 3646 • When the <PermittedNumberOfTransactions> element is null (i.e. it does not have any content  
3647 in it), the value of the “any” attribute SHALL be set to “true” AND the XML Schema Instance (XSI)  
3648 “nil” attribute SHALL be set to “true”.
- 3649 • When the <PermittedNumberOfTransactions> element is not-null (i.e. it has a positive integer  
3650 content in it), the value of the “any” attribute SHALL be set to “false” AND the XML Schema  
3651 Instance (XSI) “nil” attribute SHALL NOT be present.

3652 A null <PermittedNumberOfTransactions> element specifies that applications are permitted use of the  
3653 symmetric key for an unlimited number of encryption transactions, subject to complying with all other  
3654 permission clauses in the <Permissions> element.

3655 The value of <PermittedNumberOfTransactions> element, of the XSD *positiveInteger* type, MAY  
3656 range between 1 and 18446744073709551615.

3657 Some examples of the <PermittedNumberOfTransactions> element are shown below; other parts of  
3658 their enclosing elements are not shown for brevity:

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

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

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

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

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

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

## 4.23 Element <PermittedTimes> and <PermittedTime>

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

3708 • When the <PermittedTimes> element is null (i.e. it does not have a single <PermittedTime>  
3709 sub-element in it), the value of the “any” attribute SHALL be set to “true” AND the XML Schema  
3710 Instance (XSI) “nil” attribute SHALL be set to “true”.

3711 • When the <PermittedTimes> element is not-null (i.e. it has at least one <PermittedTime> sub-  
3712 element in it), the value of the “any” attribute SHALL be set to “false” AND the XML Schema  
3713 Instance (XSI) “nil” attribute SHALL NOT be present.

3714 A null <PermittedTimes> element specifies that applications are permitted use of the symmetric key at  
3715 ANY time of the day or night, subject to complying with all other permission clauses in the  
3716 <Permissions> element.

3717 The <PermittedTime> sub-element identifies an individual set of times between which application are  
3718 permitted to use the symmetric key in question. The <PermittedTime> element consists of the following  
3719 sub-elements:

3720 1. The <StartTime> element identifies the date from which applications may start using the  
3721 symmetric key in question. It is an XSD **Time** type that MUST be specified in a specific pattern  
3722 (see examples) where the first two digits specify the hour, the second two digits specify the  
3723 minutes and the last two digits specify the seconds in a 24 hour format.

3724  
3725 There SHALL be only one <StartTime> element within a <PermittedTime> element.

3726  
3727 Conforming **SKCL** implementations SHALL NOT start using the symmetric before the onset of  
3728 the <StartTime> on the client machine.

3729 2. The <EndTime> element identifies the time until which applications may use the symmetric key in  
3730 question. It is an XSD **Time** type that MUST be specified in a specific pattern (see examples)  
3731 where the first two digits specify the hour, the second two digits specify the minutes and the last  
3732 two digits specify the seconds in a 24 hour format.

3733  
3734 There SHALL be only one <EndTime> element within a <PermittedTime> element.

3735  
3736 Conforming **SKCL** implementations SHALL NOT use the symmetric after the end of the  
3737 <EndTime> on the client machine.

3738 Some examples of the <PermittedTimes> element are shown below; other parts of their enclosing  
3739 elements are not shown for brevity:

3740 **Example 1 – An example of a <PermittedTimes> element with a single<PermittedTime> element.**  
3741 **The <StartTime> specifies 9:00AM on the client machine while the <EndTime> specifies 5:00PM:**

```
3742 <ekmi:PermittedTimes ekmi:any="false">  
3743   <ekmi:PermittedTime>  
3744     <ekmi:StartTime>09:00:00</ekmi:StartTime>  
3745     <ekmi:EndTime>17:00:00</ekmi:EndTime>  
3746   </ekmi:PermittedTime>  
3747 </ekmi:PermittedTimes>
```

