[Symmetric Key Services Markup Language (SKSML) Version 1.0 Non-Normative DRAFT 6]

OASIS Technical Committee DRAFT

24 June 2008

Specification URIs:

This Version:

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

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

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

Previous Version:

None

Latest Version:

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

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

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

Latest Approved Version:

None

Technical Committee:

OASIS Enterprise Key Management Infrastructure (EKMI) TC

Chair(s):

Arshad Noor, StrongAuth, Inc. (arshad.noor@strongauth.com)

Editor(s):

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

Related Work:

This specification replaces or supercedes:

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

This specification is related to:

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

Declared XML Namespace(s):

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

Abstract:

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

Status:

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

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

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

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

Notices

Table of Contents

1 Introduction

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

1.1 Terminology

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

1.2 Glossary

3DES – Triple Data Encryption Standard

AES – Advanced Encryption Standard

Base64 – An encoding scheme for representing data

Ciphertext – Encrypted data

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

DES – Data Encryption Standard

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

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

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

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

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

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

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

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

Plaintext – Unencrypted data

SHA – Secure Hashing Algorithm

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

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

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

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

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

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

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

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

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

SOAP – Simple Object Access Protocol

SOAP Body – The content part of a SOAP message

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

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

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

Symkey - A symmetric encryption key

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

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

1.3 Normative References

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

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

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

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

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

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

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

1.4 Non-normative References

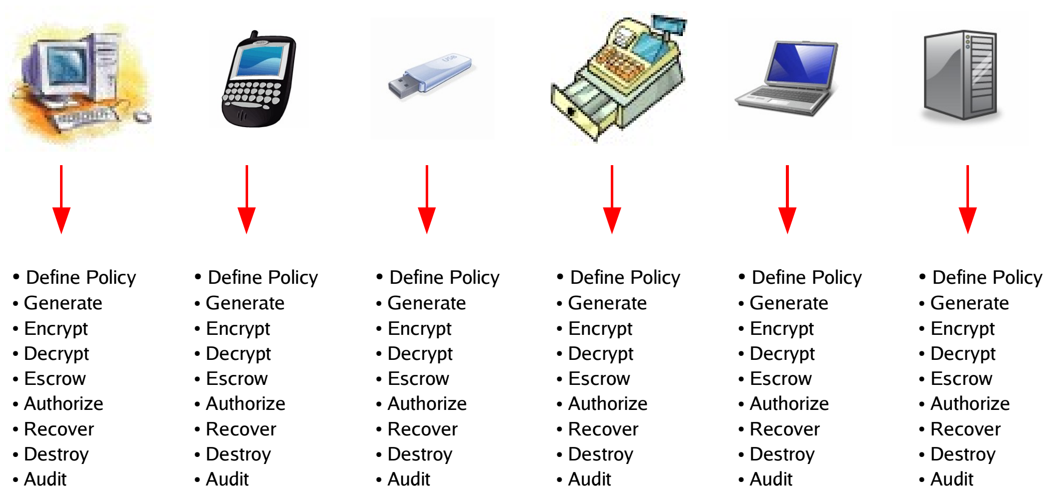
[IDtrust-EKMI] A. Noor, *Securing the core with an EKMI.* NIST-IDtrust 2008, March 2008.
http://middleware.internet2.edu/idtrust/2008/papers/07-noor-ekmi.pdf

