[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)

Editor(s):

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

Related Work:

This specification replaces or supercedes:

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

This specification is related to:

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

Declared XML Namespace(s):

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

Abstract:

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

Status:

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

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

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

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

Notices

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

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

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

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

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

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

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

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

Table of Contents

1 Introduction

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

1.1 Terminology

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

1.2 Glossary

3DES – Triple Data Encryption Standard

AES – Advanced Encryption Standard

Base64 – An encoding scheme for representing data

Ciphertext – Encrypted data

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

DES – Data Encryption Standard

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

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

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

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

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

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

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

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

Plaintext – Unencrypted data

SHA – Secure Hashing Algorithm

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

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

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

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

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

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

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

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

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

SOAP – Simple Object Access Protocol

SOAP Body – The content part of a SOAP message

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

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

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

Symkey - A symmetric encryption key

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

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

1.3 Normative References

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

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

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

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

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

[WSS] Web Services Security – SOAP Message Security 1.0
OASIS Standard 200401, March 2004
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
<ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
  <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
  <ekmi:KeyClasses>
       <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
       <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
       <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
       <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
       <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
       <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
       <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
       <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
       <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>
  </ekmi:KeyClasses>
</ekmi:SymkeyRequest>
```

2.2 Element <GlobalKeyID>

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

Schema Definition:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.3 Element <KeyClasses> and <KeyClass>

The <KeyClasses> 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.

Schema Definition:

```
<xsd:complexType name="KeyClassesType">
   <xsd:sequence>
      <xsd:element
         name="KeyClass"
         type="tns:KeyClassType"
         minOccurs="1"
         maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>


<xsd:simpleType name="KeyClassType">
   <xsd:restriction base="xsd:string">
      <xsd:maxLength value="255"/>
   </xsd:restriction>
</xsd:simpleType>
```

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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
<ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
  <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
  <ekmi:KeyClasses>
       <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
       <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
       <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
       <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
  </ekmi:KeyClasses>
</ekmi:SymkeyRequest>
```

2.4 Element <SymkeyResponse>

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

While <SymkeyResponse> is a top-level element within this specification, it 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="SymkeyResponse">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element
                    name="Symkey"
                    type="tns:SymkeyType"
                    minOccurs="0"
                    maxOccurs="unbounded"/>
                <xsd:element
                    name="SymkeyError"
                    type="tns:SymkeyErrorType"
                    minOccurs="0"
                    maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
```

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

1. <Symkey> [Optional]

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

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

2. <SymkeyError> [Optional]

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

   The <SymkeyError> element is specified in Section 2.6.

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

Example 1 – A single symmetric key within a response:

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

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

Example 2 – A response with three symmetric keys:

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

Example 3 – A response with an error:

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

Example 4 – A response with multiple errors:

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

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

```
    <ekmi:SymkeyRequest
        xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
        <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
        <ekmi:KeyClasses>
            <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
            <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
            <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
            <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
            <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
            <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
            <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
            <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
            <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>
        </ekmi:KeyClasses>
    </ekmi:SymkeyRequest>
```

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

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

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

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

2.5 Element <Symkey>

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

Schema Definition:

```
    <xsd:complexType name="SymkeyType">
        <xsd:sequence>
            <xsd:element name="GlobalKeyID" type="ekmi:GlobalKeyIDType"/>
            <xsd:element name="KeyUsePolicy" type="ekmi:KeyUsePolicyType"/>
            <xsd:element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>
            <xsd:element ref="xenc:CipherData"/>
            <xsd:element ref="xenc:EncryptionProperties" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
```

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

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

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

1. <GlobalKeyID> [Required]

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

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

2. <KeyUsePolicy> [Required]

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

   The <KeyUsePolicy> element is specified in Section 2.7.

3. <EncryptionMethod> [Required]

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

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

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

4. <CipherData> [Required]

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

   <EncryptionProperties> [Optional]

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

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

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

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

Example 1 – A symmetric key within a response:

```
<ekmi:SymkeyResponse
     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
     <ekmi:Symkey>
        <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
        <ekmi:KeyUsePolicy>.....</ekmi:KeyUsePolicy>
        <ekmi:EncryptionMethod
             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        <xenc:CipherData>
             <xenc:CipherValue>
                E9zWB/y93hVSzeTLiDcQoDxmlNxTux+SffMNwCJmt1dIqzQHBnpdQ8
                1g6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxj
                fg1pU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
           </xenc:CipherValue>
        </xenc:CipherData>
     </ekmi:Symkey>
</ekmi:SymkeyResponse>
```

Example 2 – A response with multiple symmetric keys:

```
<ekmi:SymkeyResponse
     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
     <ekmi:Symkey>
        <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
        <ekmi:KeyUsePolicy>.....</ekmi:KeyUsePolicy>
        <ekmi:EncryptionMethod
             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
        <xenc:CipherData>
             <xenc:CipherValue>
                E9zWB/y93hVSzeTLiDcQoDxmlNxTux+SffMNwCJmt1dIqzQHBnpdQ8
                1g6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxj
                fg1pU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
           </xenc:CipherValue>
        </xenc:CipherData>
     </ekmi:Symkey>
     <ekmi:Symkey>
        <ekmi:GlobalKeyID>10514-1-236</ekmi:GlobalKeyID>
        <ekmi:KeyUsePolicy>.....</ekmi:KeyUsePolicy>
        <ekmi:EncryptionMethod
             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
        <xenc:CipherData>
             <xenc:CipherValue>
                Qbg65cy93hVSzeTLiDcQoDxmlNxTux+SffMNwCJmt1dIqzQHBnpdQ8
                7k6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxj
                uyecU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
             </xenc:CipherValue>
        </xenc:CipherData>
```

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

2.6 Element <SymkeyError>

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

Schema Definition:

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

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

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

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

1. <RequestedGlobalKeyID> [Required]

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

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

2. <RequestedKeyClass> [Optional]

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

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

3. <ErrorCode> [Required]

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

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

4. <ErrorMessage> [Required]

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

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

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

Example 1 – An error within a response:

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

Example 2 – Multiple errors within a response:

```
<ekmi:SymkeyResponse
     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
     <ekmi:SymkeyError>
        <ekmi:RequestedGlobalKeyID>10514-1-0</ekmi:RequestedGlobalKeyID>
        <ekmi:ErrorCode>SKS-100001</ekmi:ErrorCode>
        <ekmi:ErrorMessage>Invalid GlobalKeyID</ekmi:ErrorMessage>
    </ekmi:SymkeyError>
    <ekmi:SymkeyError>
        <ekmi:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>
        <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
        <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
    </ekmi:SymkeyError>
</ekmi:SymkeyResponse>
```

2.7 Element <KeyUsePolicy>

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

Schema Definition:

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

     <xsd:sequence>
         <xsd:element name="KeyUsePolicyID" type="tns:TwoPartIDType"/>
         <xsd:element name="PolicyName">
             <xsd:simpleType>
                 <xsd:restriction base="xsd:string">
                     <xsd:maxLength value="255"/>
                 </xsd:restriction>
             </xsd:simpleType>
```

```
            </xsd:element>
            <xsd:element name="KeyClass" type="tns:KeyClassType"/>
            <xsd:element name="KeyAlgorithm" type="tns:EncryptionAlgorithmType"/>
            <xsd:element name="KeySize" type="tns:KeySizeType"/>
            <xsd:element name="Status" type="tns:StatusType"/>
            <xsd:element name="Permissions" type="tns:PermissionsType"/>
        </xsd:sequence>
    </xsd:complexType>
```

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

1. <KeyUsePolicyID> [Required]

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

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

2. <PolicyName> [Required]

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

3. <KeyClass> [Required]

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

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

4. <KeyAlgorithm> [Required]

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

   The <KeyAlgorithm> element is specified in Section 2.9.

5. <KeySize> [Required]

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

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

   The ***KeySizeType*** is specified in Section 2.10.

6. <Status> [Required]

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

   The ***StatusType*** is specified in Section 2.11.

7. <Permissions> [Required]

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

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

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

   The ***PermissionsType*** is specified in Section 2.12.

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

Example 1 – A <KeyUsePolicy> with some permissions:

```
<ekmi:KeyUsePolicy>
   <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>
   <ekmi:PolicyName>DES-EDE KeyUsePolicy</ekmi:PolicyName>
   <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
   <ekmi:KeyAlgorithm>
      http://www.w3.org/2001/04/xmlenc#tripledes-cbc
   </ekmi:KeyAlgorithm>
   <ekmi:KeySize>192</ekmi:KeySize>
   <ekmi:Status>Active</ekmi:Status>
   <ekmi:Permissions>
      <ekmi:PermittedApplications>
         <ekmi:PermittedApplication>
            <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
            <ekmi:ApplicationName>Payroll Application</ekmi:ApplicationName>
            <ekmi:Version>1.0</ekmi:Version>
            <ekmi:DigestAlgorithm>
               http://www.w3.org/2000/09/xmldsig#sha1
            </ekmi:DigestAlgorithm>
            <ekmi:DigestValue>NIG4bKkt4cziEqFFuOoBTM81efU=</ekmi:DigestValue>
         </ekmi:PermittedApplication>
      </ekmi:PermittedApplications>
      <ekmi:PermittedTimes>
         <ekmi:PermittedTime>
            <ekmi:StartTime>07:00:00</ekmi:StartTime>
            <ekmi:EndTime>19:00:00</ekmi:EndTime>
         </ekmi:PermittedTime>
      </ekmi:PermittedTimes>
   </ekmi:Permissions>
</ekmi:KeyUsePolicy>
```