3748 **Example 2 – An example of a <PermittedTimes> element with two <PermittedTime> elements. For**  
3749 **the first <PermittedTime> element , the <StartTime> element specifies 6:00AM while the**  
3750 **<EndTime> element specifies 12:00 Noon. For the second <PermittedTime> element, the**  
3751 **<StartTime> element specifies 3:00 PM in the afternoon, while the <EndTime> element specifies**  
3752 **7:00PM in the evening. This policy might imply that a symmetric key with this <PermittedTimes>**  
3753 **element cannot be used during a lunch break of 12:00 Noon to 3:00PM:**

```
3754 <ekmi:PermittedTimes ekmi:any="false">  
3755   <ekmi:PermittedTime>  
3756     <ekmi:StartTime>06:00:00</ekmi:StartTime>
```

```

3757         <ekmi:EndTime>12:00:00</ekmi:EndTime>
3758     </ekmi:PermittedTime>
3759     <ekmi:PermittedTime>
3760         <ekmi:StartTime>15:00:00</ekmi:StartTime>
3761         <ekmi:EndTime>19:00:00</ekmi:EndTime>
3762     </ekmi:PermittedTime>
3763 </ekmi:PermittedTimes>

```

3764 **Example 3 – An example of a null <PermittedTimes> element, specifying that the key may be used**  
3765 **at any time:**

```

3766     <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>

```

## 3767 4.24 Element <PermittedUses> and <PermittedUse>

3768 The element <PermittedUses>, of the type *PermittedUsesType*, is used to define the specific ways in  
3769 which applications are permitted to use a symmetric key within a specific <Symkey> element.

### 3770 Schema Definition:

```

3771 <xsd:complexType name="PermittedUsesType" mixed="true">
3772     <xsd:sequence>
3773         <xsd:element name="PermittedUse" minOccurs="0" maxOccurs="unbounded">
3774             <xsd:simpleType>
3775                 <xsd:restriction base="xsd:string">
3776                     <xsd:maxLength value="256"/>
3777                     <xsd:whiteSpace value="preserve"/>
3778                 </xsd:restriction>
3779             </xsd:simpleType>
3780         </xsd:element>
3781         <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
3782     </xsd:sequence>
3783     <xsd:attribute ref="ekmi:any" use="required"/>
3784 </xsd:complexType>

```

3785 There SHALL be only one <PermittedUses> element within the <Permissions> element. However,  
3786 there MAY be an unbounded (unlimited) number of <PermittedUse> sub-elements within the  
3787 <PermittedUses> element.

3788 The <PermittedUses> element SHALL have one attribute named “any”, that will have a “false” or “true”  
3789 value, based on the following:

- 3790 • When the <PermittedUses> element is null (i.e. it does not have a single <PermittedUse> sub-  
3791 element in it), the value of the “any” attribute SHALL be set to “true” AND the XML Schema  
3792 Instance (XSI) “nil” attribute SHALL be set to “true”.
- 3793 • When the <PermittedUses> element is not-null (i.e. it has at least one <PermittedUse> sub-  
3794 element in it), the value of the “any” attribute SHALL be set to “false” AND the XML Schema  
3795 Instance (XSI) “nil” attribute SHALL NOT be present.

3796 A null <PermittedUses> element specifies that applications are permitted use of the symmetric key for  
3797 ANY purpose, subject to complying with all other permission clauses in the <Permissions> element.

3798 Examples of the <PermittedUses> element are shown below; other parts of their enclosing elements are  
3799 not shown for brevity:

3800 **Example 1 – An example of a <PermittedUses> element specifying that the symmetric key may be**  
3801 **used only by VPN applications for session encryption keys:**

```

3802     <ekmi:PermittedUses ekmi:any="false">
3803         <ekmi:PermittedUse>VPN</ekmi:PermittedUse>
3804     </ekmi:PermittedUses>