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

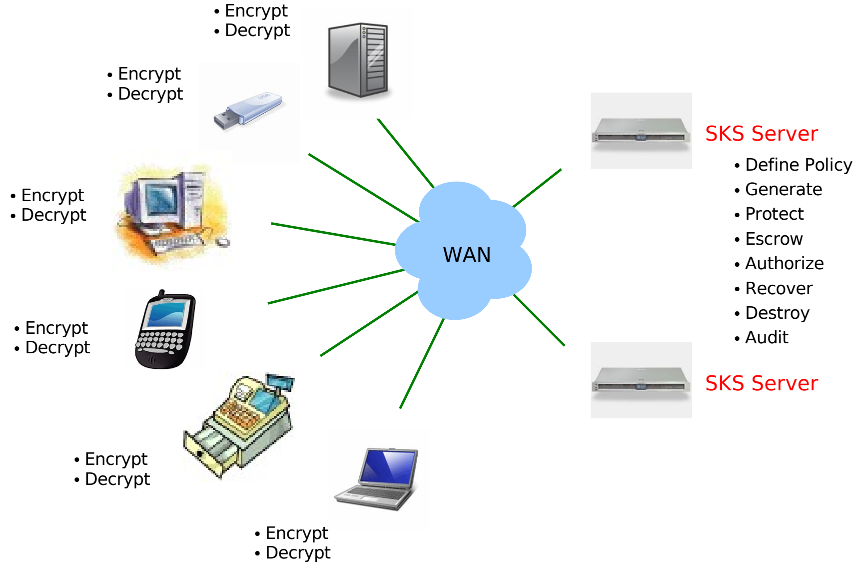
In a rush to provide solutions to the market, vendors have created many device-specific, platform-specific, database-specific and application-specific encryption and key-management tools. While these tools may be capable of performing their stated tasks adequately, a typical enterprise would have to 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 SOAP and XML for encapsulating its requests and responses and can thus, be used on any platform that supports these two underlying protocols;
- 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 the Web Services Security (WSS) standard 1.0, which in turn supports the use of XML Signature and XML Encryption within the SOAP Header. Relying only the on the WSS profile that uses

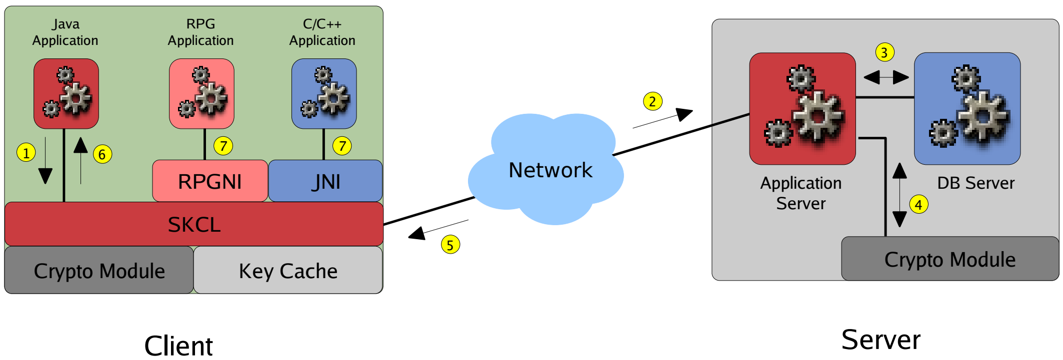
RSA cryptographic key-pairs and digital certificates, SKSML uses the digital signatures for authenticity and message-integrity, while using RSA-encryption for confidentiality;

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

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

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

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

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

While the ***SymkeyRequest*** element is very simple, the Web Service Security (WSS) envelope – which provides security for all SKSML messages – expands the size of the message. The same request shown above, is displayed below in its entirety, with its WSS envelope. Please note that some content – such as Base64-encoded binary content - has been reformatted for aesthetics and clarity of the XML elements. The actual elements and data-types have been preserved from actual SKSML messages.

For an interpretation of the XML elements shown below, please refer to **[WSS]**.

For the sake of brevity, this specification will dispense with showing the SOAP envelope and the WSS elements in all other examples, when discussing SKSML.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header>
       <wsse:Security xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" SOAP-
ENV:mustUnderstand="1">
       <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
      EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary"
      ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509v3" wsu:Id="XWSSGID-1172790302111-1738806553">
      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=
      </wsse:BinarySecurityToken>
       <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
         <ds:SignedInfo>
              <ds:CanonicalizationMethod
                 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
              <InclusiveNamespaces
                 xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"
                 PrefixList="wsse SOAP-ENV"/>
            </ds:CanonicalizationMethod>
           <ds:SignatureMethod
                 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
           <ds:Reference URI="#XWSSGID-1172790300636-653454040">
           <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
              <ds:DigestValue>lU4m+rp4oebgl9g+t3nRaZYqUlE=</ds:DigestValue>
           </ds:Reference>
           <ds:Reference URI="#XWSSGID-11727903006377088718O5">
           <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
              <ds:DigestValue>WCpOmTCbffcEHXhGf5rlEYWLrZg=</ds:DigestValue>
           </ds:Reference>
         </ds:SignedInfo>
         <ds:SignatureValue>
         svStAvBRRrF+g2biPl7uWHkJTQPIl8t4phMbOZQsZlQcn36tcMSj/a4+4LPNfOB3Y8yO2lr1Oa1
         fGqCPAWZNuEH34VQEM196rRwV258mgp8uwpXEYJIgPJqg89w8+/NdaODccLQ2Bizu7QM/HSM2ab
         ogNJwqmbSyIazOsnOcU=
         </ds:SignatureValue>
         <ds:KeyInfo>
              <wsse:SecurityTokenReference xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
           wsu:Id="XWSSGID-1172790300633-442423344">
```

