Symmetric Key Services Markup Language (SKSML) Version 1.0

Public Review Draft 03

5 August 2010

Specification URIs:

This Version:

http://docs.oasis-open.org/ekmi/sksml/v1.0/pr03/SKSML-1.0-Specification.html

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

http://docs.oasis-open.org/ekmi/sksml/v1.0/pr03/SKSML-1.0-Specification.pdf

Previous Version:

http://docs.oasis-open.org/ekmi/sksml/v1.0/cs01/SKSML-1.0-Specification.html

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

http://docs.oasis-open.org/ekmi/sksml/v1.0/cs01/SKSML-1.0-Specification.pdf

Latest Version:

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

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

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

Technical Committee:

OASIS Enterprise Key Management Infrastructure (EKMI) TC

Chair(s):

Anil Saldhana, Red Hat Inc (anil.saldhana@redhat.com)

Timothy Bruce, CA Inc (Timothy.Bruce@ca.com)

Editor(s):

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

Related Work:

This specification replaces or supercedes:

- None

This specification is related to:

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

- Simple Object Access Protocol (SOAP) - W3C Recommendation 08 May 2000. http://www.w3.org/TR/soap/
- XML Encryption - W3C Recommendation 10 Dec 2002. http://www.w3.org/TR/xmlenc-core/
- XML Signature - W3C Recommendation 10 Jun 2008. http://www.w3.org/TR/xmldsig-core/
- Web Services Security - SOAP Message Security 1.0 - OASIS Standard 200401, March 2004 - http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

Declared XML Namespace(s):

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

Abstract:

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

Status:

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

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

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

Notices

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

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

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

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

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

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

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

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

Table of Contents

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

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

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.NIST FIPS 197. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[RFC 2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997.

[SOAP] SOAP v1.2 Specification. W3C Recommendation. 27 April 2007. http://www.w3.org/TR/soap12

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

[XMLSignature] XML Signature Syntax and Processing. W3C Recommendation. 10 June 2008.
http://www.w3.org/TR/xmldsig-core/

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

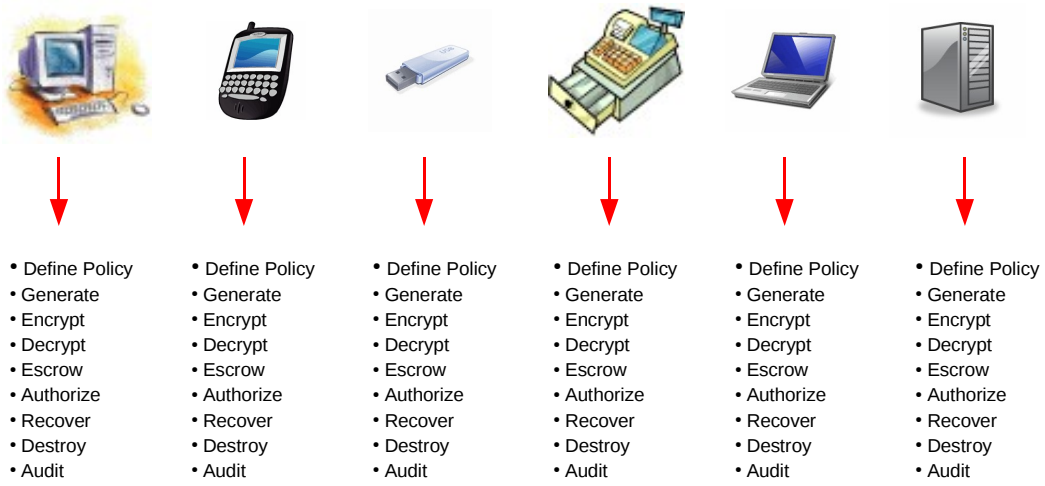
[RFC 2578] K. McCloghrie, et al. *Structure of Management Information Version 2 (SMIv2)*.IETF RFC 2578, April 1999. http://www.rfc-editor.org/rfc/rfc2578.txt

2 Background (non-normative)

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

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

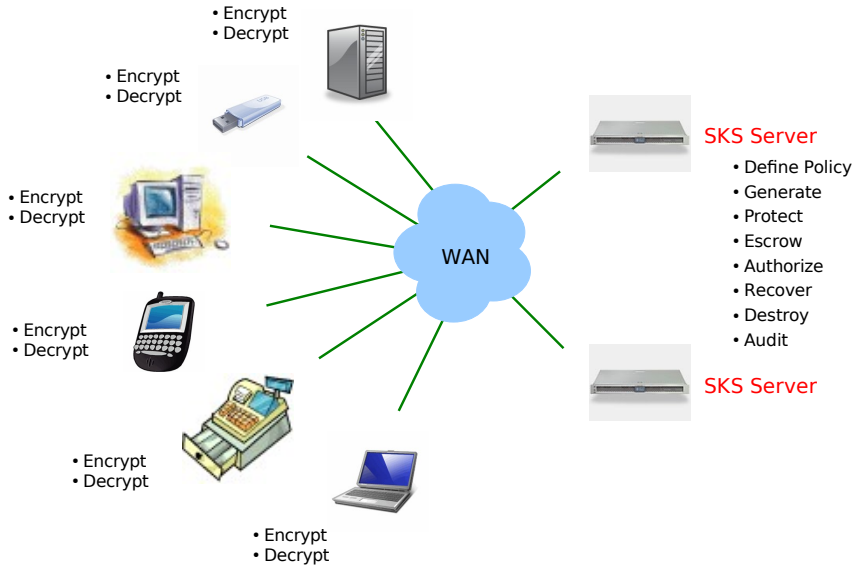
In a rush to provide solutions to the market, vendors have created many device-specific, platform-specific, database-specific and application-specific encryption and key-management tools. While these tools may be capable of performing their stated tasks adequately, a typical enterprise would have to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



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

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

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



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

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

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

2.1 Requirements (non-normative)

The requirements of the SKSML protocol are that:

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

SKSML meets the above requirements in the following manner:

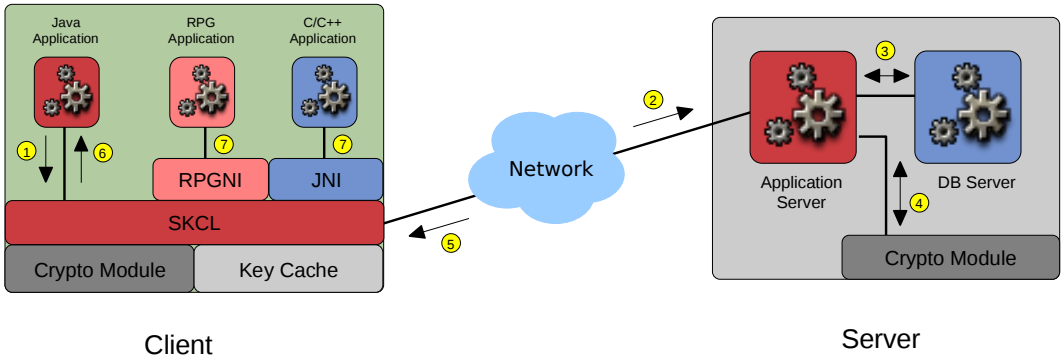
- SKSML uses XML for encapsulating its requests and responses and can thus, be used on any platform that supports XML;

- Using a scheme that concatenates unique Domain identifiers (Private Enterprise Numbers issued by the IANA), unique SKS Server identifiers within a domain and unique Key identifiers within an SKS server, SKSML creates Global Key Identifiers (GKID) that can uniquely identify symmetric keys across the internet;
- SKSML relies on XML Signature and XML Encryption. Relying on RSA cryptographic key-pairs and digital certificates, SKSML uses the digital signatures for authenticity and message-integrity, while using RSA-encryption for confidentiality;
- Using secure key-caching enabled through centrally-defined policies, SKSML supports the request and receipt of ***KeyCachePolicy*** elements by clients for the use of symmetric encryption keys even when the client is disconnected from the network and an SKS server;
- SKSML provides significant flexibility for defining policies on how symmetric encryption keys may be used by client applications. The ***KeyUsePolicy*** element allows Security Officers to define which applications may use a specific key, days and times of use, location of use, purpose of use, key-sizes, encryption algorithms, etc.

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

3 Examples of use of SKSML (non-normative)

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



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

3.1 Request for a new symmetric key

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

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

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

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

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

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

3.2 Response with a new symmetric key

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

```
[b01]  <ekmi:SymkeyResponse
[b02]         xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
[b03]            xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
[b04]      <ekmi:Symkey>
[b05]         <ekmi:SymkeyRequestID>10514-1-7476</ekmi:SymkeyRequestID>
[b06]         <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
[b07]          <ekmi:KeyUsePolicy>
[b08]            <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>
[b09]            <ekmi:PolicyName>DES-EDE KeyUsePolicy</ekmi:PolicyName>
[b10]            <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
[b11]            <ekmi:KeyAlgorithm>
[b12]               http://www.w3.org/2001/04/xmlenc#tripledes-cbc
[b13]            </ekmi:KeyAlgorithm>
[b14]             <ekmi:KeySize>192</ekmi:KeySize>
[b15]             <ekmi:Status>Active</ekmi:Status>
[b16]            <ekmi:Permissions>
[b17]               <ekmi:PermittedApplications ekmi:any=”false”>
[b18]                     <ekmi:PermittedApplication>
[b19]                     <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
[b20]                      <ekmi:ApplicationName>
[b21]                            Payroll Application
[b22]                     </ekmi:ApplicationName>
[b23]                      <ekmi:ApplicationVersion>1.0</ekmi:ApplicationVersion>
[b24]                      <ekmi:ApplicationDigestAlgorithm>
[b25]                         http://www.w3.org/2000/09/xmldsig#sha1
[b26]                      </ekmi:ApplicationDigestAlgorithm>
[b27]                      <ekmi:ApplicationDigestValue>
[b28]                         NIG4bKkt4cziEqFFuOoBTM81efU=
[b29]                      </ekmi:ApplicationDigestValue>
[b30]                  </ekmi:PermittedApplication>
[b31]               </ekmi:PermittedApplications>
[b32]               <ekmi:PermittedDates ekmi:any=”false”>
[b33]                  <ekmi:PermittedDate>
[b34]                     <ekmi:StartDate>2008-01-01</ekmi:StartDate>
[b35]                    <ekmi:EndDate>2008-12-31</ekmi:EndDate>
[b36]                  </ekmi:PermittedDate>
[b37]               </ekmi:PermittedDates>
[b38]               <ekmi:PermittedDays ekmi:any=”true” xsi:nil=”true”/>
[b39]               <ekmi:PermittedDuration ekmi:any=”true” xsi:nil=”true”/>
[b40]               <ekmi:PermittedLevels ekmi:any=”true” xsi:nil=”true”/>
[b41]               <ekmi:PermittedLocations ekmi:any=”true” xsi:nil=”true”/>
[b42]               <ekmi:PermittedNumberOfTransactions ekmi:any=”true”
xsi:nil=”true”/>
[b43]               <ekmi:PermittedTimes ekmi:any=”false”>
[b44]                  <ekmi:PermittedTime>
[b45]                     <ekmi:StartTime>07:00:00</ekmi:StartTime>
[b46]                    <ekmi:EndTime>19:00:00</ekmi:EndTime>
```

```
[b47]                    </ekmi:PermittedTime>
[b48]                 </ekmi:PermittedTimes>
[b49]                 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
[b50]              </ekmi:Permissions>
[b51]           </ekmi:KeyUsePolicy>
[b52]              <ekmi:EncryptionMethod
[b53]                    Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[b54]              <xenc:CipherData>
[b55]                 <xenc:CipherValue>
[b56]                E9zWB/y93hVSzeTLiDcQoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJM
[b57]                hQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/2fw
[b58]                UJow/qimxi8+huUYJMtaGHtXuLlWtx27STRcRpIsY=
[b59]                 </xenc:CipherValue>
[b60]              </xenc:CipherData>
[b61]        </ekmi:Symkey>
[b62]  </ekmi:SymkeyResponse>
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.3 Request for an existing symmetric key

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

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

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

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

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

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

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

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

3.4 Response with an existing symmetric key

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.7 Request for multiple new symmetric keys

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.8 Response with multiple new symmetric keys

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

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

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