```

3805 **Example 2 – An example of a <PermittedUses> element specifying that the symmetric key may be**  
3806 **used only by applications on laptops and Personal Digital Assistants (PDA):**

```

3807     <ekmi:PermittedUses ekmi:any="false">
3808         <ekmi:PermittedUse>Laptop</ekmi:PermittedUse>
3809         <ekmi:PermittedUse>PDA</ekmi:PermittedUse>
3810     </ekmi:PermittedUses>

```

3811 **Example 3 – An example of a null <PermittedUses> element specifying that the symmetric key**  
3812 **may be used for any purpose:**

```

3813     <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>

```

## 3814 4.25 Element <KeyCachePolicyRequest>

3815 The <KeyCachePolicyRequest> element is used to request a key-cache policy from the **SKS** server , so  
3816 the client may know if and how to cache symmetric keys locally.

### 3817 Schema Definition:

```

3818     <xsd:element name="KeyCachePolicyRequest">
3819         <xsd:complexType>
3820             <xsd:element ref="ds:Signature" maxOccurs="1"/>
3821         </xsd:complexType>
3822     </xsd:element>

```

3823 The <KeyCachePolicyRequest> has one child element. The child element has the XML Digital  
3824 signature that can help the server with the identity of the requester, strong authentication and message  
3825 integrity of the request.

3826 Some examples of the use of the <SymkeyRequest> element are as follows:

### 3827 Example 1 – An example of a <KeyCachePolicyRequest>:

```

3828     <ekmi:KeyCachePolicyRequest
3829         xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
3830         <ds:Signature>...</ds:Signature>
3831     </ekmi: KeyCachePolicyRequest>

```

## 3832 4.26 Element <KeyCachePolicyResponse>

3833 The <KeyCachePolicyResponse> element is the response sent by an **SKS** Server to a client that  
3834 requested a key-cache policy through a <KeyCachePolicyRequest>. The <KeyCachePolicyResponse>  
3835 contains policy elements, which define rules that conforming implementations of the **SKCL** MUST  
3836 adhere to when caching symmetric keys sent by the **SKS** Server.

### 3837 Schema Definition:

```

3838     <xsd:element name="KeyCachePolicyResponse">
3839         <xsd:complexType>
3840             <xsd:sequence>
3841                 <xsd:element
3842                     name="KeyCachePolicy"

```

```

3843         type="ekmi:KeyCachePolicyType"
3844         minOccurs="1" maxOccurs="unbounded"/>
3845     </xsd:sequence>
3846 </xsd:complexType>
3847 </xsd:element>

```

3848 The <KeyCachePolicyResponse> element consists of a minimum of one, but an unbounded (unlimited)  
3849 number of <KeyCachePolicy> children elements.

## 3850 4.27 Element <KeyCachePolicy>

3851 The <KeyCachePolicy> element contains policy elements, which define rules that conforming  
3852 implementations of the **SKCL** MUST adhere to when caching symmetric keys sent by the **SKS** Server.

### 3853 Schema Definition:

```

3854 <xsd:element name="KeyCachePolicyResponse">
3855     <xsd:complexType>
3856         <xsd:sequence>
3857             <xsd:element
3858                 name="KeyCachePolicy"
3859                 type="ekmi:KeyCachePolicyType"
3860                 minOccurs="1" maxOccurs="unbounded"/>
3861         </xsd:sequence>
3862     </xsd:complexType>
3863 </xsd:element>

3864 <xsd:complexType name="KeyCachePolicyType" mixed="true">
3865     <xsd:sequence>
3866         <xsd:element name="KeyCachePolicyID" type="ekmi:TwoPartIDType"/>
3867         <xsd:element name="PolicyName">
3868             <xsd:simpleType>
3869                 <xsd:restriction base="xsd:string">
3870                     <xsd:maxLength value="255"/>
3871                     <xsd:whiteSpace value="preserve"/>
3872                 </xsd:restriction>
3873             </xsd:simpleType>
3874         </xsd:element>
3875         <xsd:element name="Description" nillable="true">
3876             <xsd:simpleType>
3877                 <xsd:restriction base="xsd:string">
3878                     <xsd:maxLength value="2048"/>
3879                     <xsd:whiteSpace value="preserve"/>
3880                 </xsd:restriction>
3881             </xsd:simpleType>
3882         </xsd:element>
3883         <xsd:element name="KeyClass" type="ekmi:KeyClassType"/>
3884         <xsd:element name="StartDate" type="xsd:dateTime"/>
3885         <xsd:element name="EndDate" type="xsd:dateTime" nillable="true"/>
3886         <xsd:element name="PolicyCheckInterval">
3887             <xsd:simpleType>
3888                 <xsd:restriction base="xsd:nonNegativeInteger">
3889                     <xsd:minInclusive value="0"/>
3890                     <xsd:maxInclusive value="2592000"/>
3891                 </xsd:restriction>
3892             </xsd:simpleType>
3893         </xsd:element>
3894         <xsd:element name="Status" type="ekmi:StatusType"/>
3895         <xsd:element
3896             name="NewKeysCacheDetail"