```
            <wsse:Reference URI="#XWSSGID-1172790302111-1738806553"
                  ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3"/>
            </wsse:SecurityTokenReference>
         </ds:KeyInfo>
      </ds:Signature>
      <wsu:Timestamp xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="XWSSGID-11727903006377088718O5">
         <wsu:Created>2007-03-01T23:05:00Z</wsu:Created>
         <wsu:Expires>2007-03-01T23:05:05Z</wsu:Expires>
      </wsu:Timestamp>
   </wsse:Security>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd" wsu:Id="XWSSGID-1172790300636-653454040">
      <ekmi:SymkeyRequest
         xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01">
         <ekmi:GlobalKeyID>10514-0-0</ekmi:GlobalKeyID>
      </ekmi:SymkeyRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

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. (The SOAP message, as indicated earlier, is secured using WSS, but only the actual SKSML content is displayed and discussed here).

```
[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:GlobalKeyID>10514-1-235</ekmi:GlobalKeyID>
[b06]       <ekmi:KeyUsePolicy>
[b07]          <ekmi:KeyUsePolicyID>10514-4</ekmi:KeyUsePolicyID>
[b08]          <ekmi:PolicyName>DES-EDE KeyUsePolicy</ekmi:PolicyName>
[b09]          <ekmi:KeyClass>HR-Class</ekmi:KeyClass>
[b10]          <ekmi:KeyAlgorithm>
[b11]             http://www.w3.org/2001/04/xmlenc#tripledes-cbc
[b12]          </ekmi:KeyAlgorithm>
[b13]          <ekmi:KeySize>192</ekmi:KeySize>
[b14]          <ekmi:Status>Active</ekmi:Status>
[b15]          <ekmi:Permissions>
[b16]             <ekmi:PermittedApplications ekmi:any=”false”>
[b17]                   <ekmi:PermittedApplication>
[b18]                   <ekmi:ApplicationID>10514-23</ekmi:ApplicationID>
[b19]                   <ekmi:ApplicationName>
[b20]                         Payroll Application
[b21]                   </ekmi:ApplicationName>
[b22]                   <ekmi:ApplicationVersion>1.0</ekmi:ApplicationVersion>
[b23]                   <ekmi:ApplicationDigestAlgorithm>
[b24]                      http://www.w3.org/2000/09/xmldsig#sha1
[b25]                   </ekmi:ApplicationDigestAlgorithm>
[b26]                   <ekmi:ApplicationDigestValue>
[b27]                      NIG4bKkt4cziEqFFuOoBTM81efU=
[b28]                   </ekmi:ApplicationDigestValue>
```

```
[b29]                   </ekmi:PermittedApplication>
[b30]                </ekmi:PermittedApplications>
[b31]                <ekmi:PermittedDates ekmi:any=”false”>
[b32]                   <ekmi:PermittedDate>
[b33]                      <ekmi:StartDate>2008-01-01</ekmi:StartDate>
[b34]                      <ekmi:EndDate>2008-12-31</ekmi:EndDate>
[b35]                   </ekmi:PermittedDate>
[b36]                </ekmi:PermittedDates>
[b37]                <ekmi:PermittedDays ekmi:any=”true” xsi:nil=”true”/>
[b38]                <ekmi:PermittedDuration ekmi:any=”true” xsi:nil=”true”/>
[b39]                <ekmi:PermittedLevels ekmi:any=”true” xsi:nil=”true”/>
[b40]                <ekmi:PermittedLocations ekmi:any=”true” xsi:nil=”true”/>
[b41]                <ekmi:PermittedNumberOfTransactions ekmi:any=”true”
xsi:nil=”true”/>
[b42]                <ekmi:PermittedTimes ekmi:any=”false”>
[b43]                   <ekmi:PermittedTime>
[b44]                      <ekmi:StartTime>07:00:00</ekmi:StartTime>
[b45]                      <ekmi:EndTime>19:00:00</ekmi:EndTime>
[b46]                   </ekmi:PermittedTime>
[b47]                </ekmi:PermittedTimes>
[b48]                <ekmi:PermittedUses ekmi:any=”true” xsi:nil=”true”/>
[b49]             </ekmi:Permissions>
[b50]          </ekmi:KeyUsePolicy>
[b51]             <ekmi:EncryptionMethod
[b52]                   Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[b53]             <xenc:CipherData>
[b54]                <xenc:CipherValue>
[b55]             E9zWB/y93hVSzeTLiDcQoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJM
[b56]             hQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/2fw
[b57]              UJow/qimxi8+huUYJMtaGHtXuLlWtx27STRcRpIsY=
[b58]                </xenc:CipherValue>
[b59]              </xenc:CipherData>
[b60]       </ekmi:Symkey>
[b61]  </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 ***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.