Example 2 – A <KeyUsePolicy> with some permissions:

```
<ekmi:KeyUsePolicy>
   <ekmi:KeyUsePolicyID>10514-2</ekmi:KeyUsePolicyID>
   <ekmi:PolicyName>Laptop KeyUsePolicy</ekmi:PolicyName>
   <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
   <ekmi:KeyAlgorithm>
      http://www.w3.org/2001/04/xmlenc#aes256-cbc
   </ekmi:KeyAlgorithm>
   <ekmi:KeySize>256</ekmi:KeySize>
   <ekmi:Status>Active</ekmi:Status>
   <ekmi:Permissions/>
</ekmi:KeyUsePolicy>
```

2.8 Type *TwoPartIDType*

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

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

Schema Definition:

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

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

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

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

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

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

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

Example 3 – A minimum-length *TwoPartIDType* :

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

Example 4 – A maximum-length *TwoPartIDType* :

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

2.9 Element <KeyAlgorithm>

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

Schema Definition:

```
<xsd:simpleType name="EncryptionAlgorithmType">
     <xsd:restriction base="xsd:anyURI">
        <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
        <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
        <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes192-cbc"/>
        <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
     </xsd:restriction>
 </xsd:simpleType>
```

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

1. Triple Data Encryption Standard (3DES)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.10 Element <KeySize>

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

Schema Definition:

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

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

Currently, the following three key-sizes are supported:

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

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

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

```
<ekmi:KeyUsePolicy>
   ...
   <ekmi:KeySize>128</ekmi:KeySize>
   ...
</ekmi:KeyUsePolicy>
```

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

```
<ekmi:KeyUsePolicy>
   ...
   <ekmi:KeySize>192</ekmi:KeySize>
   ...
</ekmi:KeyUsePolicy>
```

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

```
<ekmi:KeyUsePolicy>
   ...
   <ekmi:KeySize>256</ekmi:KeySize>
   ...
</ekmi:KeyUsePolicy>
```

2.11 Element <Status>

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

Schema Definition:

```
<xsd:simpleType name="StatusType">
     <xsd:restriction base="xsd:string">
         <xsd:enumeration value="Active"/>
         <xsd:enumeration value="Default"/>
         <xsd:enumeration value="Inactive"/>
         <xsd:enumeration value="Other"/>
```

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

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

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

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

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

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

```
<ekmi:KeyUsePolicy>
    ...
    <ekmi:Status>Active</ekmi:Status>
    ...
</ekmi:KeyUsePolicy>
```

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

```
<ekmi:KeyUsePolicy>
    ...
    <ekmi:Status>Inactive</ekmi:Status>
    ...
</ekmi:KeyUsePolicy>
```

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

```
<ekmi:KeyUsePolicy>
    ...
    <ekmi:Status>Default</ekmi:Status>
    ...
</ekmi:KeyUsePolicy>
```

2.12 Element <Permissions>

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

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

Schema Definition:

```
<xsd:complexType name="PermissionsType">
<xsd:sequence>
    <xsd:element
              name="PermittedApplications"
```

```
                type="tns:PermittedApplicationsType"
                minOccurs="0"/>
        <xsd:element
                name="PermittedDates"
                type="tns:PermittedDatesType"
                minOccurs="0"/>
        <xsd:element
                name="PermittedDays"
                type="tns:PermittedDaysType"
                minOccurs="0"/>
        <xsd:element
                name="PermittedDuration"
                type="tns:PermittedDurationType"
                minOccurs="0"/>
        <xsd:element
                name="PermittedLevels"
                type="tns:PermittedLevelsType"
                minOccurs="0"/>
        <xsd:element
                name="PermittedLocations"
                type="tns:PermittedLocationsType"
                minOccurs="0"/>
        <xsd:element
                name="PermittedTimes"
                type="tns:PermittedTimesType"
                minOccurs="0"/>
        <xsd:element
                name="PermittedTransactions"
                type="tns:PermittedTransactionsType"
                minOccurs="0"/>
        <xsd:element
                name="PermittedUses"
                type="tns:PermittedUsesType"
                minOccurs="0"/>
        <xsd:element
                name="Other"
                type="xsd:anyType"
                minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
```

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

1. The optional <PermittedApplications> element identifies applications that are permitted the use of the symmetric key in question.

   The absence of the <PermittedApplications> element in a <Permissions> element implies that all applications are permitted to use the key. Identifying a specific application restricts the use of the key to only the identified applications.

   The <PermittedApplications> element is specified in Section 2.13.

2. The optional <PermittedDates> element identifies the dates during which applications are permitted the use of the symmetric key in question.

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

   The <PermittedDates> element is specified in Section 2.12.

3. The optional <PermittedDays> element identifies the days of week during which applications are permitted the use of the symmetric key in question.

   The absence of the <PermittedDays> element in a <Permissions> element implies that applications are permitted to use the key on any day of the week. Identifying specific days restricts the use of the key to only the identified days.

   The <PermittedDays> element is specified in Section 2.15.

4. The optional <PermittedDuration> element identifies the duration (in seconds) in which applications are permitted the use of the symmetric key in question, once the SKCL starts using the symmetric key.

   The absence of the <PermittedDuration> element in a <Permissions> element implies that applications are permitted to use the key for any duration after it has been used. Identifying the specific duration restricts the use of the key to only the period after the start of the use of the key.

