



# [Symmetric Key Services Markup Language (SKSML) Version 1.0 Normative DRAFT 5]

OASIS Technical Committee DRAFT

17 June 2008

## Specification URIs:

### This Version:

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

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

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

### Previous Version:

None

### Latest Version:

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

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

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

### Latest Approved Version:

None

### Technical Committee:

OASIS Enterprise Key Management Infrastructure (EKMI) TC

### Chair(s):

Arshad Noor, StrongAuth, Inc. ([arshad.noor@strongauth.com](mailto:arshad.noor@strongauth.com))

### Editor(s):

Allen Schaaf ([netsecurity@sound-by-design.com](mailto:netsecurity@sound-by-design.com))

### Related Work:

This specification replaces or supercedes:

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

This specification is related to:

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

### Declared XML Namespace(s):

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

33 **Abstract:**

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

36 **Status:**

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

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

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

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

---

# Notices

49

50 Copyright © OASIS® 2008. All Rights Reserved.

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

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

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

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

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

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

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

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

91

## 92 Table of Contents

93	1 Introduction.....	5
94	1.1 Terminology.....	5
95	1.2 Glossary.....	5
96	1.3 Normative References.....	6
97	2 Specification.....	7
98	2.1 Element <SymkeyRequest>.....	7
99	2.2 Element <GlobalKeyID>.....	9
100	2.3 Element <KeyClasses> and <KeyClass>.....	10
101	2.4 Element <SymkeyResponse>.....	12
102	2.5 Element <Symkey>.....	14
103	2.6 Element <SymkeyError>.....	16
104	2.7 Element <KeyUsePolicy>.....	17
105	2.8 Type TwoPartIDType.....	20
106	2.9 Element <KeyAlgorithm>.....	20
107	2.10 Element <KeySize>.....	21
108	2.11 Element <Status>.....	22
109	2.12 Element <Permissions>.....	23
110	2.13 Element <PermittedApplications> and <PermittedApplication>.....	28
111	2.14 Element <PermittedDates> and <PermittedDate>.....	30
112	2.15 Element <PermittedDays> and <PermittedDay>.....	32
113	2.16 Element <PermittedDuration>.....	33
114	2.17 Element <PermittedLevels> and <PermittedLevel>.....	34
115	2.18 Element <PermittedLocations> and <PermittedLocation>.....	35
116	2.19 Element <PermittedTimes> and <PermittedTime>.....	37
117	2.20 Element <PermittedTransactions>.....	39
118	2.21 Element <PermittedUses> and <PermittedUse>.....	39
119	2.22 Element <KeyCachePolicyRequest>.....	40
120	2.23 Element <KeyCachePolicyResponse>.....	41
121	2.24 Element <KeyCachePolicy>.....	41
122	2.25 Type KeyCacheDetailType.....	44
123		

---

# 124 1 Introduction

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

## 128 1.1 Terminology

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

## 132 1.2 Glossary

133 **3DES** – Triple Data Encryption Standard

134 **AES** – Advanced Encryption Standard

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

136 **Ciphertext** – Encrypted data

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

139 **DES** – Data Encryption Standard

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

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

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

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

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

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

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

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

156 **Plaintext** – Unencrypted data

157 **SHA** – Secure Hashing Algorithm

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

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

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

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

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

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

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

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

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

169 **SOAP** – Simple Object Access Protocol

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

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

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

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

176 **Symkey** - A symmetric encryption key

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

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

### 181 **1.3 Normative References**

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

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

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

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

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

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

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

198

199

---

## 2 Specification

### 2.1 Element <SymkeyRequest>

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

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

#### Schema Definition:

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

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

1. <GlobalKeyID> [Required]

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

The <GlobalKeyID> element is specified in Section 2.2.

2. <KeyClasses> [Optional]

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

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

When the client requires more than one new symmetric key, and each key needs to be of a different key

250 class, there MUST be only one <GlobalKeyID> element followed by as many <KeyClass> elements  
251 insides the <KeyClasses> set, as needed by the client application.

252  
253 When a client requires many symmetric keys of two or more key classes, the client MUST send multiple  
254 requests to the SKS server. See examples 4 and 5 below in this section.

255  
256 The <KeyClasses> and <KeyClass> elements are specified in Section 2.3.

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

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

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

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

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

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

```
269 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
270 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
271 <ekmi:KeyClasses>  
272 <ekmi:KeyClass>HR-Class</ekmi:KeyClass>  
273 </ekmi:KeyClasses>  
274 </ekmi:SymkeyRequest>
```

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

```
276 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
277 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
278 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
279 <ekmi:KeyClasses>  
280 <ekmi:KeyClass>FIN-FX</ekmi:KeyClass>  
281 </ekmi:KeyClasses>  
282 </ekmi:SymkeyRequest>
```

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

```
284 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
285 <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
286 <ekmi:KeyClasses>  
287 <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>  
288 <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>  
289 <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>  
290 <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>  
291 <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>  
292 <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>  
293 <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>  
294 <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>  
295 <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>  
296 </ekmi:KeyClasses>  
297 </ekmi:SymkeyRequest>
```



## 298 2.2 Element <GlobalKeyID>

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

### 302 Schema Definition:

```
303 <xsd:simpleType name="GlobalKeyIDType">  
304   <xsd:restriction base="xsd:string">  
305     <xsd:minLength value="5"/>  
306     <xsd:maxLength value="62"/>  
307     <xsd:pattern value="[0-9]{1,20}-[0-9]{1,20}-[0-9]{1,20}"/>  
308     <xsd:whiteSpace value="collapse"/>  
309   </xsd:restriction>  
310 </xsd:simpleType>
```

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

- 313 1. A non-negative integer identifying the *Domain ID*. The *DomainID* identifies the IANA-issued Private  
314 Enterprise Number (PEN) as published at <http://www.iana.org/assignments/enterprise-numbers> and is  
315 used by the SKS server to constrain the ownership of objects within the SKMS:
- 316 2. A literal hyphen ("-");
- 317 3. A non-negative integer identifying the Server ID of the server that originally generated the key;
- 318 4. Another literal hyphen ("-");
- 319 5. A non-negative integer identifying the Key ID;

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

323 The *DomainID* part of the <GlobalKeyID> element MUST be a non-negative integer in the range of 0 (zero) to  
324 18446744073709551615 (20-byte ASCII decimal).

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

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

332 The *ServerID* part of the <GlobalKeyID> element MUST be a non-negative integer in the range of 0 (zero) to  
333 18446744073709551615 (20-byte ASCII decimal).

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

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

338 The *KeyID* part of the <GlobalKeyID> element MUST be a non-negative integer in the range of 0 (zero) to  
339 18446744073709551615 (20-byte ASCII decimal).

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

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

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

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

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

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

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

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

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

```
354     <ekmi:GlobalKeyID>  
355     18446744073709551615-18446744073709551615-18446744073709551615  
356     </ekmi:GlobalKeyID>
```

## 357 2.3 Element <KeyClasses> and <KeyClass>

358 The <KeyClasses> element of type **KeyClassesType**, when specified, identifies at least one <KeyClass>  
359 element, but may specify an unbounded number of <KeyClass> elements within the <KeyClasses> set.

### 360 Schema Definition:

```
361     <xsd:complexType name="KeyClassesType">  
362         <xsd:sequence>  
363             <xsd:element  
364                 name="KeyClass"  
365                 type="tns:KeyClassType"  
366                 minOccurs="1"  
367                 maxOccurs="unbounded"/>  
368         </xsd:sequence>  
369     </xsd:complexType>  
  
370  
371     <xsd:simpleType name="KeyClassType">  
372         <xsd:restriction base="xsd:string">  
373             <xsd:maxLength value="255"/>  
374         </xsd:restriction>  
375     </xsd:simpleType>
```

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

379

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

386

387 When more than one <GlobalKeyID> is specified in the request, there MAY be only one <KeyClass> element

388 within the <KeyClasses> set. When a key class is not specified in a request, it implies a request for symmetric  
389 key(s) of a default key class configured at the SKS server. The default key class for a site is site-specific.

390  
391 When the client requires more than one symmetric key, and each key needs to be of a different key class, there  
392 MUST be only one <GlobalKeyID> element followed by as many <KeyClass> elements inside the <KeyClasses>  
393 set as needed by the client application. (Example 5 in this section).

394  
395 When a client requires many symmetric keys – say five keys – and two or more keys belong to the same key  
396 class, the client MUST send multiple requests to the SKS server. One request will contain multiple  
397 <GlobalKeyID> elements with one <KeyClass> element in the <KeyClasses> set, and the other request will  
398 contain one <GlobalKeyID> element and multiple <KeyClass> elements within the <KeyClasses> set. (Examples  
399 4 and 5 in this section).

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

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

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

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

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

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

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

```
419 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
420   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
421   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
422   <ekmi:KeyClasses>  
423     <ekmi:KeyClass>FIN-FX</ekmi:KeyClass>  
424   </ekmi:KeyClasses>  
425 </ekmi:SymkeyRequest>
```

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

```
427 <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">  
428   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>  
429   <ekmi:KeyClasses>  
430     <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>  
431     <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>  
432     <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>  
433     <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>  
434   </ekmi:KeyClasses>  
435 </ekmi:SymkeyRequest>
```

## 436 2.4 Element <SymkeyResponse>

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

440 While <SymkeyResponse> is a top-level element within this specification, it MUST be enclosed within a **SOAP**  
441 **Body** element of a **SOAP Envelope** to conform to the security requirements of this specification. The **SOAP**  
442 **Header** of the **SOAP Envelope** MUST enclose a **Security** element conforming to [WSS] with a **ValueType**  
443 attribute containing the value <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3>. The **Security** element must conform to all other requirements of the specified security profile  
444 in [WSS] to form a well-formed, secure message.  
445

### 446 Schema Definition:

```
447 <xsd:element name="SymkeyResponse">  
448   <xsd:complexType>  
449     <xsd:sequence>  
450       <xsd:element  
451         name="Symkey"  
452         type="tns:SymkeyType"  
453         minOccurs="0"  
454         maxOccurs="unbounded"/>  
455       <xsd:element  
456         name="SymkeyError"  
457         type="tns:SymkeyErrorType"  
458         minOccurs="0"  
459         maxOccurs="unbounded"/>  
460     </xsd:sequence>  
461   </xsd:complexType>  
462 </xsd:element>
```

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

#### 467 1. <Symkey> [Optional]

468  
469 This element of type **SymkeyType**, is returned by the SKS server in response to a successful  
470 processing of a <SymkeyRequest>. There MAY be more than one <Symkey> element in the  
471 <SymkeyResponse> if the client application made a request for multiple symmetric keys.  
472

473 The <Symkey> element and the **SymkeyType** are specified in Section 2.5.

#### 474 2. <SymkeyError> [Optional]

475  
476 This element of type **SymkeyErrorType**, contains a response to a failed attempt in processing a  
477 request for one or more symmetric keys. There MAY be more than one <SymkeyError> element in the  
478 <SymkeyResponse> if the client application made a request for multiple symmetric keys and the request  
479 resulted in multiple errors.  
480

481 The <SymkeyError> element is specified in Section 2.6.

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

#### 483 Example 1 – A single symmetric key within a response:

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

```
486     <ekmi:Symkey>.....</ekmi:Symkey>
487 </ekmi:SymkeyResponse>
```

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

```
489 <ekmi:SymkeyResponse
490   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
491   <ekmi:Symkey>.....</ekmi:Symkey>
492   <ekmi:Symkey>.....</ekmi:Symkey>
493   <ekmi:Symkey>.....</ekmi:Symkey>
494 </ekmi:SymkeyResponse>
```

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

```
496 <ekmi:SymkeyResponse
497   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
498   <ekmi:SymkeyError>.....</ekmi:SymkeyError>
499 </ekmi:SymkeyResponse>
```

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

```
501 <ekmi:SymkeyResponse
502   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
503   <ekmi:SymkeyError>.....</ekmi:SymkeyError>
504   <ekmi:SymkeyError>.....</ekmi:SymkeyError>
505 </ekmi:SymkeyResponse>
```

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

```
507 <ekmi:SymkeyRequest
508   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
509   <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
510   <ekmi:KeyClasses>
511     <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
512     <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
513     <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
514     <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
515     <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
516     <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
517     <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
518     <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
519     <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>
520   </ekmi:KeyClasses>
521 </ekmi:SymkeyRequest>
```

522 **Example 6 – A response with one symmetric key and one error:**

```
523 <ekmi:SymkeyResponse
524   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
525   <ekmi:Symkey>.....</ekmi:Symkey>
526   <ekmi:SymkeyError>.....</ekmi:SymkeyError>
527 </ekmi:SymkeyResponse>
```

528 **Example 7 – A response with multiple symmetric keys and multiple error:**

```
529 <ekmi:SymkeyResponse
530   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
531   <ekmi:Symkey>.....</ekmi:Symkey>
532   <ekmi:Symkey>.....</ekmi:Symkey>
533   <ekmi:Symkey>.....</ekmi:Symkey>
534   <ekmi:SymkeyError>.....</ekmi:SymkeyError>
535   <ekmi:SymkeyError>.....</ekmi:SymkeyError>
536 </ekmi:SymkeyResponse>
```

## 537 2.5 Element <Symkey>

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

### 541 Schema Definition:

```
542 <xsd:complexType name="SymkeyType">  
543   <xsd:sequence>  
544     <xsd:element name="GlobalKeyID" type="ekmi:GlobalKeyIDType"/>  
545     <xsd:element name="KeyUsePolicy" type="ekmi:KeyUsePolicyType"/>  
546     <xsd:element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>  
547     <xsd:element ref="xenc:CipherData"/>  
548     <xsd:element ref="xenc:EncryptionProperties" minOccurs="0"/>  
549   </xsd:sequence>  
550 </xsd:complexType>
```

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

554 In the event of an error in processing the request, there SHALL be no <Symkey> element in the response; there  
555 SHALL be a <SymkeyError> element, instead. The <SymkeyError> element is specified in Section 2.6.