[b06] 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.

[b07] identifies the unique ***KeyUsePolicyID*** *(KUPID)* which identifies this policy within the SKMS.

[b08] provides a descriptive name for this key-use policy, which is helpful to human readers when identifying this policy.

[b09] 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.

[b10] is the start tag of the ***KeyAlgorithm*** element.

[b11] identifies the cryptographic algorithm that this symmetric key must be used with to for cryptographic operations.

[b12] is the closing tag of the ***KeyAlgorithm*** element.

[b13] 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.

[b14] 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.

[b15] 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.

[b16] 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..

[b17] 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.***

[b18] identifies the unique ***ApplicationID*** (as identified within the SKMS) of the ***PermittedApplication***.

[b19] is the start of the ***ApplicationName*** element.

[b20] identifies the ***ApplicationName*** of the ***PermittedApplication*** (as identified within the SKMS).

[b21] is the closing tag of the ***ApplicationName*** element.

[b22] 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.

[b23] 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.

[b24] identifies the W3C-specified URL of the cryptographic algorithm used to calculate the message digest of the application's image.

[b25] is the closing tag of the ***ApplicationDigestAlgorithm*** element.

[b26] 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.

[b27] identifies the Base64-encoded message digest of the ***PermittedApplication's*** image, based on the algorithm specified in the ***ApplicationDigestAlgorithm*** element.

[b28] is the closing tag of the ***ApplicationDigestValue*** element.

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

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

[b31] 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.

[b32] is the start of the first ***PermittedDate*** element. There can be any number of ***PermittedDate*** elements within a ***PermittedDates*** element.

[b33] identifies the ***StartDate*** of the duration period during which the symmetric encryption key can be used by authorized applications.

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

[b35] is the closing tag of the ***PermittedDate*** element.

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

[b37] 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.

[b38] 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.

[b39] 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.

[b40] 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.

[b41] 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.

[b42] 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.

[b43] is the start of the first ***PermittedTime*** element. There can be any number of ***PermittedTime*** elements within a ***PermittedTimes*** element.

[b44] identifies the ***StartTime*** of the duration period during which the symmetric encryption key can be used by authorized applications.

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

[b46] is the closing tag of the ***PermittedTime*** element.

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

[b48] 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.

[b49] is the closing tag of the ***Permissions*** element.

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

[b51]-[b52] 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.

[b53] 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).

[b54] is the start of the ***CipherValue*** element. This element contains the Base64-encoded ciphertext of the symmetric encryption key.

[b55] – **[b57]** is the Base64-encoded ciphertext of the symmetric encryption key.

[b58] is the closing tag of the ***CipherValue*** element.

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

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

[b61] 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 (linked to the call SKCL) will 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. (The SOAP message is secured using WSS, but only the actual SKSML content is displayed and discussed here).