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

   The <PermittedDuration> element is specified in Section 2.x16

5. Within a Multi-Level Security (MLS) system, the optional <PermittedLevels> element identifies the levels at which applications are permitted the use of the symmetric key in question.

   The absence of the <PermittedLevels> element in a <Permissions> element implies that applications are permitted to use the key on any level of security. Identifying specific level(s) restricts the use of the key to only the identified level(s).

   The <PermittedLevels> element is specified in Section 2.x17

6. The optional <PermittedLocations> element identifies physical geographic locations where applications are permitted the use of the symmetric key in question.

   The absence of the <PermittedLocations> element in a <Permissions> element implies that applications are permitted to use the key at any physical location. Identifying specific locations restricts the use of the key to only the identified locations.

   The <PermittedLocations> element is specified in Section 2.18.

7. The optional <PermittedTimes> element identifies the times of day during which applications are permitted the use of the symmetric key in question.

   The absence of the <PermittedTimes> element in a <Permissions> element implies that applications are permitted to use the key at any time of day. Identifying specific times restricts the use of the key to only the duration of the identified times.

   The <PermittedTimes> element is specified in Section 2.19.

8. The optional <PermittedTransactions> element identifies the number of encryption transactions that applications are permitted, with the use of the symmetric key in question.

   The absence of the <PermittedTransactions> element in a <Permissions> element implies that applications are permitted to use the key for as many encryption transactions as necessary. Identifying a specific number of transactions restricts the use of the key to only the limit identified in the element.

   The <PermittedTransactions> element is specified in Section 2.20.

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

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

   The <PermittedUses> element is specified in Section 2.21.

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

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

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

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

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

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

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

```
<ekmi:Permissions>
    <ekmi:PermittedApplications>
      <ekmi:PermittedApplication>
        <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
        <ekmi:ApplicationName>Payroll Application</ekmi:ApplicationName>
        <ekmi:Version>1.0</ekmi:Version>
        <ekmi:DigestAlgorithm>http://www.w3.org/2000/09/xmldsig#sha1</ekmi:DigestAlgorithm>
        <ekmi:DigestValue>NIG4bKkt4cziEqFFuOoBTM81efU=</ekmi:DigestValue>
      </ekmi:PermittedApplication>
    </ekmi:PermittedApplications>
    <ekmi:PermittedDates>
      <ekmi:PermittedDate>
        <ekmi:StartDate>2008-01-01</ekmi:StartDate>
        <ekmi:EndDate>2008-12-31</ekmi:EndDate>
      </ekmi:PermittedDate>
    </ekmi:PermittedDates>
    <ekmi:PermittedTimes>
      <ekmi:PermittedTime>
        <ekmi:StartTime>07:00:00</ekmi:StartTime>
        <ekmi:EndTime>19:00:00</ekmi:EndTime>
```

```
            </ekmi:PermittedTime>
        </ekmi:PermittedTimes>
    </ekmi:Permissions>
```

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

```
    <ekmi:Permissions>
        <ekmi:PermittedApplications>
            <ekmi:PermittedApplication>
                <ekmi:ApplicationID>10514-24</ekmi:ApplicationID>
                <ekmi:ApplicationName>Employee Tax Reporting Application</ekmi:ApplicationName>
                <ekmi:Version>3.3</ekmi:Version>
                <ekmi:DigestAlgorithm>http://www.w3.org/2000/09/xmldsig#sha1</ekmi:DigestAlgorithm>
                <ekmi:DigestValue>G4bsdfKkt4cziEqFFuOoBTM81efU=</ekmi:DigestValue>
            </ekmi:PermittedApplication>
            <ekmi:PermittedApplication>
                <ekmi:ApplicationID>10514-25</ekmi:ApplicationID>
                <ekmi:ApplicationName>IRS Tax Reporting Application</ekmi:ApplicationName>
                <ekmi:Version>2.1</ekmi:Version>
                <ekmi:DigestAlgorithm>http://www.w3.org/2000/09/xmldsig#sha1</ekmi:DigestAlgorithm>
                <ekmi:DigestValue>4uyb4sd234cziqzlmnuOoBTMa08=</ekmi:DigestValue>
            </ekmi:PermittedApplication>
        </ekmi:PermittedApplications>
        <ekmi:PermittedDates>
            <ekmi:PermittedDate>
                <ekmi:StartDate>2009-01-01</ekmi:StartDate>
                <ekmi:EndDate>2009-01-31</ekmi:EndDate>
            </ekmi:PermittedDate>
        </ekmi:PermittedDates>
    </ekmi:Permissions>
```

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

```
    <ekmi:Permissions>
        <ekmi:PermittedTransactions>100</ekmi:PermittedTransactions>
        <ekmi:PermittedUses>
            <ekmi:PermittedUse>CCN</ekmi:PermittedUse>
        </ekmi:PermittedUses>
    </ekmi:Permissions>
```

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

```
    <ekmi:Permissions>
        <ekmi:PermittedDuration>600</ekmi:PermittedDuration>
    </ekmi:Permissions>
```