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

557 1. <GlobalKeyID> [Required]

558  
559 This element of type **GlobalKeyIDType** identifies the unique identifier of the symmetric key within an  
560 SKMS. There SHALL be only one <GlobalKeyID> within a <Symkey> element.

561 The **GlobalKeyIDType** is specified in Section 2.2.

563 2. <KeyUsePolicy> [Required]

564  
565 This element of type **KeyUsePolicyType**, defines how the symmetric key in this <Symkey> element may  
566 be used by applications. There SHALL be only one <KeyUsePolicy> element within a <Symkey>  
567 element.

568 The <KeyUsePolicy> element is specified in Section 2.7.

570 3. <EncryptionMethod> [Required]

571  
572 This element of type **EncryptionMethodType** from [XMLEncryption] describes how the symmetric key  
573 in this <Symkey> element is encrypted for transport between the SKS Server and the client application.

574  
575 The <EncryptionMethod> MUST specify one of the following two transport algorithms in the **Algorithm**  
576 attribute of the element:

- 577  
578 - [http://www.w3.org/2001/04/xmlenc#rsa-1\\_5](http://www.w3.org/2001/04/xmlenc#rsa-1_5)  
579 - <http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p>

580 4. <CipherData> [Required]

581  
582 This element of **CipherDataType** from [XMLEncryption] contains the encrypted symmetric-key. As  
583 specified in [XMLEncryption], the content of this element is Base-64 encoded and is of the XML Schema  
584 **base64Binary** type.

585 <EncryptionProperties> [Optional]

586  
587 This element of type **EncryptionPropertiesType** from [XMLEncryption] allows for any additional

588 properties that the SKS Server may wish to provide the client as hints towards decrypting the cipher-  
589 text in the <Symkey> element.

590  
591 It is currently envisioned that there is no use for this optional element in the SKSML protocol. However,  
592 since it is difficult to predict the future with any degree of accuracy, we leave this element in the schema  
593 to avoid having to make a revision to the specification for such a minor detail. Implementers are  
594 strongly encouraged to contact the EKMI Technical Committee at OASIS to voice their comments if  
595 they believe this element to be of value and require it to be a mandatory element.

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

598 **Example 1 – A symmetric key within a response:**

```
599 <ekmi:SymkeyResponse
600   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
601   <ekmi:Symkey>
602     <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
603     <ekmi:KeyUsePolicy>.....</ekmi:KeyUsePolicy>
604     <ekmi:EncryptionMethod
605       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
606     <xenc:CipherData>
607       <xenc:CipherValue>
608         E9zWB/y93hVSzeTLiDcQoDxmLNxTux+SffMNwCJmt1dIqzQHBnpdQ8
609         lg6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxj
610         fg1pU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
611       </xenc:CipherValue>
612     </xenc:CipherData>
613   </ekmi:Symkey>
614 </ekmi:SymkeyResponse>
```

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

```
616 <ekmi:SymkeyResponse
617   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
618   <ekmi:Symkey>
619     <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
620     <ekmi:KeyUsePolicy>.....</ekmi:KeyUsePolicy>
621     <ekmi:EncryptionMethod
622       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
623     <xenc:CipherData>
624       <xenc:CipherValue>
625         E9zWB/y93hVSzeTLiDcQoDxmLNxTux+SffMNwCJmt1dIqzQHBnpdQ8
626         lg6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxj
627         fg1pU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
628       </xenc:CipherValue>
629     </xenc:CipherData>
630   </ekmi:Symkey>
631   <ekmi:Symkey>
632     <ekmi:GlobalKeyID>10514-1-236</ekmi:GlobalKeyID>
633     <ekmi:KeyUsePolicy>.....</ekmi:KeyUsePolicy>
634     <ekmi:EncryptionMethod
635       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
636     <xenc:CipherData>
637       <xenc:CipherValue>
638         Qbg65cy93hVSzeTLiDcQoDxmLNxTux+SffMNwCJmt1dIqzQHBnpdQ8
639         7k6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXG0ca8EU871zBoXBjDxj
640         uyecU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
641       </xenc:CipherValue>
642     </xenc:CipherData>
```

```
643     </ekmi:Symkey>
644 </ekmi:SymkeyResponse>
```

## 645 2.6 Element <SymkeyError>

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

### 648 Schema Definition:

```
649 <xsd:complexType name="SymkeyErrorType">
650   <xsd:sequence>
651     <xsd:element name="RequestedGlobalKeyID" type="ekmi:GlobalKeyIDType"/>
652     <xsd:element
653       name="RequestedKeyClass"
654       type="ekmi:KeyClassType"
655       minOccurs="0"/>
656     <xsd:element name="ErrorCode">
657       <xsd:simpleType>
658         <xsd:restriction base="xsd:string">
659           <xsd:maxLength value="255"/>
660         </xsd:restriction>
661       </xsd:simpleType>
662     </xsd:element>
663     <xsd:element name="ErrorMessage">
664       <xsd:simpleType>
665         <xsd:restriction base="xsd:string">
666           <xsd:maxLength value="1024"/>
667         </xsd:restriction>
668       </xsd:simpleType>
669     </xsd:element>
670   </xsd:sequence>
671 </xsd:complexType>
```

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

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

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

- 679 1. <RequestedGlobalKeyID> [Required]

680  
681 This element of type *GlobalKeyIDType* identifies the unique identifier of the symmetric key requested by  
682 the client application. There SHALL be only one <RequestedGlobalKeyID> within a <SymkeyError>  
683 element.

684  
685 The *GlobalKeyIDType* is specified in Section 2.2.

- 686 2. <RequestedKeyClass> [Optional]

687  
688 This element of type *KeyClassType* identifies the key-class of the symmetric key requested by the  
689 client application. If the <RequestedKeyClass> element is not embedded in the <SymkeyError>  
690 element, this implies that the requested symmetric key was for the default key-class of the SKMS.

691  
692 The *KeyClassType* is specified in Section 2.3.



- 693 3. <ErrorCode> [Required]  
 694  
 695 This element of type **String** identifies a mnemonic code identifying the error SKS Server experienced in  
 696 processing the client's symmetric key request.  
 697  
 698 The <ErrorCode> element SHALL return one of the codes identified in Appendix D of this specification.
- 699 4. <ErrorMessage> [Required]  
 700  
 701 This element of type **String** identifies a localized message describing the error SKS Server experienced  
 702 in processing the client's symmetric key request.  
 703  
 704 The <ErrorMessage> element SHALL return the appropriate localized version of the message  
 705 corresponding to the <ErrorCode> element from Appendix D of this specification.

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

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

```
708 <ekmi:SymkeyResponse
709     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
710     <ekmi:SymkeyError>
711         <ekmi:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>
712         <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
713         <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
714     </ekmi:SymkeyError>
715 </ekmi:SymkeyResponse>
```

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

```
717 <ekmi:SymkeyResponse
718     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
719     <ekmi:SymkeyError>
720         <ekmi:RequestedGlobalKeyID>10514-1-0</ekmi:RequestedGlobalKeyID>
721         <ekmi:ErrorCode>SKS-100001</ekmi:ErrorCode>
722         <ekmi:ErrorMessage>Invalid GlobalKeyID</ekmi:ErrorMessage>
723     </ekmi:SymkeyError>
724     <ekmi:SymkeyError>
725         <ekmi:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>
726         <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
727         <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
728     </ekmi:SymkeyError>
729 </ekmi:SymkeyResponse>
```

## 730 2.7 Element <KeyUsePolicy>

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

733 **Schema Definition:**

```
734 <xsd:complexType name="KeyUsePolicyType" mixed="true">
735     <xsd:sequence>
736         <xsd:element name="KeyUsePolicyID" type="tns:TwoPartIDType"/>
737         <xsd:element name="PolicyName">
738             <xsd:simpleType>
739                 <xsd:restriction base="xsd:string">
740                     <xsd:maxLength value="255"/>
741                 </xsd:restriction>
742             </xsd:simpleType>
```

```

743         </xsd:element>
744         <xsd:element name="KeyClass" type="tns:KeyClassType"/>
745         <xsd:element name="KeyAlgorithm" type="tns:EncryptionAlgorithmType"/>
746         <xsd:element name="KeySize" type="tns:KeySizeType"/>
747         <xsd:element name="Status" type="tns:StatusType"/>
748         <xsd:element name="Permissions" type="tns:PermissionsType"/>
749     </xsd:sequence>
750 </xsd:complexType>

```

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

- 752 1. <KeyUsePolicyID> [Required]  
753  
754 The <KeyUsePolicyID> element, of type **TwoPartIDType**, identifies the unique policy object within  
755 the SKMS. There SHALL be only one <KeyUsePolicyID> element within a <KeyUsePolicy> element.  
756  
757 The **TwoPartIDType** is specified in Section 2.8.
- 758 2. <PolicyName> [Required]  
759  
760 The <PolicyName> element, of type XSD **String**, with a maximum length of 255 characters,  
761 identifies a unique name given to this <KeyUsePolicy>. There SHALL be only one <PolicyName>  
762 element within a <KeyUsePolicy> element.
- 763 3. <KeyClass> [Required]  
764  
765 The <KeyClass> element, of type **KeyClassType**, identifies a key-class assigned to this  
766 <KeyUsePolicy>. There SHALL be only one <KeyUsePolicyID> element within a <KeyUsePolicy>  
767 element.  
768  
769 The **KeyClassType** is specified in Section 2.3.
- 770 4. <KeyAlgorithm> [Required]  
771  
772 The <KeyAlgorithm> element, of type **EncryptionAlgorithmType**, identifies encryption algorithm to  
773 be used by applications when using this symmetric key. There SHALL be only one <KeyAlgorithm>  
774 element within a <KeyUsePolicy> element.  
775  
776 The <KeyAlgorithm> element is specified in Section 2.9.
- 777 5. <KeySize> [Required]  
778  
779 The <KeySize> element, of type **KeySizeType**, defines the size of the symmetric key, in bits (binary  
780 digits). There SHALL be only one <KeySize> element within a <KeyUsePolicy> element.  
781  
782 Note: It is possible to determine the size of a symmetric key in an SKCL implementation without having  
783 to send the size in the response. So, why include it? It is our belief that while network bandwidth and  
784 compute performance of devices are increasing steadily, encryption is desired in many small and  
785 portable devices. Consequently, it will speed up applications in cryptographic processing if they do not  
786 have to determine the size of each key they use. While "protocol purity" demands that implementation  
787 issues do not show up in protocol design, we believe this is justified in this case.  
788  
789 The **KeySizeType** is specified in Section 2.10.
- 790 6. <Status> [Required]  
791  
792 The <Status> element, of type **StatusType**, identifies the current status of the symmetric key. There  
793 SHALL be only one <Status> element within a <KeyUsePolicy> element.  
794  
795 The **StatusType** is specified in Section 2.11.