```
[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 <u>non-zero values</u> 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 within a WSS envelope. 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 within a WSS envelope. 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. Additionally, the SOAP message, as indicated earlier, is secured using WSS, but only the actual SKSML content is displayed and discussed here.

```
[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:GlobalKeyID>10514-4-3792</ekmi:GlobalKeyID>
[h06]         <ekmi:KeyUsePolicy>
[h07]            <ekmi:KeyUsePolicyID>10514-9</ekmi:KeyUsePolicyID>
[h08]            <ekmi:PolicyName>DES-EDE KeyUsePolicy for EHR-CDC</ekmi:PolicyName>
[h09]            <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
[h10]            <ekmi:KeyAlgorithm>
[h11]               http://www.w3.org/2001/04/xmlenc#tripledes-cbc
[h12]            </ekmi:KeyAlgorithm>
[h13]            <ekmi:KeySize>192</ekmi:KeySize>
[h14]            <ekmi:Status>Active</ekmi:Status>
[h15]            <ekmi:Permissions>
[h16]               <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
[h17]               <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
[h18]               <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
[h19]               <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
[h20]               <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
```

```
[h21]                   <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
[h22]                   <ekmi:PermittedNumberOfTransactions
[h23]                             ekmi:any="true" xsi:nil="true"/>
[h24]                   <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
[h25]                   <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
[h26]                </ekmi:Permissions>
[h27]           </ekmi:KeyUsePolicy>
[h28]           <ekmi:EncryptionMethod
[h29]                   Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[h30]           <xenc:CipherData>
[h31]                <xenc:CipherValue>
[h32]                E9zWB/y93hVSzeTLiDcQoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJMa1w
[h33]                hQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxjfg1pU8tlWtx27STRcR/2fw2ava
[h34]                 UlWtx27STRcRJMtaGHtXuLlWtx27STRcRpIsY=
[h35]                </xenc:CipherValue>
[h36]            </xenc:CipherData>
[h37]         </ekmi:Symkey>
[h38]         <ekmi:Symkey>
[h39]             <ekmi:GlobalKeyID>10514-4-3793</ekmi:GlobalKeyID>
[h40]             <ekmi:KeyUsePolicy>
[h41]                <ekmi:KeyUsePolicyID>10514-12</ekmi:KeyUsePolicyID>
[h42]                <ekmi:PolicyName>DES-EDE KeyUsePolicy for EHR-CRO</ekmi:PolicyName>
[h43]                    <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
[h44]                    <ekmi:KeyAlgorithm>
[h45]                      http://www.w3.org/2001/04/xmlenc#tripledes-cbc
[h46]                   </ekmi:KeyAlgorithm>
[h47]                <ekmi:KeySize>192</ekmi:KeySize>
[h48]                <ekmi:Status>Active</ekmi:Status>
[h49]                <ekmi:Permissions>
[h50]                   <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
[h51]                   <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
[h52]                       <ekmi:PermittedDate>
[h53]                           <ekmi:StartDate>2008-01-01</ekmi:StartDate>
[h54]                           <ekmi:EndDate>2009-12-31</ekmi:EndDate>
[h55]                       </ekmi:PermittedDate>
[h56]                    </ekmi:PermittedDates>
[h57]                   <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
[h58]                   <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
[h59]                   <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
[h60]                   <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
[h61]                   <ekmi:PermittedNumberOfTransactions
[h62]                             ekmi:any="true" xsi:nil="true"/>
[h63]                   <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
[h64]                   <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
[h65]                </ekmi:Permissions>
[h66]           </ekmi:KeyUsePolicy>
[h67]           <ekmi:EncryptionMethod
[h68]                   Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[h69]           <xenc:CipherData>
[h70]                <xenc:CipherValue>
[h71]                qUiQXGOca8EU871zBoXBjDoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJM1
[h72]                hQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/2fw1
[h73]                 UJow/qimxi8+ huUYJMtaGHtXuLlWtx27STRcRpIsY=
[h74]                </xenc:CipherValue>