```

```

3897         type="ekmi:KeyCacheDetailType"
3898         minOccurs="0"/>
3899     <xsd:element
3900         name="UsedKeysCacheDetail"
3901         type="ekmi:KeyCacheDetailType"
3902         minOccurs="0"/>
3903 </xsd:sequence>
3904 </xsd:complexType>

```

3905 The <KeyCachePolicy> element is of the **KeyCachePolicyType** and consists of the following child  
3906 elements:

3907 1. <KeyCachePolicyID> [Required]

3908

3909 The <KeyCachePolicyID> element, of type **TwoPartIDType**, identifies the unique policy object  
3910 within the **SKMS**. There SHALL be only one <KeyCachePolicyID> element within a  
3911 <KeyCachePolicy> element.

3912

3913 The **TwoPartIDType** is specified in Section 4.11.

3914 2. <PolicyName> [Required]

3915

3916 The <PolicyName> element, of type XSD **String**, with a maximum length of 255 characters,  
3917 identifies a unique name given to this <KeyCachePolicy>. There SHALL be only one  
3918 <PolicyName> element within a <KeyCachePolicy> element.

3919 3. <Description> [Required]

3920

3921 The <Description> element, of type XSD **String**, with a maximum length of 2048 characters,  
3922 provides a human-readable description of this policy. There SHALL be only one <Description>  
3923 element within a <KeyCachePolicy> element.

3924

3925 The <Description> MAY be an empty element, but MUST exist within the <KeyCachePolicy>  
3926 element.

3927 4. <KeyClass> [Required]

3928

3929 This element of type **KeyClassType** identifies the key-class of the symmetric key to which this  
3930 policy applies.

3931 5. <StartDate> [Required]

3932

3933 The <StartDate> element, of type XSD **dateTime**, specifies the date and time at which this  
3934 policy becomes effective. There SHALL be only one <StartDate> element within a  
3935 <KeyCachePolicy> element.

3936 6. <EndDate> [Required]

3937

3938 The <EndDate> element, of type XSD **dateTime**, specifies the date and time at which this policy  
3939 expires. There SHALL be only one <EndDate> element within a <KeyCachePolicy> element.

3940

3941 The <EndDate> MAY be an empty element, but MUST exist within the <KeyCachePolicy>  
3942 element.

3943 7. <PolicyCheckInterval> [Required]

3944

3945 The <PolicyCheckInterval> element, of type XSD **nonNegativeInteger**, specifies the  
3946 frequency at which the client SHALL check the **SKS** server for updates to this policy. This  
3947 frequency is specified in seconds and SHALL NOT exceed 2592000 seconds (30 calendar days).

3948 There SHALL be only one <PolicyCheckInterval> element within a <KeyCachePolicy>  
3949 element.

3950 8. <Status> [Required]

3951  
3952 The <Status> element, of type **StatusType**, identifies the current status of this policy within the  
3953 SKMS. There SHALL be only one <Status> element within a <KeyCachePolicy> element.

3954  
3955 The **StatusType** is specified in Section 4.14.