796 7. <Permissions> [Required]

797

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

801

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

805

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

807

808 The *PermissionsType* is specified in Section 2.12.

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

810 **Example 1 – A <KeyUsePolicy> with some permissions:**

```
811 <ekmi:KeyUsePolicy>
812   <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>
813   <ekmi:PolicyName>DES-EDE KeyUsePolicy</ekmi:PolicyName>
814   <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
815   <ekmi:KeyAlgorithm>
816     http://www.w3.org/2001/04/xmlenc#tripleDES-cbc
817   </ekmi:KeyAlgorithm>
818   <ekmi:KeySize>192</ekmi:KeySize>
819   <ekmi>Status>Active</ekmi>Status>
820   <ekmi:Permissions>
821     <ekmi:PermittedApplications>
822       <ekmi:PermittedApplication>
823         <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
824         <ekmi:ApplicationName>Payroll Application</ekmi:ApplicationName>
825         <ekmi:Version>1.0</ekmi:Version>
826         <ekmi:DigestAlgorithm>
827           http://www.w3.org/2000/09/xmldsig#sha1
828         </ekmi:DigestAlgorithm>
829         <ekmi:DigestValue>NIG4bKkt4cziEqFFu0oBTM81efU=</ekmi:DigestValue>
830       </ekmi:PermittedApplication>
831     </ekmi:PermittedApplications>
832     <ekmi:PermittedTimes>
833       <ekmi:PermittedTime>
834         <ekmi:StartTime>07:00:00</ekmi:StartTime>
835         <ekmi:EndTime>19:00:00</ekmi:EndTime>
836       </ekmi:PermittedTime>
837     </ekmi:PermittedTimes>
838   </ekmi:Permissions>
839 </ekmi:KeyUsePolicy>
```

840 **Example 2 – A <KeyUsePolicy> with some permissions:**

```
841 <ekmi:KeyUsePolicy>
842   <ekmi:KeyUsePolicyID>10514-2</ekmi:KeyUsePolicyID>
843   <ekmi:PolicyName>Laptop KeyUsePolicy</ekmi:PolicyName>
844   <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
845   <ekmi:KeyAlgorithm>
846     http://www.w3.org/2001/04/xmlenc#aes256-cbc
847   </ekmi:KeyAlgorithm>
848   <ekmi:KeySize>256</ekmi:KeySize>
849   <ekmi>Status>Active</ekmi>Status>
850   <ekmi:Permissions/>
851 </ekmi:KeyUsePolicy>
```

## 852 2.8 Type *TwoPartIDType*

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

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

### 856 Schema Definition:

```
857 <xsd:simpleType name="TwoPartIDType">  
858   <xsd:restriction base="xsd:string">  
859     <xsd:minLength value="3"/>  
860     <xsd:maxLength value="41"/>  
861     <xsd:pattern value="[1-9][0-9]{0,19}-[1-9][0-9]{0,19}"/>  
862     <xsd:whiteSpace value="collapse"/>  
863   </xsd:restriction>  
864 </xsd:simpleType>
```

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

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

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

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

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

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

### 872 Example 3 – A minimum-length *TwoPartIDType* :

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

### 874 Example 4 – A maximum-length *TwoPartIDType* :

```
875 <ekmi:ApplicationID>  
876   18446744073709551615-18446744073709551615  
877 </ekmi:ApplicationID>
```

## 878 2.9 Element <KeyAlgorithm>

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

### 881 Schema Definition:

```
882 <xsd:simpleType name="EncryptionAlgorithmType">  
883   <xsd:restriction base="xsd:anyURI">  
884     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>  
885     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>  
886     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes192-cbc"/>  
887     <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>  
888   </xsd:restriction>  
889 </xsd:simpleType>
```

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

- 893 1. Triple Data Encryption Standard (3DES)  
 894  
 895 Within the context of this specification, and as specified in [XMLEncryption], the form of 3DES  
 896 supported within SKSML is a 192-bit key with a 64-bit Initialization Vector. Of the key bits, the first 64  
 897 are used in the first DES operation, the second 64 bits in the second (middle) DES operation, and the  
 898 third 64 bits in the third (last) DES operation. Each of these 64 bits of key contain 56 effective bits and  
 899 8 parity bits.  
 900  
 901 The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#tripleDES-cbc>.
- 902 2. Advanced Encryption Standard (AES) – 128-bit  
 903  
 904 Within the context of this specification, and as specified in [AES], this is a 128-bit symmetric key used in  
 905 the Cipher Block Chaining (CBC) mode.  
 906  
 907 The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#aes128-cbc>.
- 908 3. Advanced Encryption Standard (AES) – 192-bit  
 909  
 910 Within the context of this specification, and as specified in [AES], this is a 192-bit symmetric key used in  
 911 the Cipher Block Chaining (CBC) mode.  
 912  
 913 The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#aes192-cbc>.
- 914 4. Advanced Encryption Standard (AES) – 256-bit  
 915  
 916 Within the context of this specification, and as specified in [AES], this is a 256-bit symmetric key used in  
 917 the Cipher Block Chaining (CBC) mode.  
 918  
 919 The algorithm standard is denoted by the URL: <http://www.w3.org/2001/04/xmlenc#aes256-cbc>.

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

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

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

```
924 <ekmi:KeyUsePolicy>
925   ...
926   <ekmi:KeyAlgorithm>
927     http://www.w3.org/2001/04/xmlenc#tripleDES-cbc
928   </ekmi:KeyAlgorithm>
929   ...
930 </ekmi:KeyUsePolicy>
```

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

```
932 <ekmi:KeyUsePolicy>
933   ...
934   <ekmi:KeyAlgorithm>
935     http://www.w3.org/2001/04/xmlenc#aes128-cbc
936   </ekmi:KeyAlgorithm>
937   ...
938 </ekmi:KeyUsePolicy>
```

939 **2.10 Element <KeySize>**

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

942 **Schema Definition:**

```
943 <xsd:simpleType name="KeySizeType">
944   <xsd:restriction base="xsd:unsignedShort">
945     <xsd:totalDigits value="3"/>
946     <xsd:fractionDigits value="0"/>
947     <xsd:enumeration value="128"/>
948     <xsd:enumeration value="192"/>
949     <xsd:enumeration value="256"/>
950   </xsd:restriction>
951 </xsd:simpleType>
```

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

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

- 954 1. 128-bits when used with the **AES-192** algorithm
- 955 2. 192-bits when used with the **AES-192** or the **3DES** algorithms
- 956 3. 256-bits when used with the **AES-256** algorithm

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

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

```
960 <ekmi:KeyUsePolicy>
961   ...
962   <ekmi:KeySize>128</ekmi:KeySize>
963   ...
964 </ekmi:KeyUsePolicy>
```

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

```
966 <ekmi:KeyUsePolicy>
967   ...
968   <ekmi:KeySize>192</ekmi:KeySize>
969   ...
970 </ekmi:KeyUsePolicy>
```

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

```
972 <ekmi:KeyUsePolicy>
973   ...
974   <ekmi:KeySize>256</ekmi:KeySize>
975   ...
976 </ekmi:KeyUsePolicy>
```

## 977 **2.11 Element <Status>**

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

980 **Schema Definition:**

```
981 <xsd:simpleType name="StatusType">
982   <xsd:restriction base="xsd:string">
983     <xsd:enumeration value="Active"/>
984     <xsd:enumeration value="Default"/>
985     <xsd:enumeration value="Inactive"/>
986     <xsd:enumeration value="Other"/>
```

```
987     </xsd:restriction>
988 </xsd:simpleType>
```

989 Where it does exist, there SHALL be only one <Status> element within the enclosing element.

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

- 991 1. The **Active** value indicates that the element that makes up the document-root is currently active in the  
992 SKMS and conforming SKCL implementations may use it within applications.
- 993 2. The **Default** value indicates that the element that makes up the document root is the default element in  
994 the SKMS, is also active, and conforming SKCL implementations may use it within applications.
- 995 3. The **Inactive** value indicates that the element that makes up the document root is not active in the  
996 SKMS, and conforming SKCL implementations may NOT use it within applications.
- 997 4. The **Other** value indicates that the element that makes up the document root has a meaning that is  
998 application-specific. However, conforming SKCL implementations may NOT use it within applications.

999 Some examples of the <Status> element are shown below; other parts of their enclosing elements are not shown  
1000 for brevity:

1001 **Example 1 – An example with an Active status within a <KeyUsePolicy> element:**

```
1002     <ekmi:KeyUsePolicy>
1003         ...
1004         <ekmi:Status>Active</ekmi:Status>
1005         ...
1006     </ekmi:KeyUsePolicy>
```

1007 **Example 2 – An example with an Inactive status within a <KeyUsePolicy> element:**

```
1008     <ekmi:KeyUsePolicy>
1009         ...
1010         <ekmi:Status>Inactive</ekmi:Status>
1011         ...
1012     </ekmi:KeyUsePolicy>
```

1013 **Example 3 – An example with a Default status within a <KeyUsePolicy> element:**

```
1014     <ekmi:KeyUsePolicy>
1015         ...
1016         <ekmi:Status>Default</ekmi:Status>
1017         ...
1018     </ekmi:KeyUsePolicy>
```

## 1019 2.12 Element <Permissions>

1020 The <Permissions> element, of type **PermissionsType** is at the heart of the <KeyUsePolicy> element. It  
1021 provides guidance to conforming SKCL implementations on who may use the symmetric key, when they may use  
1022 it, for what purposes, for how long and in which locations. For applications that conform to the Multi-Level  
1023 Security (MLS) model, there is a provision for specifying which levels are permitted use fo the key. There is also  
1024 an element that allows for extending the <Permissions> element to accommodate rules that have not been  
1025 envisioned in the current specification.

1026 There SHALL be only one <Permissions> element within a <KeyUsePolicy> element.

1027 **Schema Definition:**

```
1028     <xsd:complexType name="PermissionsType">
1029         <xsd:sequence>
1030             <xsd:element
1031                 name="PermittedApplications"
```

```

1032         type="tns:PermittedApplicationsType"
1033         minOccurs="0"/>
1034     <xsd:element
1035         name="PermittedDates"
1036         type="tns:PermittedDatesType"
1037         minOccurs="0"/>
1038     <xsd:element
1039         name="PermittedDays"
1040         type="tns:PermittedDaysType"
1041         minOccurs="0"/>
1042     <xsd:element
1043         name="PermittedDuration"
1044         type="tns:PermittedDurationType"
1045         minOccurs="0"/>
1046     <xsd:element
1047         name="PermittedLevels"
1048         type="tns:PermittedLevelsType"
1049         minOccurs="0"/>
1050     <xsd:element
1051         name="PermittedLocations"
1052         type="tns:PermittedLocationsType"
1053         minOccurs="0"/>
1054     <xsd:element
1055         name="PermittedTimes"
1056         type="tns:PermittedTimesType"
1057         minOccurs="0"/>
1058     <xsd:element
1059         name="PermittedTransactions"
1060         type="tns:PermittedTransactionsType"
1061         minOccurs="0"/>
1062     <xsd:element
1063         name="PermittedUses"
1064         type="tns:PermittedUsesType"
1065         minOccurs="0"/>
1066     <xsd:element
1067         name="Other"
1068         type="xsd:anyType"
1069         minOccurs="0"/>
1070 </xsd:sequence>
1071 </xsd:complexType>

```

1072  
1073 The <Permissions> element consists of the following sub-elements:

- 1074 1. The optional <PermittedApplications> element identifies applications that are permitted the use of the  
1075 symmetric key in question.  
1076  
1077 The absence of the <PermittedApplications> element in a <Permissions> element implies that all  
1078 applications are permitted to use the key. Identifying a specific application restricts the use of the key to  
1079 only the identified applications.  
1080  
1081 The <PermittedApplications> element is specified in Section 2.13.
  
- 1082 2. The optional <PermittedDates> element identifies the dates during which applications are permitted the  
1083 use of the symmetric key in question.  
1084  
1085 The absence of the <PermittedDates> element in a <Permissions> element implies that applications are  
1086 permitted to use the key on any date. Identifying specific dates restricts the use of the key to only the  
1087 duration between the identified dates.  
1088  
1089 The <PermittedDates> element is specified in Section 2.12.