```
[h01] <ekmi:SymkeyResponse
[h02]       xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
[h03]        xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
[h04]    <ekmi:Symkey>
[h05]       <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
[h06]       <ekmi:GlobalKeyID>10514-4-3792</ekmi:GlobalKeyID>
[h07]        <ekmi:KeyUsePolicy>
[h08]           <ekmi:KeyUsePolicyID>10514-9</ekmi:KeyUsePolicyID>
[h09]           <ekmi:PolicyName>DES-EDE KeyUsePolicy for EHR-CDC</ekmi:PolicyName>
[h10]           <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
[h11]           <ekmi:KeyAlgorithm>
[h12]              http://www.w3.org/2001/04/xmlenc#tripledes-cbc
[h13]           </ekmi:KeyAlgorithm>
[h14]           <ekmi:KeySize>192</ekmi:KeySize>
[h15]           <ekmi:Status>Active</ekmi:Status>
[h16]           <ekmi:Permissions>
[h17]              <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
[h18]              <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
[h19]              <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
[h20]              <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
[h21]              <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
[h22]              <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
[h23]              <ekmi:PermittedNumberOfTransactions
[h24]                        ekmi:any="true" xsi:nil="true"/>
[h25]              <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
[h26]              <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
[h27]           </ekmi:Permissions>
[h28]       </ekmi:KeyUsePolicy>
[h29]       <ekmi:EncryptionMethod
[h30]              Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[h31]       <xenc:CipherData>
[h32]           <xenc:CipherValue>
[h33]           E9zWB/y93hVSzeTLiDcQoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJMa1w
[h34]           hQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxjfg1pU8tlWtx27STRcR/2fw2ava
[h35]           UlWtx27STRcRJMtaGHtXuLlWtx27STRcRpIsY=
[h36]           </xenc:CipherValue>
[h37]        </xenc:CipherData>
[h38]    </ekmi:Symkey>
[h39]    <ekmi:Symkey>
[h40]       <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
[h41]       <ekmi:GlobalKeyID>10514-4-3793</ekmi:GlobalKeyID>
[h42]        <ekmi:KeyUsePolicy>
[h43]           <ekmi:KeyUsePolicyID>10514-12</ekmi:KeyUsePolicyID>
[h44]           <ekmi:PolicyName>DES-EDE KeyUsePolicy for EHR-CRO</ekmi:PolicyName>
[h45]               <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
[h46]               <ekmi:KeyAlgorithm>
[h47]                  http://www.w3.org/2001/04/xmlenc#tripledes-cbc
[h48]               </ekmi:KeyAlgorithm>
[h49]           <ekmi:KeySize>192</ekmi:KeySize>
[h50]           <ekmi:Status>Active</ekmi:Status>
[h51]           <ekmi:Permissions>
[h52]              <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
[h53]              <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
[h54]                  <ekmi:PermittedDate>
[h55]                      <ekmi:StartDate>2008-01-01</ekmi:StartDate>
[h56]                      <ekmi:EndDate>2009-12-31</ekmi:EndDate>
[h57]                  </ekmi:PermittedDate>
```

```
[h58]                </ekmi:PermittedDates>
[h59]               <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
[h60]               <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
[h61]               <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
[h62]               <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
[h63]               <ekmi:PermittedNumberOfTransactions
[h64]                        ekmi:any="true" xsi:nil="true"/>
[h65]               <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
[h66]               <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
[h67]           </ekmi:Permissions>
[h68]       </ekmi:KeyUsePolicy>
[h69]       <ekmi:EncryptionMethod
[h70]              Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[h71]       <xenc:CipherData>
[h72]           <xenc:CipherValue>
[h73]           qUiQXGOca8EU871zBoXBjDoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJM1
[h74]           hQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/2fw1
[h75]           UJow/qimxi8+ huUYJMtaGHtXuLlWtx27STRcRpIsY=
[h76]           </xenc:CipherValue>
[h77]       </xenc:CipherData>
[h78]     </ekmi:Symkey>
[h79]     <ekmi:Symkey>
[h80]       <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
[h81]       <ekmi:GlobalKeyID>10514-4-3795</ekmi:GlobalKeyID>
               ...
[h82]               <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
               ...
[h83]     </ekmi:Symkey>
[h84]     <ekmi:Symkey>
[h85]       <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
[h86]       <ekmi:GlobalKeyID>10514-4-3797</ekmi:GlobalKeyID>
               ...
[h87]               <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
               ...
[h88]     </ekmi:Symkey>
[h89]     <ekmi:Symkey>
[h90]       <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
[h91]       <ekmi:GlobalKeyID>10514-4-3798</ekmi:GlobalKeyID>
               ...
[h92]               <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
               ...
[h93]     </ekmi:Symkey>
[h94]     <ekmi:Symkey>
[h95]       <ekmi:GlobalKeyID>10514-4-3799</ekmi:GlobalKeyID>
               ...
[h96]               <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
               ...
[h97]     </ekmi:Symkey>
[h98]     <ekmi:Symkey>
[h99]       <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
[h100]      <ekmi:GlobalKeyID>10514-4-3801</ekmi:GlobalKeyID>
               ...
[h101]              <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
               ...
[h102]    </ekmi:Symkey>
[h103]    <ekmi:Symkey>
[h104]      <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
[h105]      <ekmi:GlobalKeyID>10514-4-3803</ekmi:GlobalKeyID>
               ...
[h106]              <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
```

```
                ...
[h107]      </ekmi:Symkey>
[h108]      <ekmi:Symkey>
[h109]        <ekmi:SymkeyRequestID>10514-4-78122</ekmi:SymkeyRequestID>
[h110]        <ekmi:GlobalKeyID>10514-4-3805</ekmi:GlobalKeyID>
              ...
[h111]              <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>
                ...
[h112]      </ekmi:Symkey>
[h113]    </ekmi:SymkeyResponse>
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.9 Request for a symmetric key with an Encryption Certificate

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

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

```
[i01]  <ekmi:SymkeyRequest
[i02]      xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
[i03]      <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
[i04]      <ekmi:X509EncryptionCertificate>
    MIIDfDCCAmSgAwIBAgIIAe/AvliGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdU3Ryb25nab
    S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvciBTdHJvbmdLZXkgREVNTyBVc2Ug1d
    T25seTEXMBUGA1UEChMOU3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTcxMDMwWhcNMDcwNzIa64dd3k
    A1UECxMbRm9yIFN0cm9uZ0tleSBERU1PIFVzZSBPbmx5MRcwFQYDVQQKEw5TdHJvbmdBdXRoIEl2da
    S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvciBTdHJvbmdLZXkgREVNTyBVc2Ugia
    T25seTEXMBUGA1UEChMOU3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTY0NjEwWhcNMDcwNzI1s34wdd
    NjEwWjBpMREwDwYKCZImiZPyLGQBARMBOTEVMBMGA1UEAxMMU0tTIFNlcnZlci0xMSQwIgYDVQsdw2
    ExtGb3IgU3Ryb25nS2V5IERFTU8gVXNlIE9ubHkxFzAVBgNVBAoTDlN0cm9uZ0F1dGggSW5jMIIBd2
```

```
     NBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAztppqRoU5A8plxx1Rz1QEUnlAAM1D5g9+isIr3wxa
    hbwjtFSMYilnY4iV77xU/nsMOnMZ7RxsLYKdCzQ1ODVYqQwqmAvaJ5Z6SVy34gZ51YG+rSWE3NjFsd
    bOXW8RJYA/Tn6Lmht/qngrcaqqmtP0cAAiMRZOWtCTmC2K/LEqDabXSyU6Hh8ySNE3njybvmWpresf
    zsYokTdvnWQqT6tKo1OwJsdJ1+hxM7DrnMLvMNq5reINfsKhDdX17wzhrBUx+hiYA/qo8tMXkL6wsd
     4PN5dYugtzpSzIdUO5tIg58Avhzwo7hy5oofBlKFY22CeljQ36u0bMjuyGj6UYHs3rdfdfsds32rda
    YzCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAyAmxMZhYA8wHJ4UE4b61s51JVWe4Fygj4MCf3a
    hvcNAQELBQADggEBACK05PtvZD4WPglOe=
[i05]     </ekmi:X509EncryptionCertificate>
[i06]  </ekmi:SymkeyRequest>
```

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

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

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

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

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

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

3.10 Response with an SKS error

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

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

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

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

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

[j05] identifies the ***SymkeyRequestID*** assigned by the SKS server for this request. In this example, the concatenated values of the Domain ID, Server ID and Request ID indicate that the key belongs to the organization represented by the PEN, 10514; it was generated on an SKS server with a Server ID of 1 and is the 1,044th unique request received on that SKS server.

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

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

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

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

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

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

3.11 Response with symmetric keys and errors

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

i. Return all symmetric keys as requested;

ii. Return no symmetric keys – i.e. it returns all errors;

iii. Return some symmetric keys and some errors.