[h75]            </xenc:CipherData>
[h76]          </ekmi:Symkey>
[h77]          <ekmi:Symkey>
[h78]             <ekmi:GlobalKeyID>10514-4-3795</ekmi:GlobalKeyID>
                     ...
[h79]                    <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
```

```
                ...
[h80]       </ekmi:Symkey>
[h81]       <ekmi:Symkey>
[h82]         <ekmi:GlobalKeyID>10514-4-3797</ekmi:GlobalKeyID>
                ...
[h83]               <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
                ...
[h84]       </ekmi:Symkey>
[h85]       <ekmi:Symkey>
[h86]         <ekmi:GlobalKeyID>10514-4-3798</ekmi:GlobalKeyID>
                ...
[h87]               <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
                ...
[h88]       </ekmi:Symkey>
[h89]       <ekmi:Symkey>
[h90]         <ekmi:GlobalKeyID>10514-4-3799</ekmi:GlobalKeyID>
                ...
[h91]               <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
                ...
[h92]       </ekmi:Symkey>
[h93]       <ekmi:Symkey>
[h94]         <ekmi:GlobalKeyID>10514-4-3801</ekmi:GlobalKeyID>
                ...
[h95]               <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
                ...
[h96]       </ekmi:Symkey>
[h97]       <ekmi:Symkey>
[h98]         <ekmi:GlobalKeyID>10514-4-3803</ekmi:GlobalKeyID>
                ...
[h99]               <ekmi:KeyClass>EHR-PAT</ekmi:KeyClass>
                ...
[h100]      </ekmi:Symkey>
[h101]      <ekmi:Symkey>
[h102]        <ekmi:GlobalKeyID>10514-4-3805</ekmi:GlobalKeyID>
              ...
[h103]              <ekmi:KeyClass>EHR-PHY</ekmi:KeyClass>
                ...
[h104]      </ekmi:Symkey>
[h105]    </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 ***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.

[h06] 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.

[h07] identifies the unique ***KeyUsePolicyID*** *(KUPID)* which identifies this policy within the SKMS.

[h08] provides a descriptive name for this key-use policy, which is helpful to human readers when identifying this policy.

[h09] 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.

[h10] is the start tag of the ***KeyAlgorithm*** element.

[h11] 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]**..

[h12] is the closing tag of the ***KeyAlgorithm*** element.

[h13] specifies the size of the symmetric encryption key in bits. For this Triple-DES key, it is 192-bits.

[h14] 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.

[h15] 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.

[h16] 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.

[h17] 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.

[h18] 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.

[h19] 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.

[h20] 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.

[h21] 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.

[h22] - [h23] 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.

[h24] 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.

[h25] 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.

[h26] is the closing tag of the ***Permissions*** element.

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

[h28] and **[h29]** 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.

[h30] 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).

[h31] is the start of the ***CipherValue*** element. This element contains the Base64-encoded ciphertext of the symmetric encryption key.

[h32] – **[h34]** is the Base64-encoded ciphertext of the symmetric encryption key.

[h35] is the closing tag of the ***CipherValue*** element.

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

[h37] is the closing tag of the first ***Symkey*** element within this ***SymkeyResponse****.*

[h38] - **[h76]** represents the <u>**second**</u> ***Symkey*** element in this ***SymkeyResponse***. The differences in this symmetric key element from the first, can be summarized as follows:

- **[h39]** identifies a different ***GlobalKeyID*** (10514-4-3793) for this symmetric key;
- **[h41]** identifies a different ***KeyUsePolicy*** (10514-12) for this symmetric key;
- **[h43]** identifies a different ***KeyClass*** (EHR-CRO) for this symmetric key;
- **[h49]** - **[h65]** defines a different ***Permissions*** element for this symmetric key;
- **[h71]** - **[h73]** contains a different ***CipherValue*** for this symmetric key;

[h78] – [**h80]** is the container for the <u>**third**</u> ***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.

[h81] – **[h104]** 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.

3.9 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.