- 1090 3. The optional <PermittedDays> element identifies the days of week during which applications are  
1091 permitted the use of the symmetric key in question.  
1092  
1093 The absence of the <PermittedDays> element in a <Permissions> element implies that applications are  
1094 permitted to use the key on any day of the week. Identifying specific days restricts the use of the key to  
1095 only the identified days.  
1096  
1097 The <PermittedDays> element is specified in Section 2.15.
- 1098 4. The optional <PermittedDuration> element identifies the duration (in seconds) in which applications are  
1099 permitted the use of the symmetric key in question, once the SKCL starts using the symmetric key.  
1100  
1101 The absence of the <PermittedDuration> element in a <Permissions> element implies that applications  
1102 are permitted to use the key for any duration after it has been used. Identifying the specific duration  
1103 restricts the use of the key to only the period after the start of the use of the key.  
1104  
1105 A distinction between <PermittedDates> and <PermittedDuration> is that the former has fixed start and  
1106 end-dates for the use of the key, whereas the latter has a fixed end-date-and-time after the key has  
1107 begun to be used without a fixed start-date-and-time. Thus, an application with a <PermittedDuration>  
1108 can begin the use of a symmetric key at any time, but must complete its use at the end of the  
1109 <PermittedDuration> once it has begun. With <PermittedDates>, an application can continue using the  
1110 symmetric key until the fixed date-and-time have been reached.  
1111  
1112 The <PermittedDuration> element is specified in Section 2.x16
- 1113 5. Within a Multi-Level Security (MLS) system, the optional <PermittedLevels> element identifies the levels  
1114 at which applications are permitted the use of the symmetric key in question.  
1115  
1116 The absence of the <PermittedLevels> element in a <Permissions> element implies that applications are  
1117 permitted to use the key on any level of security. Identifying specific level(s) restricts the use of the key  
1118 to only the identified level(s).  
1119  
1120 The <PermittedLevels> element is specified in Section 2.x17
- 1121 6. The optional <PermittedLocations> element identifies physical geographic locations where applications  
1122 are permitted the use of the symmetric key in question.  
1123  
1124 The absence of the <PermittedLocations> element in a <Permissions> element implies that applications  
1125 are permitted to use the key at any physical location. Identifying specific locations restricts the use of  
1126 the key to only the identified locations.  
1127  
1128 The <PermittedLocations> element is specified in Section 2.18.
- 1129 7. The optional <PermittedTimes> element identifies the times of day during which applications are  
1130 permitted the use of the symmetric key in question.  
1131  
1132 The absence of the <PermittedTimes> element in a <Permissions> element implies that applications are  
1133 permitted to use the key at any time of day. Identifying specific times restricts the use of the key to only  
1134 the duration of the identified times.  
1135  
1136 The <PermittedTimes> element is specified in Section 2.19.
- 1137 8. The optional <PermittedTransactions> element identifies the number of encryption transactions that  
1138 applications are permitted, with the use of the symmetric key in question.  
1139  
1140 The absence of the <PermittedTransactions> element in a <Permissions> element implies that  
1141 applications are permitted to use the key for as many encryption transactions as necessary. Identifying  
1142 a specific number of transactions restricts the use of the key to only the limit identified in the element.  
1143  
1144 The <PermittedTransactions> element is specified in Section 2.20.

1145 9. The optional <PermittedUses> element identifies the uses which applications are permitted with the  
1146 symmetric key in question.

1147  
1148 The absence of the <PermittedUses> element in a <Permissions> element implies that applications are  
1149 permitted to use the key for any purpose. Identifying specific uses restricts the use of the key to only  
1150 the identified uses.

1151  
1152 The <PermittedUses> element is specified in Section 2.21.

1153 10. The optional <Other> element allows implementers to specify permissions that cannot be addressed  
1154 with the above-mentioned categories, for restricting the use of the symmetric key in question.

1155  
1156 While the <Other> element provides flexibility for implementations, the disadvantage of the element is  
1157 that it may render a specific implementation incompatible with the SKSML standard. It is strongly  
1158 recommended that implementers avoid the use of the <Other> element unless they definitely do not  
1159 expect to inter-operate with other SKCL implementations. If there is a strong need for capability that  
1160 does not exist within the current <Permissions> element, implementers are encouraged to contact the  
1161 OASIS EKMI TC and raise their concerns.

1162 When the <Permissions> element is empty, that there are no restrictions on the use of the symmetric key other  
1163 than that the application calling on the SKCL be authorized to access the key in question. However, when there  
1164 are elements defined within the <Permissions> element, conforming SKCL implementations must comply with all  
1165 the permission elements, evaluating the most restrictive permissions first and in decreasing order of restriction,  
1166 before allowing the use of the key.

1167 For example, if a <Permissions> element specifies that a key may be used on Weekdays, between the hours of  
1168 0900 and 1700 Hours, then a request for a symmetric key on a Saturday at 1105 would deny use of the key in  
1169 question, since it violates the more restrictive permission of being allowed for use only on weekdays.

1170 In another example, if a <Permissions> element specifies a <PermittedDuration> of 600 seconds from the start of  
1171 use of the key, and a <PermittedTransactions> of 10 encryption transactions, conforming SKCL implementations  
1172 must evaluate both permissions before each transaction and determine if they are both within the specified  
1173 thresholds before using the key. If the 600 seconds expire before the 10 encryption transactions have been  
1174 completed, or if the 10 encryption transactions are completed before 600 seconds have expired, conforming  
1175 SKCL implementations MUST not use the key in question anymore.

1176 Some examples of the <Permissions> element are as follows; the enclosing <KeyUsePolicy> element, <Symkey>  
1177 element and <SymkeyResponse> elements are not displayed for brevity:

1178 **Example 1 – a collection of permissions that permits a single application the use of the symmetric key in**  
1179 **question, between January 01, 2008 and December 31, 2008 and between the hours of 0700 and 1900.**

```
1180 <ekmi:Permissions>
1181   <ekmi:PermittedApplications>
1182     <ekmi:PermittedApplication>
1183       <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
1184       <ekmi:ApplicationName>Payroll Application</ekmi:ApplicationName>
1185       <ekmi:Version>1.0</ekmi:Version>
1186       <ekmi:DigestAlgorithm>http://www.w3.org/2000/09/xmlsig#sha1</ekmi:DigestAlgorithm>
1187       <ekmi:DigestValue>NIG4bKkt4cziEqFFuOoBTM81efU=</ekmi:DigestValue>
1188     </ekmi:PermittedApplication>
1189   </ekmi:PermittedApplications>
1190   <ekmi:PermittedDates>
1191     <ekmi:PermittedDate>
1192       <ekmi:StartDate>2008-01-01</ekmi:StartDate>
1193       <ekmi:EndDate>2008-12-31</ekmi:EndDate>
1194     </ekmi:PermittedDate>
1195   </ekmi:PermittedDates>
1196   <ekmi:PermittedTimes>
1197     <ekmi:PermittedTime>
1198       <ekmi:StartTime>07:00:00</ekmi:StartTime>
1199       <ekmi:EndTime>19:00:00</ekmi:EndTime>
```

```
1200     </ekmi:PermittedTime>
1201     </ekmi:PermittedTimes>
1202 </ekmi:Permissions>
```

1203 **Example 2 – a collection of permissions that permits two specific applications the use of the symmetric**  
1204 **key in question, between January 01, 2009 and January 31, 2009.**

```
1205     <ekmi:Permissions>
1206         <ekmi:PermittedApplications>
1207             <ekmi:PermittedApplication>
1208                 <ekmi:ApplicationID>10514-24</ekmi:ApplicationID>
1209                 <ekmi:ApplicationName>Employee Tax Reporting Application</ekmi:ApplicationName>
1210                 <ekmi:Version>3.3</ekmi:Version>
1211                 <ekmi:DigestAlgorithm>http://www.w3.org/2000/09/xmldsig#sha1</ekmi:DigestAlgorithm>
1212                 <ekmi:DigestValue>G4bsdfKkt4cziEqFFuOoBTM81efU=</ekmi:DigestValue>
1213             </ekmi:PermittedApplication>
1214             <ekmi:PermittedApplication>
1215                 <ekmi:ApplicationID>10514-25</ekmi:ApplicationID>
1216                 <ekmi:ApplicationName>IRS Tax Reporting Application</ekmi:ApplicationName>
1217                 <ekmi:Version>2.1</ekmi:Version>
1218                 <ekmi:DigestAlgorithm>http://www.w3.org/2000/09/xmldsig#sha1</ekmi:DigestAlgorithm>
1219                 <ekmi:DigestValue>4uyb4sd234cziqzlmnuOoBTMa08=</ekmi:DigestValue>
1220             </ekmi:PermittedApplication>
1221         </ekmi:PermittedApplications>
1222         <ekmi:PermittedDates>
1223             <ekmi:PermittedDate>
1224                 <ekmi:StartDate>2009-01-01</ekmi:StartDate>
1225                 <ekmi:EndDate>2009-01-31</ekmi:EndDate>
1226             </ekmi:PermittedDate>
1227         </ekmi:PermittedDates>
1228     </ekmi:Permissions>
```

1229 **Example 3 – a collection of permissions that permits all applications the use of the symmetric key in**  
1230 **question, for 100 transactions for encrypting/decrypting credit card numbers.**

```
1231     <ekmi:Permissions>
1232         <ekmi:PermittedTransactions>100</ekmi:PermittedTransactions>
1233         <ekmi:PermittedUses>
1234             <ekmi:PermittedUse>CCN</ekmi:PermittedUse>
1235         </ekmi:PermittedUses>
1236     </ekmi:Permissions>
```

1237 **Example 4 – a collection of permissions that permits all applications the use of the symmetric key in**  
1238 **question, for 600 seconds once the SKCL starts using the key.**

```
1239     <ekmi:Permissions>
1240         <ekmi:PermittedDuration>600</ekmi:PermittedDuration>
1241     </ekmi:Permissions>
```

1242 **Example 5 – a collection of permissions that permits a specific application the use of the symmetric key**  
1243 **in question, at specific geographic locations on weekdays between the hours of 0800 and 1700, and only**  
1244 **when the application is operating at the Secret level in an MLS system.**

```
1245     <ekmi:Permissions>
1246         <ekmi:PermittedDays>
1247             <ekmi:PermittedDay>Weekday</ekmi:PermittedDay>
1248         </ekmi:PermittedDays>
1249         <ekmi:PermittedLocations>
1250             <ekmi:PermittedLocation>
1251                 <ekmi:LocationName>Facility A51</ekmi:LocationName>
1252                 <ekmi:Latitude>37.385562</ekmi:Latitude>
1253                 <ekmi:Longitude>-121.993387</ekmi:Longitude>

```

```

1254         </ekmi:PermittedLocation>
1255     </ekmi:PermittedLocation>
1256         <ekmi:LocationName>Facility DC-VA01</ekmi:LocationName>
1257         <ekmi:Latitude>88.485362</ekmi:Latitude>
1258         <ekmi:Longitude>-21.453648</ekmi:Latitude>
1259     </ekmi:PermittedLocation>
1260 </ekmi:PermittedLocations>
1261 <ekmi:PermittedLevels>
1262     <ekmi:PermittedLevel>Secret</ekmi:PermittedLevel>
1263 </ekmi:PermittedLevels>
1264 <ekmi:PermittedTimes>
1265     <ekmi:PermittedTime>
1266         <ekmi:StartTime>07:00:00</ekmi:StartTime>
1267         <ekmi:EndTime>19:00:00</ekmi:EndTime>
1268     </ekmi:PermittedTime>
1269 </ekmi:PermittedTimes>
1270 </ekmi:Permissions>

```

## 1271 2.13 Element <PermittedApplications> and <PermittedApplication>

1272 The element <PermittedApplications> , of type *PermittedApplicationsType* and its only child-element  
1273 <PermittedApplication> of type *ApplicationsType* are used to define the list of applications that are permitted to  
1274 use a symmetric key within a specific <Symkey> element.

### 1275 Schema Definition:

```

1276     <xsd:complexType name="PermittedApplicationsType">
1277         <xsd:sequence>
1278             <xsd:element
1279                 name="PermittedApplication"
1280                 type="tns:ApplicationsType"
1281                 minOccurs="0"
1282                 maxOccurs="unbounded" />
1283         </xsd:sequence>
1284     </xsd:complexType>

```

### 1285 Schema Definition:

```

1286     <xsd:complexType name="ApplicationsType">
1287         <xsd:sequence>
1288             <xsd:element name="ApplicationID" type="tns:TwoPartIDType" />
1289             <xsd:element name="ApplicationName">
1290                 <xsd:simpleType>
1291                     <xsd:restriction base="xsd:string">
1292                         <xsd:maxLength value="256" />
1293                         <xsd:whiteSpace value="preserve" />
1294                     </xsd:restriction>
1295                 </xsd:simpleType>
1296             </xsd:element>
1297             <xsd:element name="Version" minOccurs="0">
1298                 <xsd:simpleType>
1299                     <xsd:restriction base="xsd:string">
1300                         <xsd:maxLength value="32" />
1301                         <xsd:whiteSpace value="preserve" />
1302                     </xsd:restriction>
1303                 </xsd:simpleType>
1304             </xsd:element>
1305             <xsd:group ref="tns:MessageDigestGroup" minOccurs="0" />
1306             <xsd:element name="Other" type="xsd:anyType" minOccurs="0" />
1307         </xsd:sequence>

```

```

1308     </xsd:sequence>
1309 </xsd:complexType>
1310 Schema Definition:
1311
1312 <xsd:group name="MessageDigestGroup">
1313   <xsd:sequence>
1314     <xsd:element name="DigestAlgorithm">
1315       <xsd:simpleType>
1316         <xsd:restriction base="xsd:anyURI">
1317           <xsd:enumeration value="http://www.w3.org/2000/09/xmlsig#sha1"/>
1318           <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#sha256"/>
1319           <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#sha512"/>
1320         </xsd:restriction>
1321       </xsd:simpleType>
1322     </xsd:element>
1323     <xsd:element name="DigestValue">
1324       <xsd:simpleType>
1325         <xsd:restriction base="xsd:base64Binary">
1326           <xsd:maxLength value="1024"/>
1327         </xsd:restriction>
1328       </xsd:simpleType>
1329     </xsd:element>
1330   </xsd:sequence>
1331 </xsd:group>

```

1332 When it is used within a <KeyUsePolicy> element, there SHALL be only one <PermittedApplications> element  
1333 within the <KeyUsePolicy> element. However, there MAY be an unbounded number of <PermittedApplication>  
1334 elements within a <PermittedApplications> element.