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

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

```
[k01]      <ekmi:SymkeyResponse
[k02]           xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
[k03]           xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
[k04]         <ekmi:Symkey>
[k05]              <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
[k06]              <ekmi:GlobalKeyID>10514-4-3792</ekmi:GlobalKeyID>
[k07]             <ekmi:KeyUsePolicy>
[k08]               <ekmi:KeyUsePolicyID>10514-9</ekmi:KeyUsePolicyID>
[k09]                  <ekmi:PolicyName>DES-EDE Policy for EHR-CDC</ekmi:PolicyName>
[k10]                  <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
[k11]                  <ekmi:KeyAlgorithm>
[k12]                      http://www.w3.org/2001/04/xmlenc#tripledes-cbc
[k13]                  </ekmi:KeyAlgorithm>
[k14]               <ekmi:KeySize>192</ekmi:KeySize>
[k15]               <ekmi:Status>Active</ekmi:Status>
[k16]               <ekmi:Permissions>
[k17]                  <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
[k18]                  <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
[k19]                  <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
[k20]                  <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
[k21]                  <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
[k22]                  <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
[k23]                  <ekmi:PermittedNumberOfTransactions
```

```
[k24]                          ekmi:any="true" xsi:nil="true"/>
[k25]                  <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
[k26]                  <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
[k27]               </ekmi:Permissions>
[k28]            </ekmi:KeyUsePolicy>
[k29]            <ekmi:EncryptionMethod
[k30]                  Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[k31]            <xenc:CipherData>
[k32]               <xenc:CipherValue>
[k33]               E9zWB/y93hVSzeTLiDcQoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJ
[k34]               hQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxjfg1pU8tlWtx27STRcR/2fw
[k35]               UlWtx27STRcRJMtaGHtXuLlWtx27STRcRpIsY=
[k36]               </xenc:CipherValue>
[k37]            </xenc:CipherData>
[k38]         </ekmi:Symkey>
[k39]         <ekmi:Symkey>
[k40]               <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
[k41]               <ekmi:GlobalKeyID>10514-4-3793</ekmi:GlobalKeyID>
[k42]            <ekmi:KeyUsePolicy>
[k43]               <ekmi:KeyUsePolicyID>10514-12</ekmi:KeyUsePolicyID>
[k44]                  <ekmi:PolicyName>DES-EDE Policy for EHR-CRO</ekmi:PolicyName>
[k45]                  <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
[k46]                  <ekmi:KeyAlgorithm>
[k47]                     http://www.w3.org/2001/04/xmlenc#tripledes-cbc
[k48]                  </ekmi:KeyAlgorithm>
[k49]               <ekmi:KeySize>192</ekmi:KeySize>
[k50]               <ekmi:Status>Active</ekmi:Status>
[k51]               <ekmi:Permissions>
[k52]                  <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
[k53]                  <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
[k54]                      <ekmi:PermittedDate>
[k55]                          <ekmi:StartDate>2008-01-01</ekmi:StartDate>
[k56]                          <ekmi:EndDate>2009-12-31</ekmi:EndDate>
[k57]                      </ekmi:PermittedDate>
[k58]                   </ekmi:PermittedDates>
[k59]                  <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
[k60]                  <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
[k61]                  <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
[k62]                  <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
[k63]                  <ekmi:PermittedNumberOfTransactions
[k64]                            ekmi:any="true" xsi:nil="true"/>
[k65]                  <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
[k66]                  <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
[k67]               </ekmi:Permissions>
[k68]            </ekmi:KeyUsePolicy>
[k69]            <ekmi:EncryptionMethod
[k70]                  Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[k71]            <xenc:CipherData>
[k72]               <xenc:CipherValue>
[k73]               qUiQXGOca8EU871zBoXBjDoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCF
[k74]               hQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/
[k75]               UJow/qimxi8+ huUYJMtaGHtXuLlWtx27STRcRpIsY=
[k76]               </xenc:CipherValue>
[k77]            </xenc:CipherData>
[k78]         </ekmi:Symkey>
[k79]         <ekmi:Symkey>
[k80]               <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
[k81]            <ekmi:GlobalKeyID>10514-4-3795</ekmi:GlobalKeyID>
                        ...
[k82]                  <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
```

```
                        ...
[k83]           </ekmi:Symkey>
[k84]           <ekmi:Symkey>
[k85]             <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
[k86]             <ekmi:GlobalKeyID>10514-4-3797</ekmi:GlobalKeyID>
                        ...
[k87]                     <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
                        ...
[k88]           </ekmi:Symkey>
[k89]           <ekmi:Symkey>
[k90]             <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
[k91]             <ekmi:GlobalKeyID>10514-4-3798</ekmi:GlobalKeyID>
                        ...
[k92]                     <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
                        ...
[k93]           </ekmi:Symkey>
[k94]           <ekmi:Symkey>
[k95]               <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
[k96]             <ekmi:GlobalKeyID>10514-4-3799</ekmi:GlobalKeyID>
                        ...
[k97]                     <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
                        ...
[k98]           </ekmi:Symkey>
[k99]           <ekmi:Symkey>
[k100]            <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
[k101]            <ekmi:GlobalKeyID>10514-4-3801</ekmi:GlobalKeyID>
                        ...
[k102]                    <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
                        ...
[k103]          </ekmi:Symkey>
[k104]          <ekmi:SymkeyError>
[k105]            <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
[k106]            <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
[k107]            <ekmi:RequestedKeyClass>EHR-PAT</ekmi:RequestedKeyClass>
[k108]            <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
[k109]            <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
[k110]          </ekmi:SymkeyError>
[k111]          <ekmi:SymkeyError>
[k112]            <ekmi:SymkeyRequestID>10514-4-1125927</ekmi:SymkeyRequestID>
[k113]            <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
[k114]            <ekmi:RequestedKeyClass>EHR-PHY</ekmi:RequestedKeyClass>
[k115]            <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
[k116]            <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
[k117]          </ekmi:SymkeyError>
[k118]      </ekmi:SymkeyResponse>
```

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

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

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

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

3.12 Response with a pending Request ID

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

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

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

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

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

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

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

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

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

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

3.13 Request for an update of a pending Request ID

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

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

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

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

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

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

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

3.14 Request for a symmetric key-caching policy

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
[p16]         <ekmi:Status>Active</ekmi:Status>
[p17]         <ekmi:NewKeysCacheDetail>
[p18]            <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
[p19]            <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
[p20]         </ekmi:NewKeysCacheDetail>
[p21]         <ekmi:UsedKeysCacheDetail>
[p22]            <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
[p23]            <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
[p24]         </ekmi:UsedKeysCacheDetail>
[p25]      </ekmi:KeyCachePolicy>
[p26]  </ekmi:KeyCachePolicyResponse>
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
[q01]  <ekmi:KeyCachePolicyResponse
[q02]            xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
[q03]            xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
[q04]     <ekmi:KeyCachePolicy>
[q05]            <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
[q06]            <ekmi:PolicyName>No Caching Policy</ekmi:PolicyName>
[q07]            <ekmi:Description>
[q08]               This policy is for high-risk, always-connected machines on the
[q09]               network, which will never cache symmetric keys locally.  This
[q10]               policy never expires (but checks monthly for any updates).
[q11]            </ekmi:Description>
[q12]            <ekmi:KeyClass>NoCachingClass</ekmi:KeyClass>
[q13]            <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
[q14]            <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
[q15]            <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
[q16]            <ekmi:Status>Active</ekmi:Status>
[q17]     </ekmi:KeyCachePolicy>
[q18]     <ekmi:KeyCachePolicy>
[q19]        <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
[q20]        <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
[q21]        <ekmi:Description>
[q22]           This policy defines how company-issued laptops will manage symmetric
[q23]           keys used for file/disk encryption in their local cache.  This
[q24]           policy must be used by all laptops that use the company EKMI.
[q25]        </ekmi:Description>
[q26]        <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
[q27]        <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
[q28]        <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
[q29]        <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
[q30]        <ekmi:Status>Active</ekmi:Status>
[q31]        <ekmi:NewKeysCacheDetail>
[q32]           <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
[q33]           <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
[q34]        </ekmi:NewKeysCacheDetail>
[q35]        <ekmi:UsedKeysCacheDetail>
```

```
[q36]            <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
[q37]            <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
[q38]         </ekmi:UsedKeysCacheDetail>
[q39]      </ekmi:KeyCachePolicy>
[q40]      <ekmi:KeyCachePolicy>
[q41]         <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
[q42]         <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
[q43]         <ekmi:Description>
[q44]            This policy defines how company-issued laptops will manage
[q45]            symmetric keys used for file/disk encryption in their local
[q46]            cache.  This policy must be used by all laptops.
[q47]         </ekmi:Description>
[q48]         <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
[q49]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
[q50]         <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
[q51]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
[q52]         <ekmi:Status>Active</ekmi:Status>
[q53]         <ekmi:NewKeysCacheDetail>
[q54]            <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
[q55]            <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
[q56]         </ekmi:NewKeysCacheDetail>
[q57]         <ekmi:UsedKeysCacheDetail>
[q58]            <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
[q59]            <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
[q60]         </ekmi:UsedKeysCacheDetail>
[q61]      <ekmi:KeyCachePolicy>
[q62]  </ekmi:KeyCachePolicyResponse>
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1524 **[q60]** is the closing tag of the ***UsedKeysCacheDetail*** element.

1525 **[q61]** is the closing tag of the third and final ***KeyCachePolicy*** element in this response.

1526 **[q62]** is the closing tag of the ***KeyCachePolicyResponse*** element.

4 Specification

4.1 Element <SymkeyRequest>

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

Schema Definition:

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

                    <xsd:element
                       name="X509EncryptionCertificate"
                       type="ekmi:X509CertificateType"
                       minOccurs="0"
                       maxOccurs="1">
                    </xsd:element>
                </xsd:sequence>
                <xsd:sequence>
                   <xsd:element
                    name="SymkeyRequestID"
```

```
                    type="ekmi:SymkeyRequestIDType"

                    minOccurs="1"
                    maxOccurs="unbounded">
                  </xsd:element>
               </xsd:sequence>
            </xsd:choice>
      </xsd:complexType>
</xsd:element>
```

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

1. <GlobalKeyID> [Required]

   The first sequence choice consists of the <GlobalKeyID> element of type ***GlobalKeyIDType***. It identifies the unique global key identifier of the requested symmetric key within the target Symmetric Key Management System (SKMS). If this sequence is chosen, 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 and ***GlobalKeyIDType*** is specified in Section 4.2.

   <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 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.

   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 and their respective types are specified in Section 4.3.

   <X509EncryptionCertificate> [Optional]

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

a) It MUST be a valid X509 v3 digital certificate whose expiry is not earlier than the date and time the symmetric-key request is received by the **SKS** server;

b) It MUST not be revoked by the Certificate Authority (CA) who issued the encryption certificate at the date and time the symmetric-key request is received by the **SKS** server;

c) It MUST have its *keyEncipherment* bit set in the *keyUsage* extension of the certificate.

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

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

2. <SymkeyRequestID> [Required]

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

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

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

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>
```