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

```
    <ekmi:Permissions>
        <ekmi:PermittedDays>
            <ekmi:PermittedDay>Weekday</ekmi:PermittedDay>
        </ekmi:PermittedDays>
        <ekmi:PermittedLocations>
            <ekmi:PermittedLocation>
                <ekmi:LocationName>Facility A51</ekmi:LocationName>
                <ekmi:Latitude>37.385562</ekmi:Latitude>
                <ekmi:Longitude>-121.993387</ekmi:Latitude>
```

```
            </ekmi:PermittedLocation>
            <ekmi:PermittedLocation>
               <ekmi:LocationName>Facility DC-VA01</ekmi:LocationName>
               <ekmi:Latitude>88.485362</ekmi:Latitude>
               <ekmi:Longitude>-21.453648</ekmi:Latitude>
            </ekmi:PermittedLocation>
         </ekmi:PermittedLocations>
         <ekmi:PermittedLevels>
            <ekmi:PermittedLevel>Secret</ekmi:PermittedLevel>
         </ekmi:PermittedLevels>
         <ekmi:PermittedTimes>
            <ekmi:PermittedTime>
               <ekmi:StartTime>07:00:00</ekmi:StartTime>
               <ekmi:EndTime>19:00:00</ekmi:EndTime>
            </ekmi:PermittedTime>
         </ekmi:PermittedTimes>
      </ekmi:Permissions>
```

2.13 Element <PermittedApplications> and <PermittedApplication>

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

Schema Definition:

```
   <xsd:complexType name="PermittedApplicationsType">
        <xsd:sequence>
            <xsd:element
                name="PermittedApplication"
                type="tns:ApplicationsType"
                minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
```

Schema Definition:

```
    <xsd:complexType name="ApplicationsType">
        <xsd:sequence>
            <xsd:element name="ApplicationID" type="tns:TwoPartIDType"/>
            <xsd:element name="ApplicationName">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="256"/>
                        <xsd:whiteSpace value="preserve"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Version" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="32"/>
                        <xsd:whiteSpace value="preserve"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:group ref="tns:MessageDigestGroup" minOccurs="0"/>
            <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
```

```
        </xsd:sequence>
    </xsd:complexType>
```

Schema Definition:

```
   <xsd:group name="MessageDigestGroup">
      <xsd:sequence>
         <xsd:element name="DigestAlgorithm">
            <xsd:simpleType>
               <xsd:restriction base="xsd:anyURI">
                  <xsd:enumeration value="http://www.w3.org/2000/09/xmldsig#sha1"/>
                  <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#sha256"/>
                  <xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#sha512"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="DigestValue">
            <xsd:simpleType>
               <xsd:restriction base="xsd:base64Binary">
                  <xsd:maxLength value="1024"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
      </xsd:sequence>
   </xsd:group>
```

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

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

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

1. The <ApplicationID> element identifies the unique identifier assigned to this application within the SKMS. It is a ***TwoPartIDType*** as specified in Section 2.8. There SHALL be only one <ApplicationID> element within a <PermittedApplication> element.

2. The <ApplicationName> element identifies the name assigned to this application within the SKMS. It is an XSD ***String*** with a maximum length of 256 characters. There SHALL be only one <ApplicationName> element within a <PermittedApplication> element.

3. An optional <Version> element identifying the version number of this application within the SKMS. It is an XSD ***String*** with a maximum length of 32 characters. There MAY be only one <Version> element within a <PermittedApplication> element.

4. The <MessageDigestGroup> group which identifies the message digest of the application's binary image, along with the message digest algorithm used to calculate the digest value. The <MessageDigestGroup> consists of the following elements:

    a) The <DigestAlgorithm> element of the XSD type ***anyURI***, which supports one of the following three digest algorithms:

        i. http://www.w3.org/2000/09/xmldsig#sha1

        ii. http://www.w3.org/2001/04/xmlenc#sha256

        iii. http://www.w3.org/2001/04/xmlenc#sha512

    b) The <DigestValue> element of the XSD type ***base64Binary***.

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

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

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

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

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

```
<ekmi:PermittedApplications>
   <ekmi:PermittedApplication>
      <ekmi:ApplicationID>10514-24</ekmi:ApplicationID>
      <ekmi:ApplicationName>Employee Tax Reporting</ekmi:ApplicationName>
      <ekmi:Version>3.3</ekmi:Version>
      <ekmi:DigestAlgorithm>
            http://www.w3.org/2000/09/xmldsig#sha1
      </ekmi:DigestAlgorithm>
       <ekmi:DigestValue>G4bsdfKkt4cziEqFFuOoBTM81efU=</ekmi:DigestValue>
   </ekmi:PermittedApplication>
   <ekmi:PermittedApplication>
      <ekmi:ApplicationID>10514-25</ekmi:ApplicationID>
       <ekmi:ApplicationName>IRS Tax Reporting Application</ekmi:ApplicationName>
       <ekmi:Version>2.1</ekmi:Version>
       <ekmi:DigestAlgorithm>
            http://www.w3.org/2001/04/xmlenc#sha256
      </ekmi:DigestAlgorithm>
       <ekmi:DigestValue>
            ab7b85c9410d48c54fc7939c391be4028e7305085191c56e7b3740f2cbdbbc79
      </ekmi:DigestValue>
   </ekmi:PermittedApplication>
</ekmi:PermittedApplications>
```