3956 9. <NewKeysCacheDetail> [Required]

3957  
3958 The <NewKeysCacheDetail> element, of type **KeyCacheDetailType**, defines how many new (as  
3959 yet unused for any encryption transaction) symmetric keys a client may cache, and for how long.  
3960 It is the responsibility of the conforming **SKCL** implementation to enforce these rules.

3961  
3962 The absence of the <NewKeysCacheDetail> element implies that new symmetric keys SHALL  
3963 NEVER be cached on the client. New keys may be cached only when this element exists, and  
3964 SHALL conform to the rules specified in this element.

3965  
3966 When it exists, there SHALL be only one <NewKeysCacheDetail> element in a  
3967 <KeyCachePolicy> element.

3968  
3969 The **KeyCacheDetailType** is specified in Section 4.28.

3970 10. <UsedKeysCacheDetail> [Required]

3971  
3972 The <UsedKeysCacheDetail> element, of type **KeyCacheDetailType**, defines how many used  
3973 symmetric keys a client may cache, and for how long. It is the responsibility of the conforming  
3974 **SKCL** implementation to enforce these rules.

3975  
3976 The absence of the <UsedKeysCacheDetail> element implies that used symmetric keys SHALL  
3977 NEVER be cached on the client. Used keys may be cached only when this element exists, and  
3978 SHALL conform to the rules specified in this element.

3979  
3980 When it exists, there SHALL be only one <UsedKeysCacheDetail> element in a  
3981 <KeyCachePolicy> element.

3982  
3983 The **KeyCacheDetailType** is specified in Section 4.28.

3984 Some examples of the <KeyUsePolicy> element are as follows.

3985 **Example 1 – A <KeyCachePolicy> that is valid between January 01, 2008 and December 31, 2008.**  
3986 **It requires the client to check for policy updates every day and allows 3 new and 3 used keys to be**  
3987 **cached for up to 90 days:**

```
3988 <ekmi:KeyCachePolicy>
3989   <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
3990   <ekmi:PolicyName>
3991     Corporate Laptop Symmetric Key Caching Policy
3992   </ekmi:PolicyName>
3993   <ekmi:Description>
3994     This policy defines how company-issued laptops will manage
3995     symmetric keys used for file/disk encryption in each laptop's
3996     local cache. This policy must be used by all laptops that
3997     use the company EKMI.
3998   </ekmi:Description>
3999   <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
4000   <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
```



```

4001     <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
4002     <ekmi:Status>Active</ekmi:Status>
4003     <ekmi:NewKeysCacheDetail>
4004         <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
4005         <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
4006     </ekmi:NewKeysCacheDetail>
4007     <ekmi:UsedKeysCacheDetail>
4008         <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
4009         <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
4010     </ekmi:UsedKeysCacheDetail>
4011 </ekmi:KeyCachePolicy>

```

4012 **Example 2 – A <KeyCachePolicy> that is effective starting January 01, 2008 and never expires. It**  
4013 **does NOT permit any caching of symmetric keys through the absence of the detail elements on**  
4014 **caching:**

```

4015     <ekmi:KeyCachePolicy>
4016         <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
4017         <ekmi:PolicyName>
4018             No Caching Policy
4019         </ekmi:PolicyName>
4020         <ekmi:Description>
4021             This policy is for high-risk, always-connected machines on the
4022             network, which will never cache symmetric keys locally. This
4023             policy never expires (but checks monthly for any updates).
4024         </ekmi:Description>
4025         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
4026         <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
4027         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
4028         <ekmi:Status>Active</ekmi:Status>
4029     </ekmi:KeyCachePolicy>

```

## 4030 4.28 Type *KeyCacheDetailType*

4031 The ***KeyCacheDetailType*** type allows **SKS** servers to specify precisely how many symmetric keys  
4032 MAY be cached on the client machine, and for how long.