1335 NOTE: It is noteworthy to mention that the absence of a <PermittedApplications> element within a  
1336 <KeyUsePolicy> element specifies that ALL applications are permitted the use of the symmetric key, subject to  
1337 complying with all other permission clauses in the <KeyUsePolicy> element, if any.

1338 The <PermittedApplication> element provides details of the application that is permitted to use the symmetric  
1339 key in question. The <PermittedApplication> element consists of the following sub-elements:

- 1340 1. The <ApplicationID> element identifies the unique identifier assigned to this application within the  
1341 SKMS. It is a **TwoPartIDType** as specified in Section 2.8. There SHALL be only one <ApplicationID>  
1342 element within a <PermittedApplication> element.
- 1343 2. The <ApplicationName> element identifies the name assigned to this application within the SKMS. It is  
1344 an XSD **String** with a maximum length of 256 characters. There SHALL be only one  
1345 <ApplicationName> element within a <PermittedApplication> element.
- 1346 3. An optional <Version> element identifying the version number of this application within the SKMS. It is  
1347 an XSD **String** with a maximum length of 32 characters. There MAY be only one <Version> element  
1348 within a <PermittedApplication> element.
- 1349 4. The <MessageDigestGroup> group which identifies the message digest of the application's binary  
1350 image, along with the message digest algorithm used to calculate the digest value. The  
1351 <MessageDigestGroup> consists of the following elements:
  - 1352 a) The <DigestAlgorithm> element of the XSD type **anyURI**, which supports one of the following  
1353 three digest algorithms:
    - 1354 i. <http://www.w3.org/2000/09/xmlsig#sha1>
    - 1355 ii. <http://www.w3.org/2001/04/xmlenc#sha256>
    - 1356 iii. <http://www.w3.org/2001/04/xmlenc#sha512>
  - 1357 b) The <DigestValue> element of the XSD type **base64Binary**.

1358 There SHALL be only one <MessageDigestGroup> group within a <PermittedApplication> element.

1359 5. An optional<Other> element that provides implementers the ability to carry other information about the  
1360 application that may be relevant to their SKMS. Implementers are cautioned that the use of the  
1361 <Other> element may not be supported by other SKCL implementations , and may break interoperability  
1362 of the SKSML protocol. Should there be a strong need for additional features in the  
1363 <PermittedApplication> element, implementers are encouraged to contact the OASIS EKMI TC with their  
1364 requirements.

1365 NOTE: The SKSML specification does not specify how an SKCL implementation will determine the message  
1366 digest of application that needs to use the symmetric key in question. It is left to the implementers of the SKCL  
1367 how they determine the message digest of the calling application, with the specified algorithm, before it permits  
1368 the application the use of the symmetric key.

1369 Some examples of the <PermittedApplications> element are shown below; other parts of their enclosing  
1370 elements are not shown for brevity:

1371 **Example 1 – An example of a <PermittedApplications> element with two <PermittedApplication>**  
1372 **elements: of specific versions and message digest values:**

```
1373 <ekmi:PermittedApplications>
1374   <ekmi:PermittedApplication>
1375     <ekmi:ApplicationID>10514-24</ekmi:ApplicationID>
1376     <ekmi:ApplicationName>Employee Tax Reporting</ekmi:ApplicationName>
1377     <ekmi:Version>3.3</ekmi:Version>
1378     <ekmi:DigestAlgorithm>
1379       http://www.w3.org/2000/09/xmlsig#sha1
1380     </ekmi:DigestAlgorithm>
1381     <ekmi:DigestValue>G4bsdfKkt4cziEqFFu0oBTM81efU=</ekmi:DigestValue>
1382   </ekmi:PermittedApplication>
1383   <ekmi:PermittedApplication>
1384     <ekmi:ApplicationID>10514-25</ekmi:ApplicationID>
1385     <ekmi:ApplicationName>IRS Tax Reporting Application</ekmi:ApplicationName>
1386     <ekmi:Version>2.1</ekmi:Version>
1387     <ekmi:DigestAlgorithm>
1388       http://www.w3.org/2001/04/xmlenc#sha256
1389     </ekmi:DigestAlgorithm>
1390     <ekmi:DigestValue>
1391       ab7b85c9410d48c54fc7939c391be4028e7305085191c56e7b3740f2cbdbbc79
1392     </ekmi:DigestValue>
1393   </ekmi:PermittedApplication>
1394 </ekmi:PermittedApplications>
```

1395 **Example 1 – An example of a <PermittedApplications> element with one <PermittedApplication> element**  
1396 **that applies to all versions of the application; the message digest value and algorithm are not used in**  
1397 **this example:**

```
1398 <ekmi:PermittedApplications>
1399   <ekmi:PermittedApplication>
1400     <ekmi:ApplicationID>10514-14</ekmi:ApplicationID>
1401     <ekmi:ApplicationName>E-Commerce Payment</ekmi:ApplicationName>
1402   </ekmi:PermittedApplication>
1403 </ekmi:PermittedApplications>
```

## 1404 2.14 Element <PermittedDates> and <PermittedDate>

1405 The element <PermittedDates> , of type *PermittedDatesType* and its only child-element <PermittedDate> ,  
1406 which is an anonymous XSD *ComplexType*, are used to define sets of dates between which applications are  
1407 permitted to use a symmetric key within a specific <Symkey> element.

1408 **Schema Definition:**

```

1409 <xsd:complexType name="PermittedDatesType">
1410   <xsd:sequence>
1411     <xsd:element name="PermittedDate" minOccurs="0" maxOccurs="unbounded">
1412       <xsd:complexType>
1413         <xsd:sequence>
1414           <xsd:element name="StartDate">
1415             <xsd:simpleType>
1416               <xsd:restriction base="xsd:date">
1417                 <xsd:pattern value="\p{Nd}{4}-\p{Nd}{2}-\p{Nd}{2}"/>
1418               </xsd:restriction>
1419             </xsd:simpleType>
1420           </xsd:element>
1421           <xsd:element name="EndDate">
1422             <xsd:simpleType>
1423               <xsd:restriction base="xsd:date">
1424                 <xsd:pattern value="\p{Nd}{4}-\p{Nd}{2}-\p{Nd}{2}"/>
1425               </xsd:restriction>
1426             </xsd:simpleType>
1427           </xsd:element>
1428         </xsd:sequence>
1429       </xsd:complexType>
1430     </xsd:element>
1431   </xsd:sequence>
1432 </xsd:complexType>

```

1433 When it is used within a <KeyUsePolicy> element, there SHALL be only one <PermittedDates> element within  
1434 the <KeyUsePolicy> element. However, there MAY be an unbounded number of <PermittedDate> elements  
1435 within a <PermittedDates> element.

1436 NOTE: It is noteworthy to mention that the absence of a <PermittedDates> element within a <KeyUsePolicy>  
1437 element specifies that applications are permitted the use of the symmetric key on any calendar date of the year,  
1438 subject to complying with all other permission clauses in the <KeyUsePolicy> element, if any.

1439 The <PermittedDate> element identifies an individual set of dates between which application are permitted to use  
1440 the symmetric key in question. The <PermittedDate> element consists of the following sub-elements:

- 1441 1. The <StartDate> element identifies the date from which applications may start using the symmetric key  
1442 in question. It is an XSD **Date** type that MUST be specified in a specific pattern (see examples) where  
1443 the first four digits specify the year, the second two digits specify the calendar month in the year, and  
1444 the last two digits specify the calendar date of the month.

1445  
1446 There SHALL be only one <StartDate> element within a <PermittedDate> element.

1447  
1448 Conforming SKCL implementations SHALL NOT start using the symmetric before the onset of the  
1449 <StartDate> on the client machine. Unless further constrained by the <PermittedTimes> element, the  
1450 onset of the <StartDate> is specified to be the first second of the day – 00:00:01 Hours – on the client  
1451 machine.

- 1452 2. The <EndDate> element identifies the date until which applications may use the symmetric key in  
1453 question. It is an XSD **Date** type that MUST be specified in a specific pattern (see examples) where the  
1454 first four digits specify the year, the second two digits specify the calendar month in the year, and the  
1455 last two digits specify the calendar date of the month.

1456  
1457 There SHALL be only one <EndDate> element within a <PermittedDate> element.

1458  
1459 Conforming SKCL implementations SHALL NOT use the symmetric after the end of the <EndDate> on  
1460 the client machine. Unless further constrained by the <PermittedTimes> element, the end of the  
1461 <EndDate> is specified to be the last second of the day – 23:59:59 Hours – on the client machine.

1462 Some examples of the <PermittedDates> element are shown below; other parts of their enclosing elements are  
1463 not shown for brevity:

1464 **Example 1 – An example of a <PermittedDates> element with a single<PermittedDate> element. The**  
1465 **<StartDate> specifies January 01, 2009 while the <EndDate> specifies January 31, 2009:**

```
1466     <ekmi:PermittedDates>
1467         <ekmi:PermittedDate>
1468             <ekmi:StartDate>2009-01-01</ekmi:StartDate>
1469             <ekmi:EndDate>2009-01-31</ekmi:EndDate>
1470         </ekmi:PermittedDate>
1471     </ekmi:PermittedDates>
```

1472 **Example 2 – An example of a <PermittedDates> element with two <PermittedDate> elements. For the first**  
1473 **<PermittedDate> element , the <StartDate> element specifies July 01, 2008 while the <EndDate> element**  
1474 **specifies July 03, 2008. For the second <PermittedDate> element, the <StartDate> element specifies July**  
1475 **07, 2008 while the <EndDate> element specifies July 12, 2008. This policy might imply that a symmetric**  
1476 **key with this <PermittedDates> element cannot be used on the July 4<sup>th</sup> weekend of 2008:**

```
1477     <ekmi:PermittedDates>
1478         <ekmi:PermittedDate>
1479             <ekmi:StartDate>2008-07-01</ekmi:StartDate>
1480             <ekmi:EndDate>2008-07-03</ekmi:EndDate>
1481         </ekmi:PermittedDate>
1482         <ekmi:PermittedDate>
1483             <ekmi:StartDate>2008-07-07</ekmi:StartDate>
1484             <ekmi:EndDate>2009-07-12</ekmi:EndDate>
1485         </ekmi:PermittedDate>
1486     </ekmi:PermittedDates>
```

## 1487 **2.15 Element <PermittedDays> and <PermittedDay>**

1488 The element <PermittedDays> , of type *PermittedDaysType* and its only child-element <PermittedDay> of the  
1489 *<PermittedDayType>*, are used to define days of the week when applications are permitted to use a symmetric  
1490 key within a specific <Symkey> element.

### 1491 **Schema Definition:**

```
1492     <xsd:complexType name="PermittedDaysType">
1493         <xsd:sequence>
1494             <xsd:element
1495                 name="PermittedDay"
1496                 type="tns:PermittedDayType"
1497                 minOccurs="0"
1498                 maxOccurs="unbounded">
1499             </xsd:element>
1500         </xsd:sequence>
1501     </xsd:complexType>
```

### 1502 **Schema Definition:**

```
1503     <xsd:simpleType name="PermittedDayType">
1504         <xsd:restriction base="xsd:string">
1505             <xsd:enumeration value="Sunday"/>
1506             <xsd:enumeration value="Monday"/>
1507             <xsd:enumeration value="Tuesday"/>
1508             <xsd:enumeration value="Wednesday"/>
1509             <xsd:enumeration value="Thursday"/>
1510             <xsd:enumeration value="Friday"/>
1511             <xsd:enumeration value="Saturday"/>
1512             <xsd:enumeration value="Weekday"/>
1513             <xsd:enumeration value="Weekend"/>
1514         </xsd:restriction>
1515     </xsd:simpleType>
```



1516 When it is used within a <KeyUsePolicy> element, there SHALL be only one <PermittedDay> element within the  
1517 <KeyUsePolicy> element. However, there MAY be an unbounded number of <PermittedDay> elements within a  
1518 <PermittedDays> element.

1519 NOTE: It is noteworthy to mention that the absence of a <PermittedDays> element within a <KeyUsePolicy>  
1520 element specifies that applications are permitted the use of the symmetric key on all days of the week, subject to  
1521 complying with all other permission clauses in the <KeyUsePolicy> element, if any.

1522 The <PermittedDay> element, of the XSD *String* type, identifies individual days of the week – from an  
1523 enumerated list - on which application are permitted to use the symmetric key in question.

