Symmetric Key Services Markup Language (SKSML) Version 1.0 Normative DRAFT 6.0]

OASIS Technical Committee DRAFT

24 June 2008

Specification URIs:

This Version:
http://docs.oasis-open.org/ekmi/1.0/SKSML-1.0-Specification.html
http://docs.oasis-open.org/ekmi/1.0/SKSML-1.0-Specification.odt
http://docs.oasis-open.org/ekmi/1.0/SKSML-1.0-Specification.pdf

Previous Version:
None

Latest Version:
Same as “This Version”

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:
• None

This specification is related to:

Web Services Security - SOAP Message Security 1.0 - OASIS Standard 200401, March 2004 -

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

Technical Committee members should send comments on this specification to the Technical
Committee's email list. Others should send comments to the Technical Committee by using the "Send A

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

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

1 Introduction.................................................................................................................. 5
  1.1 Terminology.............................................................................................................. 5
  1.2 Glossary.................................................................................................................... 5
  1.3 Normative References............................................................................................. 6
2 Specification.................................................................................................................. 7
  2.1 Element <SymkeyRequest>.................................................................................... 7
  2.2 Element <GlobalKeyID>......................................................................................... 9
  2.3 Element <KeyClasses> and <KeyClass>................................................................. 10
  2.4 Element <SymkeyResponse>.................................................................................. 12
  2.5 Element <Symkey>............................................................................................... 13
  2.6 Element <SymkeyError>....................................................................................... 15
  2.7 Element <KeyUsePolicy>....................................................................................... 17
  2.8 Type TwoPartIDType............................................................................................. 19
  2.9 Element <KeyAlgorithm>...................................................................................... 20
  2.10 Element <KeySize>............................................................................................. 21
  2.11 Element <Status>............................................................................................... 22
  2.12 Element <Permissions>....................................................................................... 23
  2.13 Element <PermittedApplications> and <PermittedApplication>......................... 29
  2.14 Element <PermittedDates> and <PermittedDate>................................................. 32
  2.15 Element <PermittedDays> and <PermittedDay>.................................................... 34
  2.16 Element <PermittedDuration>............................................................................. 35
  2.17 Element <PermittedLevels> and <PermittedLevel>............................................. 36
  2.18 Element <PermittedLocations> and <PermittedLocation>................................. 38
  2.19 Element <PermittedNumberOfTransactions>...................................................... 40
  2.20 Element <PermittedTimes> and <PermittedTime>.............................................. 41
  2.21 Element <PermittedUses> and <PermittedUse>.................................................. 43
  2.22 Element <KeyCachePolicyRequest>................................................................. 44
  2.23 Element <KeyCachePolicyResponse>............................................................... 45
  2.24 Element <KeyCachePolicy>............................................................................... 45
  2.25 Type KeyCacheDetailType.................................................................................... 48
1 Introduction

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

1.1 Terminology

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

1.2 Glossary

3DES – Triple Data Encryption Standard

AES – Advanced Encryption Standard

Base64 – An encoding scheme for representing data

Ciphertext – Encrypted data

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

DES – Data Encryption Standard

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

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

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

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

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

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

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

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

Plaintext – Unencrypted data

SHA – Secure Hashing Algorithm

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

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

SHA-384 – Secure Hashing Algorithm with a resultant size of 384-bits
SHA-512 – Secure Hashing Algorithm with a resultant 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 a network or other mechanism selected by implementers

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

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

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

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

1.3 Normative References

[AES] Advanced Encryption Standard


[SOAP] Simple Object Access Protocol 1.1

[XMLEncryption] XML Encryption Syntax and Processing

[XMLSignature] XML Signature Syntax and Processing

OASIS Standard 200401, March 2004

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:

```xml
<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 (unlimited) 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 (unlimited) 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 is required to be of a different
There **MUST** be only one `<GlobalKeyID>` element followed by as many `<KeyClass>` elements inside the `<KeyClasses>` set, as needed by the client application.

When a client requires multiple 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 with 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:**

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

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

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

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

3. A positive integer identifying the Server ID of the server that originally generated the key;

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

5. A positive 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 positive integer in the range of 0 (zero) to 18446744073709551615 (20-byte ASCII decimal).

When an SKMS manages the symmetric keys for a single enterprise, the DomainID part of the <GlobalKeyID> element in a <SymkeyRequest> MAY be zero ("0"). When an SKMS manages symmetric keys for multiple enterprises, the DomainID in the <GlobalKeyID> of a <SymkeyRequest> MUST be positive 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 positive 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 positive 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 positive and non-zero. It will typically contain the unique server identifier of the SKS server where the symmetric key was generated.

The KeyID part of the <GlobalKeyID> element MUST be a positive 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 positive 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:

```xml
<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:

```xml
<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>`:

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

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

```xml
<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 (unlimited) number of `<KeyClass>` elements within the `<KeyClasses>` set.