### 4033 Schema Definition:

```

4034     <xsd:complexType name="KeyCacheDetailType">
4035         <xsd:sequence>
4036             <xsd:element name="MaximumKeys" minOccurs="1">
4037                 <xsd:simpleType>
4038                     <xsd:restriction base="xsd:integer">
4039                         <xsd:minInclusive value="0"/>
4040                         <xsd:maxInclusive value="18446744073709551615"/>
4041                     </xsd:restriction>
4042                 </xsd:simpleType>
4043             </xsd:element>
4044             <xsd:element name="MaximumDuration" minOccurs="1">
4045                 <xsd:simpleType>
4046                     <xsd:restriction base="xsd:integer">
4047                         <xsd:minInclusive value="0"/>
4048                         <xsd:maxInclusive value="18446744073709551615"/>
4049                     </xsd:restriction>
4050                 </xsd:simpleType>
4051             </xsd:element>
4052         </xsd:sequence>
4053     </xsd:complexType>

```

4054 The **KeyCacheDetailType** consists of the following child elements:

4055 1. <MaximumKeys> [Required]

4056 The <MaximumKeys> element, of type XSD **Integer**, specifies the maximum number of symmetric  
4057 keys that MAY be cached on a client machine. It SHALL be a positive number between the  
4058 values 0 and 18446744073709551615. There SHALL be only one <MaximumKeys> element  
4059 within an element that uses the **KeyCacheDetailType**.  
4060

4061 2. <MaximumDuration> [Required]

4062 The <MaximumDuration> element, of type XSD **Integer**, specifies the maximum number of  
4063 seconds that symmetric keys MAY be cached on a client machine. It SHALL be a positive  
4064 number between the values 0 and 18446744073709551615. There SHALL be only one  
4065 <MaximumDuration> element within an element that uses the **KeyCacheDetailType**.  
4066

4067 Examples of the **KeyCacheDetailType** when used in the <KeyCachePolicy> element are as follows.

4068 **Example 1 – A <KeyCachePolicy> that is valid between January 01, 2008 and December 31, 2008.**  
4069 **It requires the client to check for policy updates every day and allows 3 new and 3 used keys to be**  
4070 **cached for up to 90 days:**

```
4071 <ekmi:KeyCachePolicy>
4072   <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
4073   <ekmi:PolicyName>
4074     Corporate Laptop Symmetric Key Caching Policy
4075   </ekmi:PolicyName>
4076   <ekmi:Description>
4077     This policy defines how company-issued laptops will manage
4078     symmetric keys used for file/disk encryption in their local
4079     cache. This policy must be used by all laptops that use
4080     the company EKMI.
4081   </ekmi:Description>
4082   <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
4083   <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
4084   <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
4085   <ekmi:Status>Active</ekmi:Status>
4086   <ekmi:NewKeysCacheDetail>
4087     <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
4088     <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
4089   </ekmi:NewKeysCacheDetail>
4090   <ekmi:UsedKeysCacheDetail>
4091     <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
4092     <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
4093   </ekmi:UsedKeysCacheDetail>
4094 </ekmi:KeyCachePolicy>
```

4095 **Example 1 – A <KeyCachePolicy> that is valid between January 01, 2008 and December 31, 2008.**  
4096 **It requires the client to check for policy updates every day and allows 1 new and 0 used keys to be**  
4097 **cached for upto 15 days:**

```
4098 <ekmi:KeyCachePolicy>
4099   <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
4100   <ekmi:PolicyName>
4101     Corporate Laptop Symmetric Key Caching Policy
4102   </ekmi:PolicyName>
4103   <ekmi:Description>
4104     This policy defines how company-issued laptops will manage
4105     symmetric keys used for file/disk encryption in each laptop's
4106     local cache. This policy must be used by all laptops that
```

```

4107         use the company EKMI.
4108     </ekmi:Description>
4109     <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
4110     <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
4111     <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
4112     <ekmi:Status>Active</ekmi:Status>
4113     <ekmi:NewKeysCacheDetail>
4114         <ekmi:MaximumKeys>1</ekmi:MaximumKeys>
4115         <ekmi:MaximumDuration>1296000</ekmi:MaximumDuration>
4116     </ekmi:NewKeysCacheDetail>
4117     <ekmi:UsedKeysCacheDetail>
4118         <ekmi:MaximumKeys>0</ekmi:MaximumKeys>
4119         <ekmi:MaximumDuration>1296000</ekmi:MaximumDuration>
4120     </ekmi:UsedKeysCacheDetail>
4121 </ekmi:KeyCachePolicy>