1524 Some examples of the <PermittedDays> element are shown below; other parts of their enclosing elements are  
1525 not shown for brevity:

1526 **Example 1 – An example of a <PermittedDays> element with a single<PermittedDay> element, specifying**  
1527 **that the symmetric key may be used only on weekdays:**

```
1528     <ekmi:PermittedDays>  
1529         <ekmi:PermittedDay>Weekday</ekmi:PermittedDay>  
1530     </ekmi:PermittedDays>
```

1531 **Example 2 – An example of a <PermittedDays> element with three <PermittedDay> elements, specifying**  
1532 **that the symmetric key may be used only on Mondays, Wednesdays and Fridays:**

```
1533     <ekmi:PermittedDays>  
1534         <ekmi:PermittedDay>Monday</ekmi:PermittedDay>  
1535         <ekmi:PermittedDay>Wednesday</ekmi:PermittedDay>  
1536         <ekmi:PermittedDay>Friday</ekmi:PermittedDay>  
1537     </ekmi:PermittedDays>
```

## 1538 2.16 Element <PermittedDuration>

1539 The element <PermittedDuration> , of type *PermittedDurationType* is used to define the number of seconds  
1540 during which applications are permitted to use a symmetric key within a specific <Symkey> element, once the  
1541 SKCL has started using the symmetric key in question.

### 1542 Schema Definition:

```
1543     <xsd:simpleType name="PermittedDurationType">  
1544         <xsd:restriction base="xsd:positiveInteger">  
1545             <xsd:maxInclusive value="18446744073709551615"/>  
1546         </xsd:restriction>  
1547     </xsd:simpleType>
```

1548 When it is used within a <KeyUsePolicy> element, there SHALL be only one <PermittedDuration> element within  
1549 the <KeyUsePolicy> element.

1550 NOTE: It is noteworthy to mention that the absence of a <PermittedDuration> element within a <KeyUsePolicy>  
1551 element specifies that applications are permitted the use of the symmetric key forever, subject to complying with  
1552 all other permission clauses in the <KeyUsePolicy> element, if any.

1553 The <PermittedDuration> element, of the XSD *positiveInteger* type, identifies the precise number of seconds for  
1554 which the symmetric key in question may be used ONCE the key has been used by conforming SKCL  
1555 implementations for the first time.

1556 However, as long as the symmetric has not been used by an SKCL on a client device – it might be cached for  
1557 many days/weeks/months depending on the <KeyCachePolicy> in effect within the SKMS for that device – the  
1558 effective lifetime of the symmetric key may be well past the number of seconds specified in <PermittedDuration>  
1559 when calculated from the time of the key's generation time on the SKS server. It is the responsibility of the SKCL  
1560 implementation , when presented with a <PermittedDuration> element in a <KeyUsePolicy> of a symmetric key,  
1561 to keep track of the date/time when the symmetric key in question was first used on the client device, and how  
1562 long the key will last after that.

1563 Some examples of the <PermittedDuration> element are shown below; other parts of their enclosing elements  
1564 are not shown for brevity:

1565 **Example 1 – An example of a <PermittedDuration> element specifying that the symmetric key may be**  
1566 **used only for a single 24-hour period from the moment it is first used by an SKCL:**

```
1567 <ekmi:PermittedDuration>86400</ekmi:PermittedDuration>
```

1568 **Example 2 – An example of a <PermittedDuration> element specifying that the symmetric key may be**  
1569 **used only for week from the moment it is first used by an SKCL:**

```
1570 <ekmi:PermittedDuration>604800</ekmi:PermittedDuration>
```

1571 **Example 3 – An example of a <PermittedDuration> element specifying that the symmetric key may be**  
1572 **used only 5 minutes from the moment it is first used by an SKCL:**

```
1573 <ekmi:PermittedDuration>300</ekmi:PermittedDuration>
```

## 1574 2.17 Element <PermittedLevels> and <PermittedLevel>

1575 The element <PermittedLevels>, of type *LevelClassificationType*, is used to define the security level at which  
1576 applications are permitted to use a symmetric key within a specific <Symkey> element. This element is useful  
1577 only to applications and systems that conform to the multi-level security (MLS) system as defined in the Bell-  
1578 LaPadula model.

### 1579 Schema Definition:

```
1580 <xsd:complexType name="PermittedLevelsType">  
1581 <xsd:sequence>  
1582 <xsd:element  
1583 name="PermittedLevel"  
1584 type="tns:LevelClassificationType"  
1585 minOccurs="0"  
1586 maxOccurs="unbounded">  
1587 </xsd:element>  
1588 <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>  
1589 </xsd:sequence>  
1590 </xsd:complexType>
```

### 1591 Schema Definition:

```
1592 <xsd:simpleType name="LevelClassificationType">  
1593 <xsd:restriction base="xsd:string">  
1594 <xsd:enumeration value="Unclassified"/>  
1595 <xsd:enumeration value="Confidential"/>  
1596 <xsd:enumeration value="Secret"/>  
1597 <xsd:enumeration value="Top-Secret"/>  
1598 </xsd:restriction>  
1599 </xsd:simpleType>
```

1600 When it is used within a <KeyUsePolicy> element, there SHALL be only one <PermittedLevels> element within  
1601 the <KeyUsePolicy> element. However, there MAY be an unbounded number of <PermittedLevel> elements  
1602 within the <PermittedLevels> element.

1603 NOTE: It is noteworthy to mention that the absence of a <PermittedLevels> element within a <KeyUsePolicy>  
1604 element specifies that applications at ALL levels are permitted the use of the symmetric key, subject to  
1605 complying with all other permission clauses in the <KeyUsePolicy> element, if any.

1606 The <PermittedLevel> element, of the *LevelClassificationType*, identifies the precise MLS level at which the  
1607 symmetric key in question may be used. The <PermittedLevel> SHALL contain one of the following four (4)  
1608 enumerated values:

- 1609 1. Unclassified
- 1610 2. Confidential
- 1611 3. Secret
- 1612 4. Top-Secret

1613 Some examples of the <PermittedDuration> element are shown below; other parts of their enclosing elements  
1614 are not shown for brevity:

1615 **Example 1 – An example of a <PermittedLevels> element specifying that the symmetric key may be used**  
1616 **only by applications at the Confidential level:**

```
1617 <ekmi:PermittedLevels>
1618   <ekmi:PermittedLevel>Confidential</ekmi:PermittedLevel>
1619 </ekmi:PermittedLevels>
```

1620 **Example 2 – An example of a <PermittedLevels> element specifying that the symmetric key may be used**  
1621 **only by applications at the Secret or Top-Secret level:**

```
1622 <ekmi:PermittedLevels>
1623   <ekmi:PermittedLevel>Secret</ekmi:PermittedLevel>
1624   <ekmi:PermittedLevel>Top-Secret</ekmi:PermittedLevel>
1625 </ekmi:PermittedLevels>
```

## 1626 2.18 Element <PermittedLocations> and <PermittedLocation>

1627 The element <PermittedLocations>, of type *PermittedLocationsType*, is used to define the geographically  
1628 physical locations where applications are permitted to use a symmetric key within a specific <Symkey> element.  
1629 This element is useful only to applications and systems that have the ability to determine their Global Positioning  
1630 System (GPS) location of the client device using the symmetric key.

### 1631 Schema Definition:

```
1632 <xsd:complexType name="PermittedLocationsType">
1633   <xsd:sequence>
1634     <xsd:element name="PermittedLocation" minOccurs="1" maxOccurs="unbounded">
1635       <xsd:complexType>
1636         <xsd:sequence>
1637           <xsd:element name="LocationName">
1638             <xsd:simpleType>
1639               <xsd:restriction base="xsd:string">
1640                 <xsd:maxLength value="256"/>
1641                 <xsd:whiteSpace value="preserve"/>
1642               </xsd:restriction>
1643             </xsd:simpleType>
1644           </xsd:element>
1645           <xsd:group
1646             ref="tns:LocationCoordinateGroup"
1647             minOccurs="0"
1648             maxOccurs="unbounded"/>
1649           <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
1650         </xsd:sequence>
1651       </xsd:complexType>
1652     </xsd:element>
1653   </xsd:sequence>
1654 </xsd:complexType>
```

### 1655 Schema Definition:

```

1656 <xsd:group name="LocationCoordinateGroup">
1657   <xsd:sequence>
1658     <xsd:element name="Latitude">
1659       <xsd:simpleType>
1660         <xsd:restriction base="xsd:decimal">
1661           <xsd:totalDigits value="10"/>
1662           <xsd:fractionDigits value="7"/>
1663         </xsd:restriction>
1664       </xsd:simpleType>
1665     </xsd:element>
1666     <xsd:element name="Longitude">
1667       <xsd:simpleType>
1668         <xsd:restriction base="xsd:decimal">
1669           <xsd:totalDigits value="10"/>
1670           <xsd:fractionDigits value="7"/>
1671         </xsd:restriction>
1672       </xsd:simpleType>
1673     </xsd:element>
1674   </xsd:sequence>
1675 </xsd:group>

```

1676 When it is used within a <KeyUsePolicy> element, there SHALL be only one <PermittedLocations> element  
1677 within the <KeyUsePolicy> element. However, there MAY be an unbounded number of <PermittedLocation>  
1678 elements within the <PermittedLocations> element.

1679 NOTE: It is noteworthy to mention that the absence of a <PermittedLocations> element within a  
1680 <KeyUsePolicy> element specifies that applications are permitted the use of the symmetric key at ANY physical  
1681 location, subject to complying with all other permission clauses in the <KeyUsePolicy> element, if any.

1682 The <PermittedLocation> element, of the **PermittedLocationType**, identifies the precise geographical location  
1683 where the symmetric key in question may be used. The <PermittedLocation> SHALL contain the following  
1684 elements:

1685 1. The <LocationName> element identifies a human-readable name of the physical location. It is an XSD  
1686 **String** type element, with a maximum length of 256 characters.

1687 There SHALL be only one <LocationName> element within a <PermittedLocation> element.  
1688

1689 2. An optional **LocationCoordinateGroup** which, when present, SHALL contain the following two  
1690 elements:

1691 a) The <Latitude> element of XSD **Decimal** type, that identifies the horizontal coordinate location  
1692 of the client device on the Earth, measured in *degrees* and expressed as a decimal with the  
1693 *minutes* and *seconds* part of the measurement expressed as a single fraction.

1694 When used, there SHALL be only one <Latitude> element within the <PermittedLocation>  
1695 element.  
1696

1697 b) The <Longitude> element of XSD **Decimal** type, that identifies the vertical coordinate location  
1698 of the client device on the Earth, measured in *degrees* and expressed as a decimal with the  
1699 *minutes* and *seconds* part of the measurement expressed as a single fraction.

1700 When used, there SHALL be only one <Longitude> element within the <PermittedLocation>  
1701 element.  
1702

1703 Some examples of the <PermittedLocations> element are shown below; other parts of their enclosing elements  
1704 are not shown for brevity:

1705 **Example 1 – An example of a <PermittedLocations> element specifying that the symmetric key may be**  
1706 **used only by applications at a single named location:**

```

1707     <ekmi:PermittedLocations>
1708         <ekmi:PermittedLocation>
1709             <ekmi:LocationName>StrongAuth Server Room</ekmi:LocationName>
1710         </ekmi:PermittedLocation>
1711     </ekmi:PermittedLocations>

```

1712 **Example 2 – An example of a <PermittedLocations> element specifying that the symmetric key may be**  
1713 **used only by applications at a single location at the given GPS coordinates:**

```

1714     <ekmi:PermittedLocations>
1715         <ekmi:PermittedLocation>
1716             <ekmi:LocationName>StrongAuth Server Room</ekmi:LocationName>
1717             <ekmi:Latitude>37.385653 </ekmi:Latitude>
1718             <ekmi:Longitude>-121.993192 </ekmi:Longitude>
1719         </ekmi:PermittedLocation>
1720     </ekmi:PermittedLocations>

```

1721 **Example 3 – An example of a <PermittedLocations> element specifying that the symmetric key may be**  
1722 **used only by applications at multiple locations:**

```

1723     <ekmi:PermittedLocations>
1724         <ekmi:PermittedLocation>
1725             <ekmi:LocationName>Humongous Headquarters</ekmi:LocationName>
1726         </ekmi:PermittedLocation>
1727         <ekmi:PermittedLocation>
1728             <ekmi:LocationName> Humongous Primary Data Center</ekmi:LocationName>
1729             <ekmi:Latitude>37.385653 </ekmi:Latitude>
1730             <ekmi:Longitude>-121.993192 </ekmi:Longitude>
1731         </ekmi:PermittedLocation>
1732         <ekmi:PermittedLocation>
1733             <ekmi:LocationName>Humongous DR Data Center</ekmi:LocationName>
1734             <ekmi:Latitude>68.845901 </ekmi:Latitude>
1735             <ekmi:Longitude>11.393385 </ekmi:Longitude>
1736         </ekmi:PermittedLocation>
1737     </ekmi:PermittedLocations>

```

## 1738 2.19 Element <PermittedTimes> and <PermittedTime>

1739 The element <PermittedTimes>, of type *PermittedTimesType* and its only child-element <PermittedTime>, which  
1740 is an anonymous XSD *ComplexType*, are used to define sets of times during the day between which  
1741 applications are permitted to use a symmetric key within a specific <Symkey> element.