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

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

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

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

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

```
    <ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
        <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
        <ekmi:X509EncryptionCertificate>
    MIIDfDCCAmSgAwIBAgIIAe/AvliGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdU3Ryb25n
    S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvciBTdHJvbmdLZXkgREVNTyBVc2Ug
    T25seTEXMBUGA1UEChMOU3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTcxMDMwWhcNMDcwNzI1MTcy
    MDMwWjBtMREwDwYKCZImiZPyLGQBARMBMjEZMBcGA1UEAxMQUE9TIFJlZ2lzdGVyIDIyMjEkMCIG
    A1UECxMbRm9yIFN0cm9uZ0tleSBERU1PIFVzZSBPbmx5MRcwFQYDVQQKEw5TdHJvbmdBdXRoIElu
    YzCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAyAmxMZhYA8wHJ4UE4b61s51JVWe4Fygj4MCf
    U7LA3JhpUS4TlX0XFWqrcmltLOiVG7YBFarJFluBFJW2X6q8FuvUprv4V9nJrgiwAPtkiRyIx96n
    qKXIxkUlQ4idlEg1AZI9dEdf4Y5cqBBCygPYnBoTudglM7R47AjR4nr4ks8CAwEAAaOBqTCBpjAO
    BgNVHQ8BAf8EBAMCBLAwHQYDVR0OBBYEFOIOrWrZo0LdBRLVncRAwLBqVZpCMB8GA1UdIwQYMBaA
    FPTYwEHoJG4iFVHRnt2EWxGluAQVMBgGA1UdIAQRMA8wDQYLKwYEAdISg30BBAEwOgYDVR0fBDMw
    MTAvoC2gK4YpaHR0cDovL2RlbW8uc3Ryb25na2V5Lm9yZy9kZW1vLXN1Yi1jYS5jcmwwDQYJKoZI
```

```
    hvcNAQELBQADggEBACK05PtvZD4WPglOe+EHUiApzFyCdRzf0pFZtxRwG9lR1PZUWUjmwTNfGFsL
    S6kyoHgUfVa5fpT1EU1mXUB/Lmo3hFGyprZjfmD7DwuBcYgmZHv7yHrmGOMIOXjFTACvHpM0vOce
    hVx2e4VE0yhBLu/ldH9awGGDp6Bk2XzxqQcs8y6ZzOXZAnPgKQZdjbFKERSsy/d1D8pk5baBk4bd
    Zh568OcaUrbm9ZReRVTVaY5qiQpkOU+tDrBSj/HIL6GAqegYllkz6KYCy6RVOy6iVVSjHocDqdJr
    EVOR+ds6xn8mmojdlERrILmuxiLpibPp609SfnDIxNlzLwe5g7ep3lc=
         </ekmi:X509EncryptionCertificate>
    </ekmi:SymkeyRequest>
```

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

```
    <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:X509EncryptionCertificate>
    MIIDfDCCAmSgAwIBAgIIAe/AvliGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdU3Ryb25n
    S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvciBTdHJvbmdLZXkgREVNTyBVc2Ug
    T25seTEXMBUGA1UEChMOU3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTcxMDMwWhcNMDcwNzI1MTcy
    MDMwWjBtMREwDwYKCZImiZPyLGQBARMBMjEZMBcGA1UEAxMQUE9TIFJlZ2lzdGVyIDIyMjEkMCIG
    A1UECxMbRm9yIFN0cm9uZ0tleSBERU1PIFVzZSBPbmx5MRcwFQYDVQQKEw5TdHJvbmdBdXRoIElu
    YzCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAyAmxMZhYA8wHJ4UE4b61s51JVWe4Fygj4MCf
    U7LA3JhpUS4TlX0XFWqrcmltLOiVG7YBFarJFluBFJW2X6q8FuvUprv4V9nJrgiwAPtkiRyIx96n
    qKXIxkUlQ4idlEg1AZI9dEdf4Y5cqBBCygPYnBoTudglM7R47AjR4nr4ks8CAwEAAaOBqTCBpjAO
    BgNVHQ8BAf8EBAMCBLAwHQYDVR0OBBYEFOIOrWrZo0LdBRLVncRAwLBqVZpCMB8GA1UdIwQYMBaA
    FPTYwEHoJG4iFVHRnt2EWxGluAQVMBgGA1UdIAQRMA8wDQYLKwYEAdISg30BBAEwOgYDVR0fBDMw
    MTAvoC2gK4YpaHR0cDovL2RlbW8uc3Ryb25na2V5Lm9yZy9kZW1vLXN1Yi1jYS5jcmwwDQYJKoZI
    hvcNAQELBQADggEBACK05PtvZD4WPglOe+EHUiApzFyCdRzf0pFZtxRwG9lR1PZUWUjmwTNfGFsL
    S6kyoHgUfVa5fpT1EU1mXUB/Lmo3hFGyprZjfmD7DwuBcYgmZHv7yHrmGOMIOXjFTACvHpM0vOce
    hVx2e4VE0yhBLu/ldH9awGGDp6Bk2XzxqQcs8y6ZzOXZAnPgKQZdjbFKERSsy/d1D8pk5baBk4bd
    Zh568OcaUrbm9ZReRVTVaY5qiQpkOU+tDrBSj/HIL6GAqegYllkz6KYCy6RVOy6iVVSjHocDqdJr
    EVOR+ds6xn8mmojdlERrILmuxiLpibPp609SfnDIxNlzLwe5g7ep3lc=
         </ekmi:X509EncryptionCertificate>
    </ekmi:SymkeyRequest>
```

4.2 Element <GlobalKeyID>

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

Schema Definition:

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

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

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

2. A literal hyphen ("-") without surrounding spaces;
3. A positive integer identifying the Server ID of the server;
4. Another literal hyphen ("-") without surrounding spaces;
5. A positive integer identifying the unique Key ID;

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

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

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

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

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

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

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

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

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

```
<ekmi:GlobalKeyID>

18446744073709551615-18446744073709551615-18446744073709551615

</ekmi:GlobalKeyID>
```

4.3 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:

```
<xsd:complexType name="KeyClassesType">
   <xsd:sequence>
      <xsd:element
         name="KeyClass"
         type="ekmi:KeyClassType"
         minOccurs="1"
         maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="KeyClassType">
   <xsd:restriction base="xsd:string">
     <xsd:maxLength value="255"/>
   </xsd:restriction>
    </xsd:simpleType>
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.4 Element <X509EncryptionCertificate>

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

Schema Definition:

```
<xsd:simpleType name="X509EncryptionCertificateType">
    <xsd:restriction base="xsd:base64Binary"/>
</xsd:simpleType>
```

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

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

```
<ekmi:SymkeyRequest xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
     <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
     <ekmi:X509EncryptionCertificate>
MIIDfDCCAmSgAwIBAgIIAe/AvliGc3AwDQYJKoZIhvcNAQELBQAwZzEmMCQGA1UEAxMdU3Ryb25n
S2V5IERFTU8gU3Vib3JkaW5hdGUgQ0ExJDAiBgNVBAsTG0ZvciBTdHJvbmdLZXkgREVNTyBVc2Ug
T25seTEXMBUGA1UEChMOU3Ryb25nQXV0aCBJbmMwHhcNMDYwNzI1MTcxMDMwWhcNMDcwNzI1MTcy
MDMwWjBtMREwDwYKCZImiZPyLGQBARMBMjEZMBcGA1UEAxMQUE9TIFJlZ2lzdGVyIDIyMjEkMCIG
A1UECxMbRm9yIFN0cm9uZ0tleSBERU1PIFVzZSBPbmx5MRcwFQYDVQQKEw5TdHJvbmdBdXRoIElu
YzCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAyAmxMZhYA8wHJ4UE4b61s51JVWe4Fygj4MCf
U7LA3JhpUS4TlX0XFWqrcmltLOiVG7YBFarJFluBFJW2X6q8FuvUprv4V9nJrgiwAPtkiRyIx96n
qKXIxkUlQ4idlEg1AZI9dEdf4Y5cqBBCygPYnBoTudglM7R47AjR4nr4ks8CAwEAAaOBqTCBpjAO
BgNVHQ8BAf8EBAMCBLAwHQYDVR0OBBYEFOIOrWrZo0LdBRLVncRAwLBqVZpCMB8GA1UdIwQYMBaA
FPTYwEHoJG4iFVHRnt2EWxGluAQVMBgGA1UdIAQRMA8wDQYLKwYEAdISg30BBAEwOgYDVR0fBDMw
MTAvoC2gK4YpaHR0cDovL2RlbW8uc3Ryb25na2V5Lm9yZy9kZW1vLXN1Yi1jYS5jcmwwDQYJKoZI
hvcNAQELBQADggEBACK05PtvZD4WPglOe+EHUiApzFyCdRzf0pFZtxRwG9lR1PZUWUjmwTNfGFsL
S6kyoHgUfVa5fpT1EU1mXUB/Lmo3hFGyprZjfmD7DwuBcYgmZHv7yHrmGOMIOXjFTACvHpM0vOce
hVx2e4VE0yhBLu/ldH9awGGDp6Bk2XzxqQcs8y6ZzOXZAnPgKQZdjbFKERSsy/d1D8pk5baBk4bd
Zh568OcaUrbm9ZReRVTVaY5qiQpkOU+tDrBSj/HIL6GAqegYllkz6KYCy6RVOy6iVVSjHocDqdJr
EVOR+ds6xn8mmojdlERrILmuxiLpibPp609SfnDIxNlzLwe5g7ep3lc=
     </ekmi:X509EncryptionCertificate>
</ekmi:SymkeyRequest>
```

4.5 Element <SymkeyRequestID>

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

Schema Definition:

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

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

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

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

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

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

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

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

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

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

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

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

4.6 Element <SymkeyResponse>

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

Schema Definition:

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

```
            </xsd:sequence>

            <xsd:sequence>
               <xsd:element
                  name="SymkeyWorkInProgress"
                  type="ekmi:SymkeyWorkInProgressType"
                  minOccurs="1"
                  maxOccurs="unbounded"/>

               <xsd:element
                  name="SymkeyError"
                  type="ekmi:SymkeyErrorType"
                  minOccurs="0"
                  maxOccurs="unbounded"/>
            </xsd:sequence>

            <xsd:sequence>
               <xsd:element
                  name="SymkeyError"
                  type="ekmi:SymkeyErrorType"
                  minOccurs="1"
                  maxOccurs="unbounded"/>
            </xsd:sequence>
         </xsd:choice>
      </xsd:complexType>

          </xsd:element>
```

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

- A sequence of a required <Symkey> element followed with an optional <SymkeyWorkInProgress> and/or an optional <SymkeyError> element;
- A sequence of a required <SymkeyWorkInProgress> element followed by an optional <SymkeyError> element; or
- A required <SymkeyError> element .

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

1. <Symkey> [Required]

   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 4.7.

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

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

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

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

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

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

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

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

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

Example 1 – A response with a single symmetric key:

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

Example 2 – A response with three symmetric keys:

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

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

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

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

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

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

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

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

Example 6 – A response with an error:

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

Example 7 – A response with multiple errors:

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

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

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

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

```
<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>
```

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

```
<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:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
     <ekmi:SymkeyWorkInProgress>.....</ekmi:SymkeyWorkInProgress>
     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
     <ekmi:SymkeyError>.....</ekmi:SymkeyError>
</ekmi:SymkeyResponse>
```

4.7 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:

```
<xsd:complexType name="SymkeyType">
    <xsd:sequence>
        <xsd:element name="SymkeyRequestID" type="ekmi:SymkeyRequestID"/>
        <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 an **SKS** server receives the request asynchronously, the response SHALL contain a <SymkeyWorkInProgress> element implying that the request is being processed and is pending completion. The <SymkeyError> element is specified in Section 4.7.