#### Schema Definition:

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

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 inside the <KeyClasses> set as needed by the client application. (Example 5 in this section).

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

Example 1 – A symmetric key request of a default key class (when no KeyClass is specified):
<ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
  <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
</ekmi:SymkeyRequest>

Example 2 – A request for multiple new symmetric keys, each of a default key class:
<ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
  <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
  <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
</ekmi:SymkeyRequest>

Example 3 – A request for a new symmetric key of a specific key class:
<ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
  <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
  <ekmi:KeyClasses>
    <ekmi:KeyClass>256-Bit-Class</ekmi:KeyClass>
  </ekmi:KeyClasses>
</ekmi:SymkeyRequest>

Example 4 – A request for two new symmetric keys of the same key class for each symmetric key. Note that if the FIN-FX key class was the default key class, a request as shown in Example 2 of this section would result in the same response:
<ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
  <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
  <ekmi: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:

```xml
<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 response with a single symmetric key:**

```xml
<ekmi:SymkeyResponse
 xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
```
Example 2 – A response with three symmetric keys:

```xml
<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:

```xml
<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:

```xml
<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 response with one symmetric key and one error:

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

Example 6 – A response with multiple symmetric keys and multiple error:

```xml
<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. The 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:**

```xml
<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: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.

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

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

```xml
<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>
       E9zwW/y93hVSeTIcDoDxmlNxtu+xSffMNwCJmt1dIg2QHbnpdO8
       1g6DKdKCFjJkhqvywC9sfYv9h5DqUiOQXG0ca8EU871zBoXbj)xj
       fg1pU88gfBpWzcd/ATpJD/UJow/qimxi8+huUYJMaGH=
     </xenc:CipherValue>
   </xenc:CipherData>
</ekmi:Symkey>
</ekmi:SymkeyResponse>
```
Example 2 – A response with multiple symmetric keys:

```xml
<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+SffMNwCJmt1dIgzQHBnpdQ8
    lg6DKdKCFjMhQhywC9sYjv9h5FDqU5QXG0ca8EU871zBoXBjDxj
    fg1pU8tGFbWZCd/ATpJD/UJow/qimxi8+huUYJMaGH=
   </xenc:CipherValue>
  </xenc:CipherData>
 <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+SffMNwCJmt1dIgzQHBnpdQ8
    7k6DKdKCFjMhQhywC9sYjv9h5FDqU5QXG0ca8EU871zBoXBjDxj
    uycU8tGFbWZCd/ATpJD/UJow/qimxi8+huUYJMaGH=
   </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.

```
<xs:complexType name="SymkeyErrorType">
  <xs:sequence>
    <xs:element name="RequestedGlobalKeyID" type="ekmi:GlobalKeyIDType"/>
    <xs:element name="RequestedKeyClass" type="ekmi:KeyClassType" minOccurs="0"/>
    <xs:element name="ErrorCode" type="xs:simpleType">
      <xs:restriction base="xs:string">
        <xs:maxLength value="255"/>
      </xs:restriction>
    </xs:element>
    <xs:element name="ErrorMessage" type="xs:simpleType">
      <xs:restriction base="xs:string">
        <xs:maxLength value="1024"/>
      </xs:restriction>
    </xs:element>
  </xs:sequence>
</xs: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 the 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 the SKS Server experienced in processing the client's symmetric key request.
   The <ErrorMessage> element SHALL return the appropriate localized version of the message corresponding to the <ErrorCode> element from Appendix 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:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>
   <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
</ekmi:SymkeyResponse>

Example 2 – Multiple errors within a response:
<ekmi:SymkeyResponse
   xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
   <ekmi:SymkeyError>
   </ekmi:SymkeyError>
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:**

```xml
c<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 it 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 permission restrictions:

```
<ekmi:KeyUsePolicy>
  <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>
  <ekmi:PolicyName>DES-EDE KeyUsePolicy</ekmi:PolicyName>
  <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
  <ekmi:KeyAlgorithm>
    http://www.w3.org/2001/04/xmlenc#tripledes-cbc
  </ekmi:KeyAlgorithm>
  <ekmi:KeySize>192</ekmi:KeySize>
  <ekmi:Status>Active</ekmi:Status>
  <ekmi:Permissions>
    <ekmi:PermittedApplications ekmi:any="false"/>
    <ekmi:PermittedApplication>
      <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
      <ekmi:ApplicationName>Payroll Application</ekmi:ApplicationName>
    </ekmi:PermittedApplication>
  </ekmi:Permissions>
</ekmi:KeyUsePolicy>
```
Example 2 – A `<KeyUsePolicy>` with no restrictions on the key:

```
<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:PermittedApplications ekmi:any="true" xsi:nil="true"/>
    <ekmi:PermittedDates ekmi:any="false"/>
    <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
    <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
    <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
    <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
    <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
    <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
    <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
  </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:

```xml
<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:**

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

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

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

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

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

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

```xml
<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:

```xml
<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: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:**

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

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

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

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

```xml
<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:**

```xml
<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 exists within an element, 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:**

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

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

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

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

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

### 2.12 Element `<Permissions>`

The `<Permissions>` element, of the 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 of 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:**

```xml
<xsd:complexType name="PermissionsType">
  <xsd:sequence>
    <xsd:element name="PermittedApplications" type="tns:PermittedApplicationsType" minOccurs="1" nillable="true"/>
    <xsd:element name="PermittedDates" type="tns:PermittedDatesType"/>
  </xsd:sequence>
</xsd:complexType>
```
The `<Permissions>` element consists of the following sub-elements:

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

   The `<PermittedApplications>` element is specified in Section 2.13.

2. A required `<PermittedDates>` element which identifies the date ranges during which applications are permitted use of the symmetric key in question. While the `<PermittedDates>` element is required, it may be empty (NULL). The absence of sub-elements in the `<PermittedDates>` element implies that applications are permitted to use the key on any date. Identifying specific date ranges restricts the use of the key to only the duration between the identified dates.
The <PermittedDates> element is specified in Section 2.12.

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

The <PermittedDays> element is specified in Section 2.15.

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

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

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

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

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

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

The <PermittedLocations> element is specified in Section 2.18.

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

The <PermittedNumberOfTransactions> element is specified in Section 2.19.

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

The <PermittedTimes> element is specified in Section 2.20.
9. A required `<PermittedUses>` element which identifies application-uses that applications are permitted with the symmetric key in question. While the `<PermittedUses>` element is required, it may be empty (NULL). The absence of sub-elements in the `<PermittedUses>` element implies that applications are permitted to use the key for any purpose. Identifying specific uses restricts the use of the key to only the identified uses.

The `<PermittedUses>` element is specified in Section 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 other SKMS implementations that use the SKSML standard.

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

When all sub-elements of the `<Permissions>` element are empty, there are no restrictions on the use of the symmetric key other than that the application calling on the SKCL be authorized to access the key in question.

However, when there are elements defined within the sub-elements of the `<Permissions>` element, conforming SKCL implementations must comply with all the permission elements, evaluating the most restrictive permissions first and in decreasing order of restriction, before allowing the use of the key.

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

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

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

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

**Example 1 – A `<Permissions>` element that permits a single application the use of the symmetric key in question, between January 01, 2008 and December 31, 2008 and between the hours of 0700 and 1900.**

There are, however, no restrictions on what days of the week the key may be used, the locations where it may be used, at which MLS level it may be used (if it applies), the number of data files/transactions that may be encrypted with the key or the uses of the key within that application:

```xml
<ekmi:Permissions>
  <ekmi:PermittedApplications ekmi:any="false">
    <ekmi:PermittedApplication>
      <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
      <ekmi:ApplicationName>Payroll Application</ekmi:ApplicationName>
      <ekmi:Version>1.0</ekmi:Version>
      <ekmi:DigestAlgorithm>
        http://www.w3.org/2000/09/xmldsig#sha1
      </ekmi:DigestAlgorithm>
    </ekmi:PermittedApplication>
  </ekmi:PermittedApplications>
</ekmi:Permissions>
```
Example 2 – A `<Permissions>` element 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:any="false">
    <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>
        af96d65a7a2415239c8eb8be1347f704322957a4
      </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>
        a4f5925185ffe12c1a91ea3de90fc086b34b34b2
      </ekmi:DigestValue>
    </ekmi:PermittedApplication>
  </ekmi:PermittedApplications>
  <ekmi:PermittedDates ekmi:any="false">
    <ekmi:PermittedDate>
      <ekmi:StartDate>2009-01-01</ekmi:StartDate>
      <ekmi:EndDate>2009-01-31</ekmi:EndDate>
    </ekmi:PermittedDate>
  </ekmi:PermittedDates>
</ekmi:Permissions>
```
Example 3 – A <Permissions> element that permits all applications the use of the symmetric key in
question, for 100 transactions for encrypting credit card numbers.