### 1742 Schema Definition:

```

1743     <xsd:complexType name="PermittedTimesType">
1744         <xsd:sequence>
1745             <xsd:element name="PermittedTime" minOccurs="0" maxOccurs="unbounded">
1746                 <xsd:complexType>
1747                     <xsd:sequence>
1748                         <xsd:element name="StartTime">
1749                             <xsd:simpleType>
1750                                 <xsd:restriction base="xsd:time">
1751                                     <xsd:pattern value="\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}"/>
1752                                 </xsd:restriction>
1753                             </xsd:simpleType>
1754                         </xsd:element>
1755                         <xsd:element name="EndTime">
1756                             <xsd:simpleType>
1757                                 <xsd:restriction base="xsd:time">
1758                                     <xsd:pattern value="\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}"/>

```

```

1759         </xsd:restriction>
1760         </xsd:simpleType>
1761     </xsd:element>
1762     </xsd:sequence>
1763 </xsd:complexType>
1764 </xsd:element>
1765 </xsd:sequence>
1766 </xsd:complexType>

```

1767 When it is used within a <KeyUsePolicy> element, there SHALL be only one <PermittedTimes> element within  
1768 the <KeyUsePolicy> element. However, there MAY be an unbounded number of <PermittedTime> elements  
1769 within a <PermittedTimes> element.

1770 NOTE: It is noteworthy to mention that the absence of a <PermittedTimes> element within a <KeyUsePolicy>  
1771 element specifies that applications are permitted the use of the symmetric key at any time of the day, subject to  
1772 complying with all other permission clauses in the <KeyUsePolicy> element, if any.

1773 The <PermittedTime> element identifies an individual set of times between which application are permitted to use  
1774 the symmetric key in question. The <PermittedTime> element consists of the following sub-elements:

- 1775 1. The <StartTime> element identifies the date from which applications may start using the symmetric key  
1776 in question. It is an XSD *Time* type that MUST be specified in a specific pattern (see examples) where  
1777 the first two digits specify the hour, the second two digits specify the minutes and the last two digits  
1778 specify the seconds in a 24 hour format.

1779 There SHALL be only one <StartTime> element within a <PermittedTime> element.

1780 Conforming SKCL implementations SHALL NOT start using the symmetric before the onset of the  
1781 <StartTime> on the client machine.

- 1784 2. The <EndTime> element identifies the time until which applications may use the symmetric key in  
1785 question. It is an XSD *Time* type that MUST be specified in a specific pattern (see examples) where the  
1786 first two digits specify the hour, the second two digits specify the minutes and the last two digits specify  
1787 the seconds in a 24 hour format.

1788 There SHALL be only one <EndTime> element within a <PermittedTime> element.

1789 Conforming SKCL implementations SHALL NOT use the symmetric after the end of the <EndTime> on  
1790 the client machine.

1793 Some examples of the <PermittedTimes> element are shown below; other parts of their enclosing elements are  
1794 not shown for brevity:

1795 **Example 1 – An example of a <PermittedTimes> element with a single<PermittedTime> element. The**  
1796 **<StartTime> specifies 9:00AM on the client machine while the <EndTime> specifies 5:00PM:**

```

1797 <ekmi:PermittedTimes>
1798   <ekmi:PermittedTime>
1799     <ekmi:StartTime>09:00:00</ekmi:StartTime>
1800     <ekmi:EndTime>17:00:00</ekmi:EndTime>
1801   </ekmi:PermittedTime>
1802 </ekmi:PermittedTimes>

```

1803 **Example 2 – An example of a <PermittedTimes> element with two <PermittedTime> elements. For the**  
1804 **first <PermittedTime> element , the <StartTime> element specifies 6:00AM while the <EndTime> element**  
1805 **specifies 12:00 Noon. For the second <PermittedTime> element, the <StartTime> element specifies 3:00**  
1806 **PM in the afternoon, while the <EndTime> element specifies 7:00PM in the evening. This policy might**  
1807 **imply that a symmetric key with this <PermittedTimes> element cannot be used during a lunch break of**  
1808 **12:00 Noon to 3:00PM:**

```

1809 <ekmi:PermittedTimes>
1810   <ekmi:PermittedTime>

```

```

1811         <ekmi:StartTime>06:00:00</ekmi:StartTime>
1812         <ekmi:EndTime>12:00:00</ekmi:EndTime>
1813     </ekmi:PermittedTime>
1814     <ekmi:PermittedTime>
1815         <ekmi:StartTime>15:00:00</ekmi:StartTime>
1816         <ekmi:EndTime>19:00:00</ekmi:EndTime>
1817     </ekmi:PermittedTime>
1818 </ekmi:PermittedTimes>

```

## 1819 2.20 Element <PermittedTransactions>

1820 The element <PermittedTransactions>, of type *PermittedTransactionsType* is used to define the number of  
1821 **encryption** transactions that applications are permitted with a symmetric key within a specific <Symkey>  
1822 element, once the SKCL has started using the symmetric key in question. It does not limit the number of  
1823 **decryption** transactions with the same symmetric key.

### 1824 Schema Definition:

```

1825     <xsd:simpleType name="PermittedTransactionsType">
1826         <xsd:restriction base="xsd:positiveInteger">
1827             <xsd:maxInclusive value="18446744073709551615"/>
1828         </xsd:restriction>
1829     </xsd:simpleType>

```

1830 When it is used within a <KeyUsePolicy> element, there SHALL be only one <PermittedTransactions> element  
1831 within the <KeyUsePolicy> element.

1832 NOTE: It is noteworthy to mention that the absence of a <PermittedTransactions> element within a  
1833 <KeyUsePolicy> element specifies that applications are permitted the use of the symmetric key for an unlimited  
1834 number of encryption transactions, subject to complying with all other permission clauses in the <KeyUsePolicy>  
1835 element, if any.

1836 The <PermittedTransactions> element, of the XSD *positiveInteger* type, identifies the precise number of  
1837 encryption transactions that may be performed by the symmetric key in question, once the key has been used by  
1838 conforming SKCL implementations for the first time.

1839 Some examples of the <PermittedTransactions> element are shown below; other parts of their enclosing  
1840 elements are not shown for brevity:

1841 **Example 1 – An example of a <PermittedTransactions> element specifying that the symmetric key may be  
1842 used only for a single encryption transaction by an SKCL:**

```
1843     <ekmi:PermittedTransactions>1</ekmi:PermittedTransactions>
```

1844 **Example 2 – An example of a <PermittedTransactions> element specifying that the symmetric key may be  
1845 used only for 100 transactions by an SKCL:**

```
1846     <ekmi:PermittedTransactions>100</ekmi:PermittedTransactions>
```

## 1847 2.21 Element <PermittedUses> and <PermittedUse>

### 1848 Schema Definition:

1849 The element <PermittedUses>, of type *PermittedUsesType*, is used to define the specific ways in which  
1850 applications are permitted to use a symmetric key within a specific <Symkey> element.

```

1851     <xsd:complexType name="PermittedUsesType" mixed="true">
1852         <xsd:sequence>
1853             <xsd:element name="PermittedUse" minOccurs="0" maxOccurs="unbounded">
1854                 <xsd:simpleType>
1855                     <xsd:restriction base="xsd:string">

```

```

1856         <xsd:maxLength value="256"/>
1857         <xsd:whiteSpace value="preserve"/>
1858     </xsd:restriction>
1859 </xsd:simpleType>
1860 </xsd:element>
1861 <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
1862 </xsd:sequence>
1863 </xsd:complexType>

```

1864 When it is used within a <KeyUsePolicy> element, there SHALL be only one <PermittedUses> element within the  
1865 <KeyUsePolicy> element. However, there MAY be an unbounded number of <PermittedUse> elements within  
1866 the <PermittedUses> element.

1867 NOTE: It is noteworthy to mention that the absence of a <PermittedUses> element within a <KeyUsePolicy>  
1868 element specifies that applications are permitted the use of the symmetric key for any purpose, subject to  
1869 complying with all other permission clauses in the <KeyUsePolicy> element, if any.

1870 Some examples of the <PermittedUses> element are shown below; other parts of their enclosing elements are  
1871 not shown for brevity:

1872 **Example 1 – An example of a <PermittedUses> element specifying that the symmetric key may be used**  
1873 **only by VPN applications for session encryption keys:**

```

1874 <ekmi:PermittedUses>
1875   <ekmi:PermittedUse>VPN</ekmi:PermittedUse>
1876 </ekmi:PermittedUses>

```

1877 **Example 2 – An example of a <PermittedUses> element specifying that the symmetric key may be used**  
1878 **only by applications on laptops and Personal Digital Assistants (PDA):**

```

1879 <ekmi:PermittedUses>
1880   <ekmi:PermittedUse>Laptop</ekmi:PermittedUse>
1881   <ekmi:PermittedUse>PDA</ekmi:PermittedUse>
1882 </ekmi:PermittedUses>

```

## 1883 2.22 Element <KeyCachePolicyRequest>

1884 The <KeyCachePolicyRequest> element is used to request a key-cache policy from the SKS server , so the  
1885 client may know if and how to cache symmetric keys locally.

1886 While it is a top-level element within this specification, a <SymkeyRequest> element MUST be enclosed within a  
1887 **SOAP Body** element of a **SOAP Envelope** to conform to the security requirements of this specification. The  
1888 **SOAP Header** of the **SOAP Envelope** MUST enclose a **Security** element conforming to [WSS] with a  
1889 **ValueType** attribute containing the value <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3>. The **Security** element must conform to all other requirements of the specified security profile  
1890 in [WSS] to form a well-formed, secure message.  
1891

### 1892 Schema Definition:

```

1893 <xsd:element name="KeyCachePolicyRequest">
1894   <xsd:complexType>
1895     <xsd:annotation>
1896       <xsd:documentation>
1897         No elements/attributes are defined for KeyCachePolicyRequest.
1898       </xsd:documentation>
1899     </xsd:annotation>
1900   </xsd:complexType>
1901 </xsd:element>

```



1902 The <KeyCachePolicyRequest> has no children elements. The SOAP Header of the signed request provides  
1903 the SKS server with all the information it needs to process the request: the identity of the requester, strong  
1904 authentication and message integrity of the request.

1905 Some examples of the <SymkeyRequest> element are as follows:

1906 **Example 1 – An example of a <KeyCachePolicyRequest>; the surrounding SOAP envelope is not**  
1907 **displayed here for brevity:**

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

## 1910 2.23 Element <KeyCachePolicyResponse>

1911 The <KeyCachePolicyResponse> element is the response sent by an SKS Server to a client that requested a  
1912 key-cache policy through a <KeyCachePolicyRequest>. The <KeyCachePolicyResponse> contains  
1913 policy elements, which define rules that conforming implementations of the SKCL MUST adhere to when  
1914 caching symmetric keys sent by the SKS Server.

### 1915 Schema Definition:

```
1916     <xsd:element name="KeyCachePolicyResponse">  
1917         <xsd:complexType>  
1918             <xsd:sequence>  
1919                 <xsd:element  
1920                     name="KeyCachePolicy"  
1921                     type="ekmi:KeyCachePolicyType"  
1922                     minOccurs="1" maxOccurs="unbounded"/>  
1923             </xsd:sequence>  
1924         </xsd:complexType>  
1925     </xsd:element>
```

1926 The <KeyCachePolicyResponse> element consists of a minimum of one, but an unbounded number of  
1927 <KeyCachePolicy> children elements.

## 1928 2.24 Element <KeyCachePolicy>

1929 The <KeyCachePolicy> element contains policy elements, which define rules that conforming implementations  
1930 of the SKCL MUST adhere to when caching symmetric keys sent by the SKS Server.