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

```
<ekmi:PermittedApplications>
   <ekmi:PermittedApplication>
      <ekmi:ApplicationID>10514-14</ekmi:ApplicationID>
      <ekmi:ApplicationName>E-Commerce Payment</ekmi:ApplicationName>
   </ekmi:PermittedApplication>
</ekmi:PermittedApplications>
```

2.14 Element <PermittedDates> and <PermittedDate>

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

Schema Definition:

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

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

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

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

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

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

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

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

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

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

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

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

```
<ekmi:PermittedDates>
   <ekmi:PermittedDate>
      <ekmi:StartDate>2009-01-01</ekmi:StartDate>
       <ekmi:EndDate>2009-01-31</ekmi:EndDate>
   </ekmi:PermittedDate>
</ekmi:PermittedDates>
```

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

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

2.15 Element <PermittedDays> and <PermittedDay>

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

Schema Definition:

```
<xsd:complexType name="PermittedDaysType">
   <xsd:sequence>
      <xsd:element
            name="PermittedDay"
            type="tns:PermittedDayType"
            minOccurs="0"
            maxOccurs="unbounded">
      </xsd:element>
   </xsd:sequence>
 </xsd:complexType>
```

Schema Definition:

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

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

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

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

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

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

```
<ekmi:PermittedDays>
   <ekmi:PermittedDay>Weekday</ekmi:PermittedDay>
</ekmi:PermittedDays>
```

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

```
<ekmi:PermittedDays>
   <ekmi:PermittedDay>Monday</ekmi:PermittedDay>
   <ekmi:PermittedDay>Wednesday</ekmi:PermittedDay>
   <ekmi:PermittedDay>Friday</ekmi:PermittedDay>
</ekmi:PermittedDays>
```

2.16 Element <PermittedDuration>

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

Schema Definition:

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

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

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

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

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

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

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

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

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

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

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

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

2.17 Element <PermittedLevels> and <PermittedLevel>

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

Schema Definition:

```
<xsd:complexType name="PermittedLevelsType">
     <xsd:sequence>
         <xsd:element
                name="PermittedLevel"
                type="tns:LevelClassificationType"
                minOccurs="0"
                maxOccurs="unbounded">
         </xsd:element>
         <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
     </xsd:sequence>
</xsd:complexType>
```

Schema Definition:

```
<xsd:simpleType name="LevelClassificationType">
     <xsd:restriction base="xsd:string">
         <xsd:enumeration value="Unclassified"/>
         <xsd:enumeration value="Confidential"/>
         <xsd:enumeration value="Secret"/>
         <xsd:enumeration value="Top-Secret"/>
     </xsd:restriction>
 </xsd:simpleType>
```

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

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

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

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

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

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

```
<ekmi:PermittedLevels>
   <ekmi:PermittedLevel>Confidential</ekmi:PermittedLevel>
</ekmi:PermittedLevels>
```

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

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

2.18 Element <PermittedLocations> and <PermittedLocation>

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

Schema Definition:

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

Schema Definition:

```
    <xsd:group name="LocationCoordinateGroup">
         <xsd:sequence>
             <xsd:element name="Latitude">
                 <xsd:simpleType>
                     <xsd:restriction base="xsd:decimal">
                         <xsd:totalDigits value="10"/>
                         <xsd:fractionDigits value="7"/>
                     </xsd:restriction>
                 </xsd:simpleType>
             </xsd:element>
             <xsd:element name="Longitude">
                 <xsd:simpleType>
                     <xsd:restriction base="xsd:decimal">
                         <xsd:totalDigits value="10"/>
                         <xsd:fractionDigits value="7"/>
                     </xsd:restriction>
                 </xsd:simpleType>
             </xsd:element>
         </xsd:sequence>
     </xsd:group>
```

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

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

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

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

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

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

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

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

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

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

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

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

```
   <ekmi:PermittedLocations>
      <ekmi:PermittedLocation>
         <ekmi:LocationName>StrongAuth Server Room</ekmi:LocationName>
      </ekmi:PermittedLocation>
   </ekmi:PermittedLocations>
```

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

```
   <ekmi:PermittedLocations>
      <ekmi:PermittedLocation>
         <ekmi:LocationName>StrongAuth Server Room</ekmi:LocationName>
         <ekmi:Latitude>37.385653 </ekmi:Latitude>
         <ekmi:Longitude>-121.993192 </ekmi:Longitude>
      </ekmi:PermittedLocation>
   </ekmi:PermittedLocations>
```