```

## 4122 4.29 Use of Web Services Security (WSS)

4123 While it has been mentioned earlier in this specification, it is explicitly noted here that conforming  
4124 implementations of the SKSML protocol MAY enclose all SKSML messages between participants in an  
4125 SKMS in the **SOAP Body** of a **SOAP Envelope**. Additionally, the contents of the **SOAP Body** MUST be  
4126 secured using digital signatures conforming to [XMLSignature] in the **SOAP Header**. Specifically, the  
4127 **SOAP Header** of the **SOAP Envelope** MUST enclose a **Security** element conforming to [WSS] with a  
4128 **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  
4129 specified security profile in [WSS] to form a well-formed, secure message.  
4130

4131 While the payload (the symmetric-key in the <Symkey> element ) is already encrypted by the SKS server  
4132 using the Public Key of the requesting client, the contents of the **SOAP Body** MAY, additionally, be  
4133 secured with encryption using [XMLEncryption], if desired. The choice of using the requesting client's  
4134 Public Key or a SOAP-layer Public Key to encrypt the **SOAP Body** is left to the implementers of the  
4135 SKMS.

4136 The SOAP security objects must be layered on the SKSML messages **after** the SKSML message has  
4137 been fully constructed (and secured in the case of the symmetric-key payload) by the SKS server.

4138 *Note: In the event that SKMS sites choose to encrypt the SOAP Body's contents at the SOAP layer (in*  
4139 *addition to the symmetric key encryption performed at the SKSML layer), the **Security** element in some*  
4140 *implementations of [WSS] will have two <BinarySecurityToken> elements with digital certificates in them:*  
4141 *one for the sender's signing certificate, and the other for the recipient's encryption certificate. While*  
4142 *SKMS sites have the choice of encrypting the contents of the SOAP Body, they do NOT have a choice*  
4143 *about signing the contents of the SOAP Body - all SKSML messages enclosed in the SOAP Body MUST*  
4144 *be signed by clients and servers within an SKMS.*

## 4145 4.30 Use of SKMS Error Codes & Messages

4146 SKSML defines many messages between the Symmetric Key Client Library (SKCL) implementations and  
4147 Symmetric Key Services (SKS) servers. Given the complexity of layered-technology, there are ample  
4148 opportunities for components to fail during processing.

4149 The EKMI TC believes that adoption of this technology can be encouraged by standardizing codes for  
4150 errors and informational messages used by SKMS clients and servers. At the same time, the TC  
4151 recognizes the value of providing flexibility for vendor implementations to innovate beyond what the TC  
4152 has envisioned.

4153 To ensure a baseline level of conformance, the TC has chosen to standardize some common error-codes  
4154 and messages, while creating a process for vendors to request and receive a block of codes reserved for

4155 the vendor's use within their implementation of SKSML. These codes are defined in **Appendix C** and the  
4156 process is described in **Appendix D** of this specification.

4157 Conforming implementations of SKSML MUST implement the Standard Error Codes and Message within  
4158 their implementations. It is left up to the vendor to determine if they wish to apply for, and receive, a  
4159 reserved block of codes for their own implementation's unique codes and messages.

---

## 5 Bindings

An SKSML implementation can provide symmetric key services over a wide variety of transport mechanisms. This is referred to as “Bindings” of this specification. To maintain compliance with the specification, implementations must support the bindings that are REQUIRED.

### 5.1 W3C Security Binding

An SKSML implementation is REQUIRED to provide a binding that transports SKSML messages with support for XML Signatures and XML Encryption.