### 1931 Schema Definition:

```
1932     <xsd:element name="KeyCachePolicyResponse">  
1933         <xsd:complexType>  
1934             <xsd:sequence>  
1935                 <xsd:element  
1936                     name="KeyCachePolicy"  
1937                     type="ekmi:KeyCachePolicyType"  
1938                     minOccurs="1" maxOccurs="unbounded"/>  
1939             </xsd:sequence>  
1940         </xsd:complexType>  
1941     </xsd:element>  
  
1942     <xsd:complexType name="KeyCachePolicyType" mixed="true">  
1943         <xsd:sequence>  
1944             <xsd:element name="KeyCachePolicyID" type="tns:TwoPartIDType"/>  
1945             <xsd:element name="PolicyName">  
1946                 <xsd:simpleType>  
1947                     <xsd:restriction base="xsd:string">  
1948                         <xsd:maxLength value="255"/>
```

```

1949         <xsd:whiteSpace value="preserve"/>
1950     </xsd:restriction>
1951 </xsd:simpleType>
1952 </xsd:element>
1953 <xsd:element name="Description" nillable="true">
1954     <xsd:simpleType>
1955         <xsd:restriction base="xsd:string">
1956             <xsd:maxLength value="2048"/>
1957             <xsd:whiteSpace value="preserve"/>
1958         </xsd:restriction>
1959     </xsd:simpleType>
1960 </xsd:element>
1961 <xsd:element name="KeyClass" type="tns:KeyClassType"/>
1962 <xsd:element name="StartDate" type="xsd:date"/>
1963 <xsd:element name="EndDate" type="xsd:date" nillable="true"/>
1964 <xsd:element name="PolicyCheckInterval">
1965     <xsd:simpleType>
1966         <xsd:restriction base="xsd:nonNegativeInteger">
1967             <xsd:minInclusive value="0"/>
1968             <xsd:maxInclusive value="2592000"/>
1969         </xsd:restriction>
1970     </xsd:simpleType>
1971 </xsd:element>
1972 <xsd:element name="Status" type="tns:StatusType"/>
1973 <xsd:element
1974     name="NewKeysCacheDetail"
1975     type="tns:KeyCacheDetailType"
1976     minOccurs="0"/>
1977 <xsd:element
1978     name="UsedKeysCacheDetail"
1979     type="tns:KeyCacheDetailType"
1980     minOccurs="0"/>
1981 </xsd:sequence>
1982 </xsd:complexType>

```

1983 The <KeyCachePolicy> element is of the **KeyCachePolicyType** and consists of the following child elements:

1984 1. <KeyCachePolicyID> [Required]

1985

1986 The <KeyCachePolicyID> element, of type **TwoPartIDType**, identifies the unique policy object within  
1987 the SKMS. There SHALL be only one <KeyCachePolicyID> element within a <KeyCachePolicy>  
1988 element.

1989

1990 The **TwoPartIDType** is specified in Section 2.8.

1991 2. <PolicyName> [Required]

1992

1993 The <PolicyName> element, of type XSD **String**, with a maximum length of 255 characters, identifies  
1994 a unique name given to this <KeyCachePolicy>. There SHALL be only one <PolicyName> element  
1995 within a <KeyCachePolicy> element.

1996 3. <Description> [Required]

1997

1998 The KeyCachePolicy element, of type XSD **String**, with a maximum length of 2048 characters,  
1999 provides a human-readable description of this policy. There SHALL be only one <Description>  
2000 element within a <KeyCachePolicy> element.

2001

2002 The <Description> MAY be an empty element, but MUST exist within the <KeyCachePolicy>  
2003 element.

- 2004 4. <KeyClass> [Required]  
2005  
2006 This element of type **KeyClassType** identifies the key-class of the symmetric key to which this policy  
2007 applies.
- 2008 5. <StartDate> [Required]  
2009  
2010 The <StartDate> element , of type XSD **dateTime**, specifies the date and time at which this policy  
2011 becomes effective. There SHALL be only one <StartDate> element within a <KeyCachePolicy>  
2012 element.
- 2013 6. <EndDate> [Required]  
2014  
2015 The <EndDate> element , of type XSD **dateTime**, specifies the date and time at which this policy  
2016 expires. There SHALL be only one <EndDate> element within a <KeyCachePolicy> element.  
2017  
2018 The <EndDate> MAY be an empty element, but MUST exist within the <KeyCachePolicy> element.
- 2019 7. <PolicyCheckInterval> [Required]  
2020  
2021 The <PolicyCheckInterval> element , of type XSD **nonNegativeInteger**, specifies the frequency at  
2022 which the client SHALL check the SKS server for updates to this policy. This frequency is specified in  
2023 seconds and SHALL NOT exceed 2592000 seconds (30 calendar days). There SHALL be only one  
2024 <PolicyCheckInterval> element within a <KeyCachePolicy> element.
- 2025 8. <Status> [Required]  
2026  
2027 The <Status> element, of type **StatusType**, identifies the current status of this policy within the SKMS.  
2028 There SHALL be only one <Status> element within a <KeyCachePolicy> element.  
2029  
2030 The **StatusType** is specified in Section 2.11.
- 2031 9. <NewKeysCacheDetail> [Required]  
2032  
2033 The <NewKeysCacheDetail> element, of type **KeyCacheDetailType**, defines how many new (unused  
2034 for any encryption transaction) symmetric keys a client may cache, and for how long. It is the  
2035 responsibility of the conforming SKCL implementation to enforce these rules.  
2036  
2037 The absence of the <NewKeysCacheDetail> element implies that new symmetric keys SHALL NEVER  
2038 be cached on the client. New keys may be cached only when this element exists, and SHALL conform  
2039 to the rules specified in this element.  
2040  
2041 When it exists, there SHALL be only one <NewKeysCacheDetail> element in a <KeyCachePolicy>  
2042 element.  
2043  
2044 The **KeyCacheDetailType** is specified in Section 2.22.
- 2045 10. <UsedKeysCacheDetail> [Required]  
2046  
2047 The <UsedKeysCacheDetail> element, of type **KeyCacheDetailType**, defines how many used  
2048 symmetric keys a client may cache, and for how long. It is the responsibility of the conforming SKCL  
2049 implementation to enforce these rules.  
2050  
2051 The absence of the <UsedKeysCacheDetail> element implies that used symmetric keys SHALL  
2052 NEVER be cached on the client. Used keys may be cached only when this element exists, and SHALL  
2053 conform to the rules specified in this element.  
2054  
2055 When it exists, there SHALL be only one <UsedKeysCacheDetail> element in a <KeyCachePolicy>  
2056 element.  
2057  
2058 The **KeyCacheDetailType** is specified in Section 2.22.

2059 Some examples of the <KeyUsePolicy> element are as follows.

2060 **Example 1 – A <KeyCachePolicy> that is valid between January 01, 2008 and December 31, 2008. It**  
2061 **requires the client to check for policy updates every day and allows 3 new and 3 used keys to be cached**  
2062 **for upto 90 days:**

```
2063 <ekmi:KeyCachePolicy>
2064 <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
2065 <ekmi:PolicyName>
2066 Corporate Laptop Symmetric Key Caching Policy
2067 </ekmi:PolicyName>
2068 <ekmi:Description>
2069 This policy defines how company-issued laptops will manage
2070 symmetric keys used for file/disk encryption in their local
2071 cache. This policy must be used by all laptops that use
2072 the company EKMI.
2073 </ekmi:Description>
2074 <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
2075 <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
2076 <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
2077 <ekmi:Status>Active</ekmi:Status>
2078 <ekmi:NewKeysCacheDetail>
2079 <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
2080 <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
2081 </ekmi:NewKeysCacheDetail>
2082 <ekmi:UsedKeysCacheDetail>
2083 <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
2084 <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
2085 </ekmi:UsedKeysCacheDetail>
2086 </ekmi:KeyCachePolicy>
```

2087 **Example 2 – A <KeyCachePolicy> that is effective starting January 01, 2008 and never expires. It does**  
2088 **NOT permit any caching of symmetric keys through the absence of the detail elements on caching:**

```
2089 <ekmi:KeyCachePolicy>
2090 <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
2091 <ekmi:PolicyName>
2092 No Caching Policy
2093 </ekmi:PolicyName>
2094 <ekmi:Description>
2095 This policy is for high-risk, always-connected machines on the
2096 network, which will never cache symmetric keys locally. This
2097 policy never expires (but checks monthly for any updates).
2098 </ekmi:Description>
2099 <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
2100 <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
2101 <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
2102 <ekmi:Status>Active</ekmi:Status>
2103 </ekmi:KeyCachePolicy>
```

## 2104 2.25 Type *KeyCacheDetailType*

2105 The *KeyCacheDetailType* type allows SKS servers to specify precisely how many symmetric keys MAY be  
2106 cached on the client machine, and for how long.

### 2107 Schema Definition:

```
2108 <xsd:complexType name="KeyCacheDetailType">
2109 <xsd:sequence>
2110 <xsd:element name="MaximumKeys" minOccurs="1">
```

```

2111         <xsd:simpleType>
2112             <xsd:restriction base="xsd:integer">
2113                 <xsd:minInclusive value="0"/>
2114                 <xsd:maxInclusive value="18446744073709551615"/>
2115             </xsd:restriction>
2116         </xsd:simpleType>
2117     </xsd:element>
2118     <xsd:element name="MaximumDuration" minOccurs="1">
2119         <xsd:simpleType>
2120             <xsd:restriction base="xsd:integer">
2121                 <xsd:minInclusive value="0"/>
2122                 <xsd:maxInclusive value="18446744073709551615"/>
2123             </xsd:restriction>
2124         </xsd:simpleType>
2125     </xsd:element>
2126 </xsd:sequence>
2127 </xsd:complexType>

```

2128 The **KeyCacheDetailType** consists of the following child elements:

2129 1. <MaximumKeys> [Required]

2130

2131 The <MaximumKeys> element, of type XSD **Integer**, specifies the maximum number of symmetric keys  
2132 that MAY be cached on a client machine. It SHALL be a positive number between the values 0 and  
2133 18446744073709551615. There SHALL be only one <MaximumKeys> element within an element that  
2134 uses the **KeyCacheDetailType**.

2135 2. <MaximumDuration> [Required]

2136

2137 The <MaximumDuration> element, of type XSD **Integer**, specifies the maximum number of seconds  
2138 that symmetric keys MAY be cached on a client machine. It SHALL be a positive number between the  
2139 values 0 and 18446744073709551615. There SHALL be only one <MaximumDuration> element  
2140 within an element that uses the **KeyCacheDetailType**.

2141 Examples of the **KeyCacheDetailType** when used in the <KeyCachePolicy> element are as follows.

2142 **Example 1 – A <KeyCachePolicy> that is valid between January 01, 2008 and December 31, 2008. It**  
2143 **requires the client to check for policy updates every day and allows 3 new and 3 used keys to be cached**  
2144 **for upto 90 days:**

```

2145 <ekmi:KeyCachePolicy>
2146     <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
2147     <ekmi:PolicyName>
2148         Corporate Laptop Symmetric Key Caching Policy
2149     </ekmi:PolicyName>
2150     <ekmi:Description>
2151         This policy defines how company-issued laptops will manage
2152         symmetric keys used for file/disk encryption in their local
2153         cache. This policy must be used by all laptops that use
2154         the company EKMI.
2155     </ekmi:Description>
2156     <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
2157     <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
2158     <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
2159     <ekmi>Status>Active</ekmi>Status>
2160     <ekmi:NewKeysCacheDetail>
2161         <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
2162         <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
2163     </ekmi:NewKeysCacheDetail>
2164     <ekmi:UsedKeysCacheDetail>
2165         <ekmi:MaximumKeys>3</ekmi:MaximumKeys>

```

```
2166         <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
2167     </ekmi:UsedKeysCacheDetail>
2168 </ekmi:KeyCachePolicy>
```

2169 **Example 1 – A <KeyCachePolicy> that is valid between January 01, 2008 and December 31, 2008. It**  
2170 **requires the client to check for policy updates every day and allows 1 new and 0 used keys to be cached**  
2171 **for upto 15 days:**

```
2172 <ekmi:KeyCachePolicy>
2173     <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
2174     <ekmi:PolicyName>
2175         Corporate Laptop Symmetric Key Caching Policy
2176     </ekmi:PolicyName>
2177     <ekmi:Description>
2178         This policy defines how company-issued laptops will manage
2179         symmetric keys used for file/disk encryption in their local
2180         cache. This policy must be used by all laptops that use
2181         the company EKMI.
2182     </ekmi:Description>
2183     <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
2184     <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
2185     <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
2186     <ekmi:Status>Active</ekmi:Status>
2187     <ekmi:NewKeysCacheDetail>
2188         <ekmi:MaximumKeys>1</ekmi:MaximumKeys>
2189         <ekmi:MaximumDuration>1296000</ekmi:MaximumDuration>
2190     </ekmi:NewKeysCacheDetail>
2191     <ekmi:UsedKeysCacheDetail>
2192         <ekmi:MaximumKeys>0</ekmi:MaximumKeys>
2193         <ekmi:MaximumDuration>1296000</ekmi:MaximumDuration>
2194     </ekmi:UsedKeysCacheDetail>
2195 </ekmi:KeyCachePolicy>
```

---

2196 **Appendix A. Acknowledgments**

2197 The following individuals have participated in the creation of this specification and are gratefully  
2198 acknowledged

2199 **Participants:**

2200 •

2201

2202

---

## Appendix B. Revision History

Version	Date	Author	Notes
DRAFT 4	June 08, 2008	Arshad Noor	Initial version
DRAFT 5	June 17, 2008	Arshad Noor	Moved non-normative sections to their own document. KeyClass element was added to KeyCachePolicy. KeyCachePolicy is now embedded inside a KeyCachePolicyResponse.

2203

2204

2205



2206

---

## Appendix C. Non-Normative Text

2207

---

## Appendix D. SKSML Error Codes and Error Messages