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

```
   <ekmi:PermittedLocations>
      <ekmi:PermittedLocation>
         <ekmi:LocationName>Humongous Headquarters</ekmi:LocationName>
      </ekmi:PermittedLocation>
      <ekmi:PermittedLocation>
         <ekmi:LocationName> Humongous Primary Data Center</ekmi:LocationName>
         <ekmi:Latitude>37.385653 </ekmi:Latitude>
         <ekmi:Longitude>-121.993192 </ekmi:Longitude>
      </ekmi:PermittedLocation>
      <ekmi:PermittedLocation>
         <ekmi:LocationName>Humongous DR Data Center</ekmi:LocationName>
         <ekmi:Latitude>68.845901 </ekmi:Latitude>
         <ekmi:Longitude>11.393385 </ekmi:Longitude>
      </ekmi:PermittedLocation>
   </ekmi:PermittedLocations>
```

2.19 Element <PermittedTimes> and <PermittedTime>

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

Schema Definition:

```
   <xsd:complexType name="PermittedTimesType">
      <xsd:sequence>
         <xsd:element name="PermittedTime" minOccurs="0" maxOccurs="unbounded">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="StartTime">
                     <xsd:simpleType>
                        <xsd:restriction base="xsd:time">
                           <xsd:pattern value="\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}"/>
                        </xsd:restriction>
                     </xsd:simpleType>
                  </xsd:element>
                  <xsd:element name="EndTime">
                     <xsd:simpleType>
                        <xsd:restriction base="xsd:time">
                           <xsd:pattern value="\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}"/>
```

```
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

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

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

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

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

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

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

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

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

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

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

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

```
<ekmi:PermittedTimes>
   <ekmi:PermittedTime>
      <ekmi:StartTime>09:00:00</ekmi:StartTime>
       <ekmi:EndTime>17:00:00</ekmi:EndTime>
   </ekmi:PermittedTime>
</ekmi:PermittedTimes>
```

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

```
<ekmi:PermittedTimes>
   <ekmi:PermittedTime>
```

```
            <ekmi:StartTime>06:00:00</ekmi:StartTime>
            <ekmi:EndTime>12:00:00</ekmi:EndTime>
         </ekmi:PermittedTime>
         <ekmi:PermittedTime>
            <ekmi:StartTime>15:00:00</ekmi:StartTime>
            <ekmi:EndTime>19:00:00</ekmi:EndTime>
         </ekmi:PermittedTime>
      </ekmi:PermittedTimes>
```

2.20 Element <PermittedTransactions>

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

Schema Definition:

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

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

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

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

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

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

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

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

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

2.21 Element <PermittedUses> and <PermittedUse>

Schema Definition:

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

```
   <xsd:complexType name="PermittedUsesType" mixed="true">
      <xsd:sequence>
         <xsd:element name="PermittedUse" minOccurs="0" maxOccurs="unbounded">
            <xsd:simpleType>
               <xsd:restriction base="xsd:string">
```

```
                    <xsd:maxLength value="256"/>
                    <xsd:whiteSpace value="preserve"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
        <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
```

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

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

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

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

```
<ekmi:PermittedUses>
    <ekmi:PermittedUse>VPN</ekmi:PermittedUse>
</ekmi:PermittedUses>
```

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

```
<ekmi:PermittedUses>
    <ekmi:PermittedUse>Laptop</ekmi:PermittedUse>
    <ekmi:PermittedUse>PDA</ekmi:PermittedUse>
</ekmi:PermittedUses>
```

2.22 Element <KeyCachePolicyRequest>

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

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

Schema Definition:

```
<xsd:element name="KeyCachePolicyRequest">
    <xsd:complexType>
        <xsd:annotation>
            <xsd:documentation>
                No elements/attributes are defined for KeyCachePolicyRequest.
            </xsd:documentation>
        </xsd:annotation>
    </xsd:complexType>
</xsd:element>
```

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

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

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

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

2.23 Element <KeyCachePolicyResponse>

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

Schema Definition:

```
<xsd:element name="KeyCachePolicyResponse">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element
                name="KeyCachePolicy"
                type="ekmi:KeyCachePolicyType"
                minOccurs="1" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
 </xsd:element>
```

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

2.24 Element <KeyCachePolicy>

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

Schema Definition:

```
<xsd:element name="KeyCachePolicyResponse">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element
                name="KeyCachePolicy"
                type="ekmi:KeyCachePolicyType"
                minOccurs="1" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
 </xsd:element>

<xsd:complexType name="KeyCachePolicyType" mixed="true">
    <xsd:sequence>
        <xsd:element name="KeyCachePolicyID" type="tns:TwoPartIDType"/>
        <xsd:element name="PolicyName">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="255"/>
```