<ekmi:Permissions>
  <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedNumberOfTransactions ekmi:any="false">100</ekmi:PermittedNumberOfTransactions>
  <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedUses ekmi:any="false">
    <ekmi:PermittedUse>CCN</ekmi:PermittedUse>
  </ekmi:PermittedUses>
</ekmi:Permissions>

Example 4 – A <Permissions> element 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:PermittedApplications ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedDuration ekmi:any="false">600</ekmi:PermittedDuration>
  <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
</ekmi:Permissions>

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

<ekmi:Permissions>
  <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedDays ekmi:any="false">
    <ekmi:PermittedDay>Weekday</ekmi:PermittedDay>
  </ekmi:PermittedDays>
  <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
  <ekmi:PermittedLevels ekmi:any="false">
    <ekmi:PermittedLevel>Secret</ekmi:PermittedLevel>
  </ekmi:PermittedLevels>
  <ekmi:PermittedLocations ekmi:any="false"/>
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:**

```xml
<xsd:complexType name="PermittedApplicationsType">
  <xsd:sequence>
    <xsd:element name="PermittedApplication" type="tns:ApplicationsType" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute ref="tns:any" use="required"/>
</xsd:complexType>
```

**Schema Definition:**

```xml
<xsd:complexType name="ApplicationsType">
  <xsd:sequence>
    <xsd:element name="ApplicationID" type="tns:TwoPartIDType"/>
    <xsd:element name="ApplicationName" type="xsd:string" minOccurs="0" maxOccurs="unbounded">
      <xsd:restriction base="xsd:string">
        <xsd:maxLength value="256"/>
        <xsd:WhiteSpace value="preserve"/>
      </xsd:restriction>
    </xsd:element>
    <xsd:element name="Version" minOccurs="0" maxOccurs="unbounded">
      <xsd:restriction base="xsd:string">
        <xsd:maxLength value="32"/>
        <xsd:WhiteSpace value="preserve"/>
      </xsd:restriction>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```
There SHALL be only one <PermittedApplications> element within the <Permissions> element.

However, there MAY be an unbounded (unlimited) number of <PermittedApplication> elements within a <PermittedApplications> element.

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

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

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

The <PermittedApplication> sub-element of type ApplicationsType, provides details of the application which is permitted use of 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 value of the application's binary image, along with the message digest algorithm used to calculate the digest value. The `<MessageDigestGroup>` consists of the following elements:
   a) The `<DigestAlgorithm>` element of the XSD type `anyURI`, which supports one of the following three digest algorithms:
      i. http://www.w3.org/2000/09/xmldsig#sha1
      ii. http://www.w3.org/2001/04/xmlenc#sha256
      iii. http://www.w3.org/2001/04/xmlenc#sha512
   b) The `<DigestValue>` element of the XSD type `base64Binary`.

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

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

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

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

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

```xml
<ekmi:PermittedApplications ekmi:any="false">
  <ekmi:PermittedApplication>
    <ekmi:ApplicationID>10514-24</ekmi:ApplicationID>
    <ekmi:ApplicationName>Employee Tax Reporting</ekmi:ApplicationName>
    <ekmi:Version>3.3</ekmi:Version>
    <ekmi:DigestAlgorithm>
      http://www.w3.org/2000/09/xmldsig#sha1
    </ekmi:DigestAlgorithm>
    <ekmi:DigestValue>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:PermittedApplication>
</ekmi:PermittedApplications>
```
Example 2 – An example of a `<PermittedApplications>` element with one child `<PermittedApplication>` element that applies to all versions of the application; the message digest value and algorithm are not used in this example:

```xml
<ekmi:PermittedApplications ekmi:any="false">
  <ekmi:PermittedApplication>
    <ekmi:ApplicationID>10514-14</ekmi:ApplicationID>
    <ekmi:ApplicationName>E-Commerce Payment</ekmi:ApplicationName>
  </ekmi:PermittedApplication>
</ekmi:PermittedApplications>
```

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

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

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 ranges of dates between which applications are permitted to use the symmetric key within a specific `<Symkey>` element.

Schema Definition:

```xml
<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>
```

There SHALL be only one `<PermittedDates>` element within the `<Permissions>` element. However, there MAY be an unbounded number of `<PermittedDate>` elements within a `<PermittedDates>` element.
The `<PermittedDates>` element SHALL have one attribute named “any”, that will have a “false” or “true” value, based on the following:

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

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

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

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.

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

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

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

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

```
<ekmi:PermittedDates ekmi:any="false">
  <ekmi:PermittedDate>
    <ekmi:StartDate>2008-07-01</ekmi:StartDate>
    <ekmi:EndDate>2008-07-03</ekmi:EndDate>
  </ekmi:PermittedDate>
  <ekmi:PermittedDate>
    <ekmi:StartDate>2008-07-07</ekmi:StartDate>
    <ekmi:EndDate>2008-07-12</ekmi:EndDate>
  </ekmi:PermittedDate>
</ekmi:PermittedDates>
```
Example 3 – An example of a null `<PermittedDates>` element, specifying that applications are permitted use of the symmetric key on any date:

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

### 2.15 Element `<PermittedDays>` and `<PermittedDay>`

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

**Schema Definition:**

#### `<PermittedDaysType>`

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

#### `<PermittedDayType>`

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

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

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

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

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

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

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

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

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

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

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

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

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

2.16 Element `<PermittedDuration>`

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

**Schema Definition:**

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

There SHALL be only one `<PermittedDuration>` element within the `<Permissions>` element.
The `<PermittedDuration>` element SHALL have one attribute, named “any” that will have a “false” or “true” value, based on the following:

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

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

The `<PermittedDuration>` element, of the XSD `positiveInteger` type, identifies the 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. The values for `<PermittedDuration>` may range between 1 and 18446744073709551615.

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.

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

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

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

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

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

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

The element `<PermittedLevels>`, of the type `LevelClassificationType`, is used to define the security level at which applications are permitted use of a symmetric key. 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:**

```xml
<xsd:complexType name="PermittedLevelsType">
  <xsd:sequence>
    <xsd:element name="PermittedLevel"/>
  </xsd:sequence>
</xsd:complexType>
```

---

24. June 2008

Copyright © OASIS® 2008. All Rights Reserved.

Page 36 of 54
There SHALL be only one `<PermittedLevels>` element within the `<Permissions>` element. However, there MAY be an unbounded (unlimited) number of `<PermittedLevel>` elements within the `<PermittedLevels>` element. (Practically, it makes no sense to have more than the known levels; however, this specification leaves itself open to the possibility that other levels may be defined).

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

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

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

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

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

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

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

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

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

```xml
<ekmi:PermittedLevels>
  <ekmi:PermittedLevel>Secret</ekmi:PermittedLevel>
  <ekmi:PermittedLevel>Top-Secret</ekmi:PermittedLevel>
</ekmi:PermittedLevels>
```
Example 3 – An example of a null `<ekmi:PermittedLevels>` element specifying that the symmetric key may be used at any level:

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

2.18 Element `<PermittedLocations>` and `<PermittedLocation>`

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

**Schema Definition:**

```
<xsd:complexType name="PermittedLocationsType">
  <xsd:sequence>
    <xsd:element name="PermittedLocation" minOccurs="1" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="LocationName">
            <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                <xsd:maxLength value="256"/>
                <xsd:whiteSpace value="preserve"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:group ref="tns:LocationCoordinateGroup" minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
          </xsd:group>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute ref="tns:any" use="required"/>
</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:restriction>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
</xsd:group>
```
There SHALL be only one `<PermittedLocations>` element within the `<Permissions>` element. However, there MAY be an unbounded (unlimited) number of `<PermittedLocation>` sub-elements within the `<PermittedLocations>` element.

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

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

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

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:**

```xml
<ekmi:PermittedLocations ekmi:any="false">
    <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:

```xml
<ekmi:PermittedLocations ekmi:any="false">
  <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:

```xml
<ekmi:PermittedLocations ekmi:any="false">
  <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>
```

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

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

2.19 Element `<PermittedNumberOfTransactions>`

The element `<PermittedNumberOfTransactions>`, of type `PermittedNumberOfTransactionsType` 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:**

```xml
<xsd:complexType name="PermittedNumberOfTransactionsType">
  <xsd:simpleContent>
    <xsd:extension base="tns:NumberOfTransactionsType">
      <xsd:attribute ref="tns:any" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

**Schema Definition:**

```xml
<xsd:simpleType name="NumberOfTransactionsType">
  <xsd:restriction base="xsd:positiveInteger">
    <xsd:minInclusive value="1"/>
    <xsd:maxInclusive value="18446744073709551615"/>
  </xsd:restriction>
</xsd:simpleType>
```
There SHALL be only one `<PermittedNumberOfTransactions>` element within the `<Permissions>` element.