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 4.9.

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

1. <SymkeyRequestID> [Required]

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

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

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

3. <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 and ***KeyUsePolicyType*** are specified in Section 4.10.

4. <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

5. <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:

```
<ekmi:SymkeyResponse
     xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
     <ekmi:Symkey>
        <ekmi:SymkeyRequestID>10514-1-7455</ekmi:SymkeyRequestID>
        <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
        <ekmi:KeyUsePolicy>.....</ekmi:KeyUsePolicy>
        <ekmi:EncryptionMethod
             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        <xenc:CipherData>
             <xenc:CipherValue>
                E9zWB/y93hVSzeTLiDcQoDxmlNxTux+SffMNwCJmt1dIqzQHBnpdQ8
```

```
                    1g6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxj
                    fg1pU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
                </xenc:CipherValue>
            </xenc:CipherData>
        </ekmi:Symkey>
    </ekmi:SymkeyResponse>
```

Example 2 – A response with multiple symmetric keys:

```
    <ekmi:SymkeyResponse
         xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
         <ekmi:Symkey>
            <ekmi:SymkeyRequestID>10514-1-12455</ekmi:SymkeyRequestID>
            <ekmi:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
            <ekmi:KeyUsePolicy>.....</ekmi:KeyUsePolicy>
            <ekmi:EncryptionMethod
                 Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
            <xenc:CipherData>
                 <xenc:CipherValue>
                    E9zWB/y93hVSzeTLiDcQoDxmlNxTux+SffMNwCJmt1dIqzQHBnpdQ8
                    1g6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxj
                    fg1pU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
                </xenc:CipherValue>
            </xenc:CipherData>
        </ekmi:Symkey>
        <ekmi:Symkey>
            <ekmi:SymkeyRequestID>10514-1-12467</ekmi:SymkeyRequestID>
            <ekmi:GlobalKeyID>10514-1-236</ekmi:GlobalKeyID>
            <ekmi:KeyUsePolicy>.....</ekmi:KeyUsePolicy>
            <ekmi:EncryptionMethod
                 Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
            <xenc:CipherData>
                 <xenc:CipherValue>
                    Qbg65cy93hVSzeTLiDcQoDxmlNxTux+SffMNwCJmt1dIqzQHBnpdQ8
                    7k6DKdkCFjJMhQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxj
                    uyecU8tGFbpWZcd/ATpJD/UJow/qimxi8+huUYJMtaGH=
                 </xenc:CipherValue>
            </xenc:CipherData>
        </ekmi:Symkey>
    </ekmi:SymkeyResponse>
```

4.8 Element <SymkeyWorkInProgress>

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

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

Schema Definition:

```
    <xsd:complexType name="SymkeyWorkInProgressType">
        <xsd:sequence>

            <xsd:element

                name="RequestedGlobalKeyID"

                type="ekmi:GlobalKeyIDType"
```

```
                minOccurs=”1”
                maxOccurs=”1”>
            </xsd:element>
            <xsd:element
                name="RequestedKeyClass"
                type="ekmi:KeyClassType"
                minOccurs="0"
                maxOccurs=”1”>
            </xsd:element>
            <xsd:element
                name="SymkeyRequestID"
                type="ekmi:SymkeyRequestIDType"
                minOccurs=”1”

                maxOccurs=”1”>
            </xsd:element>
            <xsd:element
                name="RequestCheckInterval"
                type="ekmi:RequestCheckIntervalType"
                minOccurs="1"
                maxOccurs="1">
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
```

Schema Definition:

```
    <xsd:simpleType name="RequestCheckIntervalType">
        <xsd:restriction base="xsd:positiveInteger">
            <xsd:minInclusive value="60"/>
            <xsd:maxInclusive value="86400"/>
        </xsd:restriction>
    </xsd:simpleType>
```

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

1. <RequestedGlobalKeyID> [Required]

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

<SymkeyWorkInProgress> element.

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

2. <RequestedKeyClass> [Optional]

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

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

3. <SymkeyRequestID> [Required]

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

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

4. <RequestCheckInterval> [Required]

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

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

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

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

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

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

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

4.9 Element <SymkeyError>

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

Schema Definition:

```
   <xsd:complexType name="SymkeyErrorType">
      <xsd:sequence>
         <xsd:element
              name="SymkeyRequestID" type="ekmi:SymkeyRequestIDType">
         <xsd:element
             name="RequestedGlobalKeyID" type="ekmi:GlobalKeyIDType"/>
         <xsd:element
                name="RequestedKeyClass"

                type="ekmi:KeyClassType"
                minOccurs="0"/>
         <xsd:element name="ErrorCode">
            <xsd:simpleType>
               <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="255"/>
               </xsd:restriction>
             </xsd:simpleType>
         </xsd:element>

         <xsd:element name="ErrorMessage">
            <xsd:simpleType>
               <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="1024"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
      </xsd:sequence>
   </xsd:complexType>
```

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

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

1. <SymkeyRequestID> [Required]

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

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

2. <RequestedGlobalKeyID> [Required]

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

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

3. <RequestedKeyClass> [Optional]

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

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

4. <ErrorCode> [Required]

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

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

5. <ErrorMessage> [Required]

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

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

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

Example 1 – An error within a response:

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

Example 2 – Multiple errors within a response:

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

4.10 Element <KeyUsePolicy>

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

Schema Definition:

```
<xsd:complexType name="KeyUsePolicyType" mixed="true">
     <xsd:sequence>
         <xsd:element name="KeyUsePolicyID" type="ekmi: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="ekmi:KeyClassType"/>
         <xsd:element name="KeyAlgorithm" type="ekmi:EncryptionAlgorithmType"/>
         <xsd:element name="KeySize" type="ekmi:KeySizeType"/>
         <xsd:element name="Status" type="ekmi:StatusType"/>
         <xsd:element name="Permissions" type="ekmi: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 4.11.

2. <PolicyName> [Required]

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

3. <KeyClass> [Required]

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

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

4. <KeyAlgorithm> [Required]

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

   The <KeyAlgorithm> element is specified in Section 4.12.

5. <KeySize> [Required]

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

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

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

6. <Status> [Required]

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

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

7. <Permissions> [Required]

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

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

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

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

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

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

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

        <ekmi:PermittedDate>
                <ekmi:StartDate>2008-01-01</ekmi:StartDate>
                <ekmi:EndDate>2008-12-31</ekmi:EndDate>
            </ekmi:PermittedDate>
        </ekmi:PermittedDates>
        <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
        <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
        <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
        <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
        <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
        <ekmi:PermittedTimes ekmi:any="false">
            <ekmi:PermittedTime>
                <ekmi:StartTime>07:00:00</ekmi:StartTime>
                <ekmi:EndTime>19:00:00</ekmi:EndTime>
            </ekmi:PermittedTime>
        </ekmi:PermittedTimes>
        <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
    </ekmi:Permissions>
</ekmi:KeyUsePolicy>
```

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="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="true" xsi:nil="true"/>
      <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
      <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
   </ekmi:Permissions>
</ekmi:KeyUsePolicy>
```

4.11 Type *TwoPartIDType*

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

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

Schema Definition:

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

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

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

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

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

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

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

Example 3 – A minimum-length *TwoPartIDType* :

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

Example 4 – A maximum-length *TwoPartIDType* :

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

4.12 Element <KeyAlgorithm>

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

Schema Definition:

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

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

1. Triple Data Encryption Standard (3DES)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.13 Element <KeySize>

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

Schema Definition:

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

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

Currently, the following three key-sizes are supported:

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

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

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

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

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

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

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

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

4.14 Element <Status>

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

Schema Definition:

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

Where it 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:

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

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

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

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

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

4.15 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:

```
          <xsd:complexType name="PermissionsType">
          <xsd:sequence>
             <xsd:element
                      name="PermittedApplications"
                      type="ekmi:PermittedApplicationsType"
                      minOccurs="1"
                      nillable=”true”/>
     <xsd:element
                      name="PermittedDates"
                      type="ekmi:PermittedDatesType"
                      minOccurs="1"
                      nillable=”true”/>
     <xsd:element
                      name="PermittedDays"
                      type="ekmi:PermittedDaysType"
                      minOccurs="1"
                      nillable=”true”/>
     <xsd:element name="PermittedDuration"
                      type="ekmi:PermittedDurationType"
                      minOccurs="1"
                      nillable=”true”/>
     <xsd:element
                      name="PermittedLevels"
                      type="ekmi:PermittedLevelsType"
```

```
                        minOccurs="1"
                        nillable=”true”/>
                <xsd:element
                        name="PermittedLocations"
                        type="ekmi:PermittedLocationsType"
                        minOccurs="1"
                        nillable=”true”/>
                <xsd:element
                        name="PermittedNumberOfTransactions"
                        type="ekmi:PermittedNumberOfTransactionsType"
                        minOccurs="1"
                        nillable=”true”/>
                <xsd:element
                        name="PermittedTimes"
                        type="ekmi:PermittedTimesType"
                        minOccurs="1"
                        nillable=”true”/>
                <xsd:element
                        name="PermittedUses"
                        type="ekmi:PermittedUsesType"
                        minOccurs="1"
                        nillable=”true”/>
                <xsd:element
                        name="Other"
                        type="xsd:anyType"
                        minOccurs="0"/>
            </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 4.16.

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

   The <PermittedDates> element is specified in Section 4.17.

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

   The <PermittedDays> element is specified in Section 4.18.

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

a non-zero, positive duration value restricts the use of the key to only the period after the start of the use of the key.

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

The <PermittedDuration> element is specified in Section 4.19.

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

   The <PermittedLevels> element is specified in Section 4.20.

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

   The <PermittedLocations> element is specified in Section 4.21.

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

   The <PermittedNumberOfTransactions> element is specified in Section 4.22.

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

   The <PermittedTimes> element is specified in Section 4.23.

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

   The <PermittedUses> element is specified in Section 4.24.