### 5.2 Mutually Authenticated TLS Binding

An SKSML implementation is REQUIRED to provide a binding that transports SKSML messages over a TLS connection that is mutually authenticated.

### 5.3 SOAP-WSS Binding

An SKSML implementation MAY provide SOAP and Web Services Security [WSS] support for transporting SKSML messages.

---

## 6 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	<ul style="list-style-type: none"> <li>- Moved non-normative sections to their own document.</li> <li>- KeyClass element was added to KeyCachePolicy.</li> <li>- KeyCachePolicy is now embedded inside a KeyCachePolicyResponse.</li> </ul>
DRAFT 6	July 7, 2008	Arshad Noor	<ul style="list-style-type: none"> <li>- Modified Permissions object to include all sub-elements on a mandatory basis.</li> <li>- Modified all abbreviations in elements to expand to full names.</li> </ul>
PR 1	July 22, 2008	Arshad Noor	<ul style="list-style-type: none"> <li>- Modified Title page information to conform with OASIS standards.</li> <li>- Brought some Background information into this document from the information document, into Section 2.</li> <li>- Added a Conformance section to Section 4.</li> </ul>
PR 2	11/04/08	Arshad Noor	<ul style="list-style-type: none"> <li>- Added SymkeyWorkInProgress, SymkeyRequestID and RequestCheckInterval elements to support sending and receiving request/responses asynchronously.</li> <li>- Modified SymkeyResponse to include SymkeyWorkInProgress as a valid response to a key request.</li> <li>- Modified Symkey to include the SymkeyRequestID to support asynchronous request/responses.</li> <li>- Modified SymkeyError to include the SymkeyRequestID to support asynchronous request/responses.</li> <li>- Modified SymkeyRequest to send a SymkeyRequestID to poll the SKS server on the status of a symmetric-key request.</li> <li>- Added Standard Error Codes &amp; Messages in Appendix C.</li> <li>- Added Appendix D to define the process for vendors to apply for a reserved block of codes.</li> </ul>
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	<ul style="list-style-type: none"> <li>- 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</li> </ul>
PR 3 (2.3)	08/05/10	Anil Saldhana	<ul style="list-style-type: none"> <li>- Made the requirement for SOAP/WSS optional as per TC decision. SOAP/WSS will go into a separate profile of SKSML.</li> </ul>



## 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

4239

4240 *Note: The {0} symbol at the end of each message is a placeholder for a parameter that can be used by*  
4241 *implementations for adding additional information pertaining to the error. The additional information will*  
4242 *be useful to administrators and software developers in helping them focus on the part of the system*  
4243 *where the underlying problem has manifested itself.*

## 4244 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}

4245

## 4246 Authorization ERROR Codes (00100 - 00199)

Code	Message
SKMS-ERR-00101	Authorization failure – invalid signature: {0}
SKMS-ERR-00102	Authorization failure – invalid status: {0}
SKMS-ERR-00103	Authorization failure – invalid requester: {0}

Code	Message
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}

4247

## 4248 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}
SKMS-ERR-00203	Cryptographic-module failure – invalid cryptographic provider: {0}

Code	Message
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}

4249

## 4250 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}
SKMS-ERR-00309	Key-cache failure – unauthorized write-access to cache: {0}
SKMS-ERR-00399	Key-cache failure – other key-cache error: {0}

4251

4252

## 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}

4253

4254

## 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}
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}

Code	Message
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}

4255

## 4256 **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}
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}

Code	Message
SKMS-ERR-00623	SKS error – database layer error: {0}
SKMS-ERR-00699	SKS error - other SKS error: {0}

4257

4258

## 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}

4259

4260

## 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](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
8. In the event that a Vendor does not implement the protocol within the 6-12 month period, the TC will vote and reclaim the private-block assigned to the original requester. Any use of the private-



4305 block of assigned numbers, after the TC has voted to reclaim them, will be a violation of the TC's  
4306 guidelines for the SKSML Specification. OASIS Administration will then be directed by this TC to  
4307 take whatever action it is legally permitted to address this violation.