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

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

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

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

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

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

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

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

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

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

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

### 2.20 Element `<PermittedTimes>` and `<PermittedTime>`

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

**Schema Definition:**

```
<xs:complexType name="PermittedTimesType">
  <xs:sequence>
    <xs:element name="PermittedTime" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="StartTime">
            <xs:simpleType>
              <xs:restriction base="xs:time">
                <xs:pattern value="\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element name="EndTime">
            <xs:simpleType>
              <xs:restriction base="xs:time">
                <xs:pattern value="\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```
There SHALL be only one `<PermittedTimes>` element within the `<Permissions>` element. However, there MAY be an unbounded (unlimited) number of `<PermittedTime>` sub-elements within a `<PermittedTimes>` element.

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

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

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

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

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

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

   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:
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:

```xml
<ekmi:PermittedTimes ekmi:any="false">
  <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>
```

Example 3 – An example of a null `<PermittedTimes>` element, specifying that the key may be used at any time:

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

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

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

**Schema Definition:**

```xml
<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:attribute ref="tns:any" use="required"/>
</xsd:complexType>
```

There SHALL be only one `<PermittedUses>` element within the `<Permissions>` element. However, there MAY be an unbounded (unlimited) number of `<PermittedUse>` sub-elements within the `<PermittedUses>` element.

The `<PermittedUses>` element SHALL have one attribute named “any”, that will have a “false” or “true” value, based on the following:
• When the `<PermittedUses>` element is null (i.e. it does not have a single `<PermittedUse>` sub-element in it), the value of the "any" attribute SHALL be set to "true" AND the XML Schema Instance (XSI) "nil" attribute SHALL be set to "true".

• When the `<PermittedUses>` element is not-null (i.e. it has at least one `<PermittedUse>` sub-element in it), the value of the "any" attribute SHALL be set to "false" AND the XML Schema Instance (XSI) "nil" attribute SHALL NOT be present.

A null `<PermittedUses>` element specifies that applications are permitted use of the symmetric key for ANY purpose, subject to complying with all other permission clauses in the `<Permissions>` element.

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:

```xml
<ekmi:PermittedUses ekmi:any="false">
  <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):

```xml
<ekmi:PermittedUses ekmi:any="false">
  <ekmi:PermittedUse>Laptop</ekmi:PermittedUse>
  <ekmi:PermittedUse>PDA</ekmi:PermittedUse>
</ekmi:PermittedUses>
```

Example 3 – An example of a null `<PermittedUses>` element specifying that the symmetric key may be used for any purpose:

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

### 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:**

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

The `<KeyCachePolicyRequest>` has no child elements. The SOAP Header of the signed request provides the SKS server with all the information it needs to process the request: the identity of the requester, strong authentication and message integrity of the request.
Some examples of the use of the <SymkeyRequest> element are as follows:

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

```xml
<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:

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

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

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:

```xml
<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>
```

```xml
<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: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 `<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 `<Description>` 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 (as yet unused for any encryption transaction) symmetric keys a client may cache, and for how long. It is the responsibility of the conforming SKCL implementation to enforce these rules.

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

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

The KeyCacheDetailType is specified in Section 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 up to 90 days:

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

```xml
<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:

```xml
<xsd:complexType name="KeyCacheDetailType">
  <xsd:sequence>
    <xsd:element name="MaximumKeys" minOccurs="1">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer"/>
      </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 up to 90 days:

```xml
<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 up to 15 days:

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

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAFT 4</td>
<td>June 08, 2008</td>
<td>Arshad Noor</td>
<td>Initial version</td>
</tr>
<tr>
<td>DRAFT 5</td>
<td>June 17, 2008</td>
<td>Arshad Noor</td>
<td>Moved non-normative sections to their own document. KeyClass element was added to KeyCachePolicy. KeyCachePolicy is now embedded inside a KeyCachePolicyResponse.</td>
</tr>
</tbody>
</table>
Appendix C. Non-Normative Text