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

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

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

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

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

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

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

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

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

```
<ekmi:Permissions>
   <ekmi:PermittedApplications ekmi:any="false">
      <ekmi:PermittedApplication>
         <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
         <ekmi:ApplicationName>Payroll Application</ekmi:ApplicationName>
         <ekmi:Version>1.0</ekmi:Version>
         <ekmi:DigestAlgorithm>
            http://www.w3.org/2000/09/xmldsig#sha1
         </ekmi:DigestAlgorithm>
         <ekmi:DigestValue>
            229ea73a5e76eabd183663d332b283948a9202a1
```

```
            </ekmi:DigestValue>
         </ekmi:PermittedApplication>
      </ekmi:PermittedApplications>
      <ekmi:PermittedDates ekmi:any="false">
         <ekmi:PermittedDate>
            <ekmi:StartDate>2008-01-01</ekmi:StartDate>
            <ekmi:EndDate>2008-12-31</ekmi:EndDate>
         </ekmi:PermittedDate>
      </ekmi:PermittedDates>
      <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
      <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
      <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
      <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
      <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
      <ekmi:PermittedTimes ekmi:any="false">
         <ekmi:PermittedTime>
            <ekmi:StartTime>07:00:00</ekmi:StartTime>
            <ekmi:EndTime>19:00:00</ekmi:EndTime>
         </ekmi:PermittedTime>
      </ekmi:PermittedTimes>
      <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
   </ekmi:Permissions>
```

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>2009-12-31</ekmi:EndDate>
         </ekmi:PermittedDate>
      </ekmi:PermittedDates>
```

```
      <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
      <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
      <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
      <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
      <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
      <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
      <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
   </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”>
         <ekmi:PermittedLocation>
            <ekmi:LocationName>Facility A51</ekmi:LocationName>
```

```
            <ekmi:Latitude>37.385562</ekmi:Latitude>
            <ekmi:Longitude>-121.993387</ekmi:Latitude>
         </ekmi:PermittedLocation>
         <ekmi:PermittedLocation>
            <ekmi:LocationName>Facility DC-VA01</ekmi:LocationName>
            <ekmi:Latitude>88.485362</ekmi:Latitude>
            <ekmi:Longitude>-21.453648</ekmi:Latitude>
         </ekmi:PermittedLocation>
      </ekmi:PermittedLocations>
      <ekmi:PermittedNumberOfTransactions ekmi:any="true" xsi:nil="true"/>
      <ekmi:PermittedTimes ekmi:any="false”>
         <ekmi:PermittedTime>
            <ekmi:StartTime>08:00:00</ekmi:StartTime>
            <ekmi:EndTime>17:00:00</ekmi:EndTime>
         </ekmi:PermittedTime>
      </ekmi:PermittedTimes>
      <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
   </ekmi:Permissions>
```

4.16 Element <PermittedApplications> and <PermittedApplication>

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

Schema Definition:

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

Schema Definition:

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

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

Schema Definition:

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

Schema Definition:

```
    <xsd:attribute name="any">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:enumeration value="false"/>
                <xsd:enumeration value="true"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>
```

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 4.10. There SHALL be only one <ApplicationID> element within a <PermittedApplication> element.

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

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

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

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

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

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

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

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

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

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

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

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

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

```
<ekmi:PermittedApplications ekmi:any="false">
   <ekmi:PermittedApplication>
            <ekmi:ApplicationID>10514-24</ekmi:ApplicationID>
         <ekmi:ApplicationName>Employee Tax
Reporting</ekmi:ApplicationName>
      <ekmi:Version>3.3</ekmi:Version>
         <ekmi:DigestAlgorithm>
            http://www.w3.org/2000/09/xmldsig#sha1
      </ekmi:DigestAlgorithm>
       <ekmi:DigestValue>G4bsdfKkt4cziEqFFuOoBTM81efU=</ekmi:DigestValue>
   </ekmi:PermittedApplication>
```

```
      <ekmi:PermittedApplication>
               <ekmi:ApplicationID>10514-25</ekmi:ApplicationID>
          <ekmi:ApplicationName>IRS Tax Reporting
   Application</ekmi:ApplicationName>
          <ekmi:Version>2.1</ekmi:Version>
          <ekmi:DigestAlgorithm>
               http://www.w3.org/2001/04/xmlenc#sha256
         </ekmi:DigestAlgorithm>
          <ekmi:DigestValue>

      ab7b85c9410d48c54fc7939c391be4028e7305085191c56e7b3740f2cbdbbc79
         </ekmi:DigestValue>
      </ekmi:PermittedApplication>
   </ekmi:PermittedApplications>
```

Example 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:

```
   <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:

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

4.17 Element <PermittedDates> and <PermittedDate>

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

Schema Definition:

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

```
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
    <xsd:attribute ref="ekmi:any" use="required"/>
 </xsd:complexType>
```

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

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

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

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

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

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>2009-07-12</ekmi:EndDate>
   </ekmi:PermittedDate>
</ekmi:PermittedDates>
```

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

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

4.18 Element <PermittedDays> and <PermittedDay>

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

Schema Definition:

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

Schema Definition:

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

```
            <xsd:enumeration value="Friday"/>
            <xsd:enumeration value="Saturday"/>
            <xsd:enumeration value="Weekday"/>
            <xsd:enumeration value="Weekend"/>
            </xsd:restriction>
   </xsd:simpleType>
```

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

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

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

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

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

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

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

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

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

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

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

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

4.19 Element <PermittedDuration>

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

Schema Definition:

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

Schema Definition:

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

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

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

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

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 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”/>
```

4.20 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:

```
<xsd:complexType name="PermittedLevelsType">
   <xsd:sequence>
      <xsd:element
                name="PermittedLevel"
                type="ekmi:LevelClassificationType"
                minOccurs="0"
                maxOccurs="unbounded">
      </xsd:element>
      <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute ref="ekmi:any" use="required"/>
</xsd:complexType>
```

Schema Definition:

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

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:

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

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

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

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

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

4.21 Element <PermittedLocations> and <PermittedLocation>

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

Schema Definition:

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

```
                        maxOccurs="unbounded"/>
                    <xsd:element name="Other" type="xsd:anyType" minOccurs="0"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
  <xsd:attribute ref="ekmi: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:fractionDigits value="7"/>
                  </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:

```
<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:

```
<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:

```
<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:

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

4.22 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:

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

Schema Definition:

```
   <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”/>
```

4.23 Element <PermittedTimes> and <PermittedTime>

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

Schema Definition:

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

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

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

- 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:

```
<ekmi:PermittedTimes ekmi:any=”false”>
   <ekmi:PermittedTime>
            <ekmi:StartTime>09:00:00</ekmi:StartTime>
        <ekmi:EndTime>17:00:00</ekmi:EndTime>
   </ekmi:PermittedTime>
</ekmi:PermittedTimes>
```

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

```
<ekmi:PermittedTimes ekmi: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:

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

4.24 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:

```
    <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="ekmi: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:

```
<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):

```
<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:

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

4.25 Element <KeyCachePolicyRequest>

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

Schema Definition:

```
<xsd:element name="KeyCachePolicyRequest">
    <xsd:complexType>
     <xsd:element ref="ds:Signature" maxOccurs="1"/>
    </xsd:complexType>
</xsd:element>
```

The <KeyCachePolicyRequest> has one child element. The child element has the XML Digital signature that can help the server with 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>:

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

4.26 Element <KeyCachePolicyResponse>

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

Schema Definition:

```
<xsd:element name="KeyCachePolicyResponse">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element
                name="KeyCachePolicy"
```

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

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

4.27 Element <KeyCachePolicy>

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

Schema Definition:

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

   <xsd:complexType name="KeyCachePolicyType" mixed="true">
        <xsd:sequence>
            <xsd:element name="KeyCachePolicyID" type="ekmi:TwoPartIDType"/>
            <xsd:element name="PolicyName">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="255"/>
                        <xsd:whiteSpace value="preserve"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Description" nillable="true">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="2048"/>
                        <xsd:whiteSpace value="preserve"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="KeyClass" type="ekmi:KeyClassType"/>
            <xsd:element name="StartDate" type="xsd:dateTime"/>
            <xsd:element name="EndDate" type="xsd:dateTime" nillable="true"/>
            <xsd:element name="PolicyCheckInterval">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:nonNegativeInteger">
                        <xsd:minInclusive value="0"/>
                        <xsd:maxInclusive value="2592000"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Status" type="ekmi:StatusType"/>
            <xsd:element
                 name="NewKeysCacheDetail"
```

```
                type="ekmi:KeyCacheDetailType"
                minOccurs="0"/>
            <xsd:element
                name="UsedKeysCacheDetail"
                type="ekmi:KeyCacheDetailType"
                minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
```

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 4.11.

2. <PolicyName> [Required]

   The <PolicyName> element ,of type XSD ***String****,* with a maximum length of 255 characters, identifies a unique name given to this <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 4.14.

9. <NewKeysCacheDetail> [Required]

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

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

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

   The ***KeyCacheDetailType*** is specified in Section 4.28.

10. <UsedKeysCacheDetail> [Required]

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

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

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

    The ***KeyCacheDetailType*** is specified in Section 4.28.

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

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

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

```
        <ekmi:PolicyCheckInterval>86400</ekmi:PolicyCheckInterval>
        <ekmi:Status>Active</ekmi:Status>
        <ekmi:NewKeysCacheDetail>
            <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
            <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
        </ekmi:NewKeysCacheDetail>
        <ekmi:UsedKeysCacheDetail>
            <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
            <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
        </ekmi:UsedKeysCacheDetail>
    </ekmi:KeyCachePolicy>
```

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

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

4.28 Type *KeyCacheDetailType*

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

Schema Definition:

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

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

1. <MaximumKeys> [Required]

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

2. <MaximumDuration> [Required]

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

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

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

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

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