```
                        <xsd:whiteSpace value="preserve"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Description" nillable="true">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="2048"/>
                        <xsd:whiteSpace value="preserve"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="KeyClass" type="tns:KeyClassType"/>
            <xsd:element name="StartDate" type="xsd:dateTime"/>
            <xsd:element name="EndDate" type="xsd:dateTime" nillable="true"/>
            <xsd:element name="PolicyCheckInterval">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:nonNegativeInteger">
                        <xsd:minInclusive value="0"/>
                        <xsd:maxInclusive value="2592000"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Status" type="tns:StatusType"/>
            <xsd:element
                name="NewKeysCacheDetail"
                type="tns:KeyCacheDetailType"
                minOccurs="0"/>
            <xsd:element
                name="UsedKeysCacheDetail"
                type="tns:KeyCacheDetailType"
                minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
```

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

1. <KeyCachePolicyID> [Required]

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

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

2. <PolicyName> [Required]

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

3. <Description> [Required]

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

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

4. <KeyClass> [Required]

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

5. <StartDate> [Required]

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

6. <EndDate> [Required]

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

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

7. <PolicyCheckInterval> [Required]

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

8. <Status> [Required]

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

   The ***StatusType*** is specified in Section 2.11.

9. <NewKeysCacheDetail> [Required]

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

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

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

   The ***KeyCacheDetailType*** is specified in Section 2.22.

10. <UsedKeysCacheDetail> [Required]

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

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

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

    The ***KeyCacheDetailType*** is specified in Section 2.22.

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

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

```
<ekmi:KeyCachePolicy>
    <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
    <ekmi:PolicyName>
        Corporate Laptop Symmetric Key Caching Policy
    </ekmi:PolicyName>
    <ekmi:Description>
        This policy defines how company-issued laptops will manage
        symmetric keys used for file/disk encryption in their local
        cache.  This policy must be used by all laptops that use
        the company EKMI.
    </ekmi:Description>
    <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
    <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
    <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
    <ekmi:Status>Active</ekmi:Status>
    <ekmi:NewKeysCacheDetail>
        <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
        <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
    </ekmi:NewKeysCacheDetail>
    <ekmi:UsedKeysCacheDetail>
        <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
        <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
    </ekmi:UsedKeysCacheDetail>
</ekmi:KeyCachePolicy>
```

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

```
<ekmi:KeyCachePolicy>
    <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
    <ekmi:PolicyName>
        No Caching Policy
    </ekmi:PolicyName>
    <ekmi:Description>
        This policy is for high-risk, always-connected machines on the
        network, which will never cache symmetric keys locally.  This
        policy never expires (but checks monthly for any updates).
    </ekmi:Description>
    <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
    <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
    <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
    <ekmi:Status>Active</ekmi:Status>
</ekmi:KeyCachePolicy>
```

2.25 Type *KeyCacheDetailType*

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

Schema Definition:

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

```
                <xsd:simpleType>
                    <xsd:restriction base="xsd:integer">
                        <xsd:minInclusive value="0"/>
                        <xsd:maxInclusive value="18446744073709551615"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="MaximumDuration" minOccurs="1">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:integer">
                        <xsd:minInclusive value="0"/>
                        <xsd:maxInclusive value="18446744073709551615"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
```

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

1. <MaximumKeys> [Required]

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

2. <MaximumDuration> [Required]

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

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

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

```
<ekmi:KeyCachePolicy>
    <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
    <ekmi:PolicyName>
        Corporate Laptop Symmetric Key Caching Policy
    </ekmi:PolicyName>
    <ekmi:Description>
        This policy defines how company-issued laptops will manage
        symmetric keys used for file/disk encryption in their local
        cache.  This policy must be used by all laptops that use
        the company EKMI.
    </ekmi:Description>
    <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
    <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
    <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
    <ekmi:Status>Active</ekmi:Status>
    <ekmi:NewKeysCacheDetail>
        <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
        <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
    </ekmi:NewKeysCacheDetail>
    <ekmi:UsedKeysCacheDetail>
        <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
```

```
            <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
        </ekmi:UsedKeysCacheDetail>
    </ekmi:KeyCachePolicy>
```

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

```
    <ekmi:KeyCachePolicy>
        <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
        <ekmi:PolicyName>
            Corporate Laptop Symmetric Key Caching Policy
        </ekmi:PolicyName>
        <ekmi:Description>
            This policy defines how company-issued laptops will manage
            symmetric keys used for file/disk encryption in their local
            cache.  This policy must be used by all laptops that use
            the company EKMI.
        </ekmi:Description>
        <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
        <ekmi:EndDate>2008-12-31T24:00:00.0</ekmi:EndDate>
        <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
        <ekmi:Status>Active</ekmi:Status>
        <ekmi:NewKeysCacheDetail>
            <ekmi:MaximumKeys>1</ekmi:MaximumKeys>
            <ekmi:MaximumDuration>1296000</ekmi:MaximumDuration>
        </ekmi:NewKeysCacheDetail>
        <ekmi:UsedKeysCacheDetail>
            <ekmi:MaximumKeys>0</ekmi:MaximumKeys>
            <ekmi:MaximumDuration>1296000</ekmi:MaximumDuration>
        </ekmi:UsedKeysCacheDetail>
    </ekmi:KeyCachePolicy>
```

Appendix A. Acknowledgments

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

Participants:

-

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.

Appendix C. Non-Normative Text

Appendix D. SKSML Error Codes and Error Messages