```
<ekmi:KeyCachePolicy>
    <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
    <ekmi:PolicyName>
        Corporate Laptop Symmetric Key Caching Policy
    </ekmi:PolicyName>
    <ekmi:Description>
        This policy defines how company-issued laptops will manage
        symmetric keys used for file/disk encryption in 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>
```

4.29 Use of Web Services Security (WSS)

While it has been mentioned earlier in this specification, it is explicitly noted here that conforming implementations of the SKSML protocol MAY enclose all SKSML messages between participants in an SKMS in the ***SOAP Body*** of a ***SOAP Envelope***. Additionally, the contents of the ***SOAP Body*** MUST be secured using digital signatures conforming to [**XMLSignature**] in the ***SOAP Header.*** Specifically, 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.

While the payload (the symmetric-key in the <Symkey> element) is already encrypted by the SKS server using the Public Key of the requesting client, the contents of the ***SOAP Body*** MAY, additionally, be secured with encryption using [**XMLEncryption**], if desired. The choice of using the requesting client's Public Key or a SOAP-layer Public Key to encrypt the ***SOAP Body*** is left to the implementers of the SKMS.

The SOAP security objects must be layered on the SKSML messages **after** the SKSML message has been fully constructed (and secured in the case of the symmetric-key payload) by the SKS server.

*Note: In the event that SKMS sites choose to encrypt the SOAP Body's contents at the SOAP layer (in addition to the symmetric key encryption performed at the SKSML layer), the **Security** element in some implementations of [WSS] will have two <BinarySecurityToken> elements with digital certificates in them: one for the sender's signing certificate, and the other for the recipient's encryption certificate. While SKMS sites have the choice of encrypting the contents of the SOAP Body, they do NOT have a choice about signing the contents of the SOAP Body -* all *SKSML messages enclosed in the SOAP Body MUST be signed by clients and servers within an SKMS.*

4.30 Use of SKMS Error Codes & Messages

SKSML defines many messages between the Symmetric Key Client Library (SKCL) implementations and Symmetric Key Services (SKS) servers. Given the complexity of layered-technology, there are ample opportunities for components to fail during processing.

The EKMI TC believes that adoption of this technology can be encouraged by standardizing codes for errors and informational messages used by SKMS clients and servers. At the same time, the TC recognizes the value of providing flexibility for vendor implementations to innovate beyond what the TC has envisioned.

To ensure a baseline level of conformance, the TC has chosen to standardize some common error-codes and messages, while creating a process for vendors to request and receive a block of codes reserved for

the vendor's use within their implementation of SKSML. These codes are defined in **Appendix C** and the process is described in **Appendix D** of this specification.

Conforming implementations of SKSML MUST implement the Standard Error Codes and Message within their implementations. It is left up to the vendor to determine if they wish to apply for, and receive, a reserved block of codes for their own implementation's unique codes and messages.

5 Bindings

An SKSML implementation can provide symmetric key services over a wide variety of transport mechanisms. This is referred to as “Bindings” of this specification. To maintain compliance with the specification, implementations must support the bindings that are REQUIRED.

5.1 W3C Security Binding

An SKSML implementation is REQUIRED to provide a binding that transports SKSML messages with support for XML Signatures and XML Encryption.

5.2 Mutually Authenticated TLS Binding

An SKSML implementation is REQUIRED to provide a binding that transports SKSML messages over a TLS connection that is mutually authenticated.

5.3 SOAP-WSS Binding

An SKSML implementation MAY provide SOAP and Web Services Security [WSS] support for transporting SKSML messages.

6 Conformance

An implementation conforms to this specification if it satisfies all of the MUST or REQUIRED level requirements defined within this specification. An SKSML Node MUST NOT use the XML namespace identifier for this specification (listed in the Title section under Declared Namespace(s)) within SOAP Envelopes unless it is compliant with this specification.

This specification references a number of other specifications (see the table above). In order to comply with this specification, an implementation MUST implement the portions of referenced specifications necessary to comply with the required provisions of this specification. Additionally, the implementation of the portions of the referenced specifications that are specifically cited in this specification MUST comply with the rules for those portions as established in the referenced specification.

Additionally normative text within this specification takes precedence over normative outlines, which in turn take precedence over the XML Schema [XML Schema Part 1, Part 2] descriptions. That is, the normative text in this specification further constrains the schema part of this specification; and this specification contains further constraints on the elements defined in referenced schemas.

If an OPTIONAL message is not supported, then the implementation SHOULD Fault just as it would for any other unrecognized/unsupported message. If an OPTIONAL message is supported, then the implementation MUST satisfy all of the MUST and REQUIRED sections of the message.

Appendix A. Acknowledgments

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

Participants:

Ezer Farhi, Associate
Benjamin Tomhave, Btplc
Tim Bruce, CA
June Leung, Associate
Shaheen N Abdul Jabbar, Individual
Ken Adler, Individual
Stefan Drees. Individual
Marc Massar, Individual
Michael Nelson, Individual
Davi Ottenheimer, Individual
Allen Schaaf, Individual
Harry Haury, NuParadigm Government Systems, Inc.
Tomas Gustavsson, PrimeKey Solutions AB
Anil Saldhana, Red Hat
Arshad Noor, Associate
Sandi Roddy, US Department of Defense (DoD)
Thomas Hardjono, Associate
Upendra Mardikar, Associate
Eric Lengvenis, Wells Fargo

Appendix B. Revision History

Version	Date	Author	Notes
DRAFT 4	June 08, 2008	Arshad Noor	Initial version
DRAFT 5	June 17, 2008	Arshad Noor	- Moved non-normative sections to their own document. - KeyClass element was added to KeyCachePolicy. - KeyCachePolicy is now embedded inside a KeyCachePolicyResponse.
DRAFT 6	July 7, 2008	Arshad Noor	- Modified Permissions object to include all sub-elements on a mandatory basis. - Modified all abbreviations in elements to expand to full names.
PR 1	July 22, 2008	Arshad Noor	- Modified Title page information to conform with OASIS standards. - Brought some Background information into this document from the information document, into Section 2. - Added a Conformance section to Section 4.
PR 2	11/04/08	Arshad Noor	- Added SymkeyWorkInProgress, SymkeyRequestID and RequestCheckInterval elements to support sending and receiving request/responses asynchronously. - Modified SymkeyResponse to include SymkeyWorkInProgress as a valid response to a key request. - Modified Symkey to include the SymkeyRequestID to support asynchronous request/responses. - Modified SymkeyError to include the SymkeyRequestID to support asynchronous request/responses. - Modified SymkeyRequest to send a SymkeyRequestID to poll the SKS server on the status of a symmetric-key request.. - Added Standard Error Codes & Messages in Appendix C. - Added Appendix D to define the process for vendors to apply for a reserved block of codes.
PR 2 (2.1)	11/11/08	Arshad Noor	Typographical corrections in Sections 3.3, 3.8 and 3.11
PR 2 (2.2)	11/18/08	Arshad Noor	- Added X509EncryptionCertificateType and X509EncryptionCertificate element to support sending a PKI X509-compliant digital certificate from the client to the server for encrypting the symmetric-key payload in the response
PR 3 (2.3)	08/05/10	Anil Saldhana	- Made the requirement for SOAP/WSS optional as per TC decision. SOAP/WSS will go into a separate profile of SKSML.

Appendix C. SKMS Error Codes and Messages

The OASIS EKMI TC has determined that it is useful to standardize on the structure and content of error and informational messages within SKSML so that implementations and their users are clear about the problem they are dealing within EKMIs.

Structure

The following structure is proposed for Symmetric Key Management System (SKMS) messages. The message will consist of a three-part string, with each part separated from the others by a hyphen ("-"). The 3-part message consists of the following:

- The first part is the fixed string "**SKMS**"
- The second part is a fixed string consisting of one of the following choices:
  - **ERR**
  - **MSG**
- The third, and last, part is a 5-digit integer identifying the message

Thus, an SKMS error message might look like the following: **SKMS-ERR-NNNNN**; and an SKMS informational message might look like the following: **SKMS-MSG-NNNNN**

Five-digit Codes

The 5-digit integer is divided into the following groups to ensure consistency amongst implementations:

- 00001 – 10000 Reserved for OASIS EKMI TC use (as described below);
- 10001 – 99999 Reserved for vendor implementations of SKSML on a first-come, first-served basis (the process is described below);

SKMS Standard Code-ranges

The 5-digit code range reserved for OASIS EKMI TC SKMS standards use will be reserved as follows:

Code-range	Reserved for
00001 - 00099	Authentication related errors and messages
00100 - 00199	Authorization related errors and messages
00200 - 00299	Cryptographic-module related errors and messages
00300 - 00399	Key-cache and KeyCachePolicy related errors and messages
00400 - 00499	Key-usage and KeyUsePolicy related errors and messages
00500 - 00599	Symmetric Key Client Library related errors and messages
00600 - 00699	Symmetric Key Services server related errors and messages
00700 - 00799	Request checking related errors and messages
00800 - 00899	Miscellaneous errors and messages

Code-range	Reserved for
00900 - 10000	Future OASIS EKMI TC use

Note: The {0} symbol at the end of each message is a placeholder for a parameter that can be used by implementations for adding additional information pertaining to the error. The additional information will be useful to administrators and software developers in helping them focus on the part of the system where the underlying problem has manifested itself.

Authentication ERROR Codes (00001 - 00099)

Code	Message
SKMS-ERR-00001	Authentication failure – invalid signature: {0}
SKMS-ERR-00002	Authentication failure – invalid status: {0}
SKMS-ERR-00003	Authentication failure – unverifiable certificate: {0}
SKMS-ERR-00004	Authentication failure – expired certificate: {0}
SKMS-ERR-00005	Authentication failure – revoked certificate: {0}
SKMS-ERR-00006	Authentication failure – revoked certificate issuer: {0}
SKMS-ERR-00007	Authentication failure – missing certificate: {0}
SKMS-ERR-00008	Authentication failure – missing certificate keyUsage: {0}
SKMS-ERR-00009	Authentication failure – missing certificate crlDistributionPoint: {0}
SKMS-ERR-00010	Authentication failure – missing certificate authorityInfoAccess: {0}
SKMS-ERR-00011	Authentication failure – invalid certificate Subject DN: {0}
SKMS-ERR-00012	Authentication failure – invalid certificate Validity: {0}
SKMS-ERR-00013	Authentication failure – invalid certificate keyUsage: {0}
SKMS-ERR-00014	Authentication failure – invalid certificate crlDistributionPoint: {0}
SKMS-ERR-00015	Authentication failure – invalid certificate authorityInfoAccess: {0}
SKMS-ERR-00016	Authentication failure – unreachable certificate crlDistributionPoint: {0}
SKMS-ERR-00017	Authentication failure – unreachable certificate authorityInfoAccess: {0}
SKMS-ERR-00099	Authentication failure – other authentication error: {0}

Authorization ERROR Codes (00100 - 00199)

Code	Message
SKMS-ERR-00101	Authorization failure – invalid signature: {0}
SKMS-ERR-00102	Authorization failure – invalid status: {0}
SKMS-ERR-00103	Authorization failure – invalid requester: {0}

Code	Message
SKMS-ERR-00104	Authorization failure – invalid request: {0}
SKMS-ERR-00105	Authorization failure – invalid identifier: {0}
SKMS-ERR-00106	Authorization failure – invalid keyclass: {0}
SKMS-ERR-00107	Authorization failure – invalid policy: {0}
SKMS-ERR-00108	Authorization failure – invalid role: {0}
SKMS-ERR-00109	Authorization failure – invalid application: {0}
SKMS-ERR-00110	Authorization failure – invalid date: {0}
SKMS-ERR-00111	Authorization failure – invalid day: {0}
SKMS-ERR-00112	Authorization failure – invalid duration: {0}
SKMS-ERR-00113	Authorization failure – invalid level: {0}
SKMS-ERR-00114	Authorization failure – invalid location: {0}
SKMS-ERR-00115	Authorization failure – invalid number of transactions: {0}
SKMS-ERR-00116	Authorization failure – invalid time: {0}
SKMS-ERR-00117	Authorization failure – invalid use: {0}
SKMS-ERR-00118	Authorization failure – unauthorized access: {0}
SKMS-ERR-00119	Authorization failure – unauthorized application: {0}
SKMS-ERR-00120	Authorization failure – unauthorized date: {0}
SKMS-ERR-00121	Authorization failure – unauthorized day: {0}
SKMS-ERR-00122	Authorization failure – unauthorized duration: {0}
SKMS-ERR-00123	Authorization failure – unauthorized level: {0}
SKMS-ERR-00124	Authorization failure – unauthorized location: {0}
SKMS-ERR-00125	Authorization failure – unauthorized number of transactions: {0}
SKMS-ERR-00126	Authorization failure – unauthorized time: {0}
SKMS-ERR-00127	Authorization failure – unauthorized use: {0}
SKMS-ERR-00118	Authorization failure – other authorization error: {0}

Cryptographic-module ERROR Codes (00200 - 00299)

Code	Message
SKMS-ERR-00201	Cryptographic-module failure – invalid signature: {0}
SKMS-ERR-00202	Cryptographic-module failure – invalid status: {0}
SKMS-ERR-00203	Cryptographic-module failure – invalid cryptographic provider: {0}

Code	Message
SKMS-ERR-00204	Cryptographic-module failure – invalid algorithm: {0}
SKMS-ERR-00205	Cryptographic-module failure – invalid initialization vector: {0}
SKMS-ERR-00206	Cryptographic-module failure – invalid padding: {0}
SKMS-ERR-00207	Cryptographic-module failure – invalid key-size: {0}
SKMS-ERR-00208	Cryptographic-module failure – invalid password or PIN: {0}
SKMS-ERR-00209	Cryptographic-module failure – missing cryptographic provider: {0}
SKMS-ERR-00210	Cryptographic-module failure – missing cryptographic module: {0}
SKMS-ERR-00211	Cryptographic-module failure – missing password or PIN: {0}
SKMS-ERR-00212	Cryptographic-module failure – missing private key: {0}
SKMS-ERR-00213	Cryptographic-module failure – missing digital certificate: {0}
SKMS-ERR-00214	Cryptographic-module failure – missing certificate chain: {0}
SKMS-ERR-00215	Cryptographic-module failure – failed to sign: {0}
SKMS-ERR-00216	Cryptographic-module failure – failed to verify: {0}
SKMS-ERR-00217	Cryptographic-module failure – failed to encrypt: {0}
SKMS-ERR-00218	Cryptographic-module failure – failed to decrypt: {0}
SKMS-ERR-00219	Cryptographic-module failure – failed to digest (hash): {0}
SKMS-ERR-00220	Cryptographic-module failure – failed to generate key: {0}
SKMS-ERR-00299	Cryptographic-module failure – other cryptographic-module error: {0}

4249

4250 Key-cache and KeyCachePolicy ERROR Codes (00300 - 00399)

Code	Message
SKMS-ERR-00301	Key-cache failure – invalid signature: {0}
SKMS-ERR-00302	Key-cache failure – invalid status: {0}
SKMS-ERR-00303	Key-cache failure – invalid cache: {0}
SKMS-ERR-00304	Key-cache failure – invalid policy: {0}
SKMS-ERR-00305	Key-cache failure – missing policy: {0}
SKMS-ERR-00306	Key-cache failure – expired policy: {0}
SKMS-ERR-00307	Key-cache failure – unauthorized create-access to cache: {0}
SKMS-ERR-00308	Key-cache failure – unauthorized read-access to cache: {0}
SKMS-ERR-00309	Key-cache failure – unauthorized write-access to cache: {0}
SKMS-ERR-00399	Key-cache failure – other key-cache error: {0}

Key-usage and KeyUsePolicy ERROR Codes (00400 - 00499)

Code	Message
SKMS-ERR-00401	Key-usage failure – invalid signature: {0}
SKMS-ERR-00402	Key-usage failure – invalid status: {0}
SKMS-ERR-00403	Key-usage failure – invalid policy: {0}
SKMS-ERR-00404	Key-usage failure – invalid application: {0}
SKMS-ERR-00405	Key-usage failure - expired policy: {0}
SKMS-ERR-00406	Key-usage failure – missing policy: {0}
SKMS-ERR-00407	Key-usage failure - missing library: {0}
SKMS-ERR-00499	Key-usage failure – other key-usage error: {0}

Symmetric Key Client Library ERROR Codes (00500 - 00599)

Code	Message
SKMS-ERR-00501	SKCL error - invalid signature: {0}
SKMS-ERR-00502	SKCL error - invalid status: {0}
SKMS-ERR-00503	SKCL error - invalid parameter: {0}
SKMS-ERR-00504	SKCL error - invalid domain ID: {0}
SKMS-ERR-00505	SKCL error - invalid server ID: {0}
SKMS-ERR-00506	SKCL error - invalid key ID: {0}
SKMS-ERR-00507	SKCL error - invalid request ID: {0}
SKMS-ERR-00508	SKCL error - invalid key-class: {0}
SKMS-ERR-00509	SKCL error - invalid server URL: {0}
SKMS-ERR-00510	SKCL error - invalid plaintext: {0}
SKMS-ERR-00511	SKCL error - invalid ciphertext: {0}
SKMS-ERR-00512	SKCL error - missing parameter: {0}
SKMS-ERR-00513	SKCL error - missing domain ID: {0}
SKMS-ERR-00514	SKCL error - missing server ID: {0}
SKMS-ERR-00515	SKCL error - missing key ID: {0}
SKMS-ERR-00516	SKCL error - missing request ID: {0}
SKMS-ERR-00517	SKCL error - missing key-class: {0}
SKMS-ERR-00518	SKCL error - missing server URL: {0}

Code	Message
SKMS-ERR-00519	SKCL error - missing plaintext: {0}
SKMS-ERR-00520	SKCL error - missing ciphertext: {0}
SKMS-ERR-00521	SKCL error – SOAP layer error: {0}
SKMS-ERR-00522	SKCL error - network layer error: {0}
SKMS-ERR-00523	SKCL error – database layer error: {0}
SKMS-ERR-00599	SKCL error - other SKCL error: {0}

4255

4256 ## Symmetric Key Services server ERROR Codes (00600 - 00699)

Code	Message
SKMS-ERR-00601	SKS error - invalid signature: {0}
SKMS-ERR-00602	SKS error - invalid status: {0}
SKMS-ERR-00603	SKS error - invalid parameter: {0}
SKMS-ERR-00604	SKS error - invalid domain ID: {0}
SKMS-ERR-00605	SKS error - invalid server ID: {0}
SKMS-ERR-00606	SKS error - invalid key ID: {0}
SKMS-ERR-00607	SKS error - invalid request ID: {0}
SKMS-ERR-00608	SKS error - invalid key-class: {0}
SKMS-ERR-00609	SKS error - invalid server URL: {0}
SKMS-ERR-00610	SKS error - invalid plaintext: {0}
SKMS-ERR-00611	SKS error - invalid ciphertext: {0}
SKMS-ERR-00612	SKS error - missing parameter: {0}
SKMS-ERR-00613	SKS error - missing domain ID: {0}
SKMS-ERR-00614	SKS error - missing server ID: {0}
SKMS-ERR-00615	SKS error - missing key ID: {0}
SKMS-ERR-00616	SKS error - missing request ID: {0}
SKMS-ERR-00617	SKS error - missing key-class: {0}
SKMS-ERR-00618	SKS error - missing server URL: {0}
SKMS-ERR-00619	SKS error - missing default key-cache policy: {0}
SKMS-ERR-00620	SKS error - missing default key-use policy: {0}
SKMS-ERR-00621	SKS error – SOAP layer error: {0}
SKMS-ERR-00622	SKS error - network layer error: {0}

Code	Message
SKMS-ERR-00623	SKS error – database layer error: {0}
SKMS-ERR-00699	SKS error - other SKS error: {0}

4257

4258 ## Request checking ERROR Codes (00700 - 00799)

Code	Message
SKMS-ERR-00701	Request-check error - invalid signature: {0}
SKMS-ERR-00702	Request-check error - invalid status: {0}
SKMS-ERR-00703	Request-check error - invalid parameter: {0}
SKMS-ERR-00704	Request-check error - invalid domain ID: {0}
SKMS-ERR-00705	Request-check error - invalid server ID: {0}
SKMS-ERR-00706	Request-check error - invalid polling frequency: {0}
SKMS-ERR-00707	Request-check error - invalid request ID: {0}
SKMS-ERR-00799	Request-check error - other request-check error: {0}

4259

4260 ## Miscellaneous ERROR Codes (00800 - 00899)

Code	Message
SKMS-ERR-00801	Miscellaneous error - invalid signature: {0}
SKMS-ERR-00802	Miscellaneous error - invalid status: {0}
SKMS-ERR-00899	Miscellaneous error - other miscellaneous error: {0}

Appendix D. Process for requesting a block of SKSML Error Codes for Vendor Use

Vendors who choose to implement SKSML will be able to apply for a unique block of 1,000 code numbers to be assigned for their exclusive use within their implementation of the SKSML protocol. They may choose to use these numbers for messages related to their implementation, **in addition** to the Standard SKMS Codes & Messages.

In order to receive this unique block of code-numbers, the Vendor must use the following process:

1. An authorized representative of the Vendor must send an e-mail to the OASIS EKMI TC (using the guidelines in this URL: http://www.oasis-open.org/committees/comments/index.php?wg_abbrev=ekmi) asserting the following:
   a) That they intend to implement the SKSML 1.0 specification within 6-12 months of their dated e-mail;
   b) That they will implement ALL the Standard Codes & Messages as described in this document, in their implementation;
   c) That they will not duplicate ANY Standard Code-message within their assigned private-block of numbers;
   d) That if the TC later chooses to standardize a specific message within the Standard Codes, that may overlap with a Member's private-block message, the Member will use the Standard Code in implementations created subsequent to the standardization of the code/message;
   e) That they will notify this TC of the release date of their product, with the relevant section of their documentation pointing to the use of the Standard Codes in their product
2. The OASIS EKMI TC Chair or Secretary will verify that the e-mail contains all assertions;
3. The OASIS EKMI TC Chair or Secretary will setup a ballot for the TC to vote to issue the next available block of 1,000 code-numbers to the requesting Vendor. (The first OASIS Member to apply will receive the numbers 10001 – 11000; the next Vendor request will receive 11001 – 12000 and so on).
4. Upon the conclusion of the vote, the TC Chair or Secretary will notify the Vendor of the assigned block of code-numbers (copying the TC);
5. The TC Chair or Secretary will update a web-page on the TC's home-page with the following information:
   a) The name of the Vendor
   b) A link to the e-mail request from the Vendor
   c) A link to the TC ballot authorizing the assignment
   d) The date of assignment and
   e) The assigned block of code-numbers
6. When the Vendor releases the product, they will notify the OASIS TC of the product, a link to the product's web URL and a link to the specific section of their documentation high-lighting the Standard Codes & Messages, as well as the Private Block Codes & Messages;
7. The TC Chair or Secretary will then update the above-mentioned web-page to now include this additional information:
   a) The web-link to the Vendor's implementation
   b) The web-link to the documentation highlighting the Codes and Messages
8. In the event that a Vendor does not implement the protocol within the 6-12 month period, the TC will vote and reclaim the private-block assigned to the original requester. Any use of the private-

block of assigned numbers, after the TC has voted to reclaim them, will be a violation of the TC's guidelines for the SKSML Specification. OASIS Administration will then be directed by this TC to take whatever action it is legally permitted to address this violation.