```
[i01]    <ekmi:SymkeyResponse
[i02]           xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
[i03]           xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
[i04]        <ekmi:SymkeyError>
[i05]            <ekmi:RequestedGlobalKeyID>10514-2-22</ekmi:RequestedGlobalKeyID>
[i06]            <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
[i07]            <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
[i08]        </ekmi:SymkeyError>
[i09]    </ekmi:SymkeyResponse>
```

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

[i02] and **[i03]** identify the namespaces to which this XML conforms, and the location of their XML Schema Definitions (XSD).

[i04] 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.

[i05] 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.

[i06] 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.

[i07] is the text of the ***ErrorMessage***, localized to the region and language of the requesting client application.

[i08] is the closing tag of the ***SymkeyError*** tag.

[i09] is the closing tag of the ***SymkeyResponse*** tag.

3.10 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.

```
[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:Symkey>
[j05]              <ekmi:GlobalKeyID>10514-4-3792</ekmi:GlobalKeyID>
[j06]            <ekmi:KeyUsePolicy>
[j07]              <ekmi:KeyUsePolicyID>10514-9</ekmi:KeyUsePolicyID>
[j08]                 <ekmi:PolicyName>DES-EDE Policy for EHR-CDC</ekmi:PolicyName>
[j09]                 <ekmi:KeyClass>EHR-CDC</ekmi:KeyClass>
[j10]                 <ekmi:KeyAlgorithm>
[j11]                    http://www.w3.org/2001/04/xmlenc#tripledes-cbc
[j12]                 </ekmi:KeyAlgorithm>
[j13]              <ekmi:KeySize>192</ekmi:KeySize>
[j14]              <ekmi:Status>Active</ekmi:Status>
[j15]              <ekmi:Permissions>
[j16]                 <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
[j17]                 <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
[j18]                 <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
[j19]                 <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
[j20]                 <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
[j21]                 <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
[j22]                 <ekmi:PermittedNumberOfTransactions
[j23]                          ekmi:any="true" xsi:nil="true"/>
[j24]                 <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
[j25]                 <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
```

```
[j26]                 </ekmi:Permissions>
[j27]             </ekmi:KeyUsePolicy>
[j28]             <ekmi:EncryptionMethod
[j29]                     Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[j30]             <xenc:CipherData>
[j31]                 <xenc:CipherValue>
[j32]                 E9zWB/y93hVSzeTLiDcQoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCFjJ
[j33]                 hQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxjfg1pU8tlWtx27STRcR/2fw
[j34]                 UlWtx27STRcRJMtaGHtXuLlWtx27STRcRpIsY=
[j35]                 </xenc:CipherValue>
[j36]              </xenc:CipherData>
[j37]           </ekmi:Symkey>
[j38]           <ekmi:Symkey>
[j39]                 <ekmi:GlobalKeyID>10514-4-3793</ekmi:GlobalKeyID>
[j40]               <ekmi:KeyUsePolicy>
[j41]                 <ekmi:KeyUsePolicyID>10514-12</ekmi:KeyUsePolicyID>
[j42]                    <ekmi:PolicyName>DES-EDE Policy for EHR-CRO</ekmi:PolicyName>
[j43]                    <ekmi:KeyClass>EHR-CRO</ekmi:KeyClass>
[j44]                    <ekmi:KeyAlgorithm>
[j45]                       http://www.w3.org/2001/04/xmlenc#tripledes-cbc
[j46]                    </ekmi:KeyAlgorithm>
[j47]                 <ekmi:KeySize>192</ekmi:KeySize>
[j48]                 <ekmi:Status>Active</ekmi:Status>
[j49]                 <ekmi:Permissions>
[j50]                    <ekmi:PermittedApplications ekmi:any="true" xsi:nil="true"/>
[j51]                    <ekmi:PermittedDates ekmi:any="true" xsi:nil="true"/>
[j52]                        <ekmi:PermittedDate>
[j53]                            <ekmi:StartDate>2008-01-01</ekmi:StartDate>
[j54]                            <ekmi:EndDate>2009-12-31</ekmi:EndDate>
[j55]                        </ekmi:PermittedDate>
[j56]                     </ekmi:PermittedDates>
[j57]                    <ekmi:PermittedDays ekmi:any="true" xsi:nil="true"/>
[j58]                    <ekmi:PermittedDuration ekmi:any="true" xsi:nil="true"/>
[j59]                    <ekmi:PermittedLevels ekmi:any="true" xsi:nil="true"/>
[j60]                    <ekmi:PermittedLocations ekmi:any="true" xsi:nil="true"/>
[j61]                    <ekmi:PermittedNumberOfTransactions
[j62]                              ekmi:any="true" xsi:nil="true"/>
[j63]                    <ekmi:PermittedTimes ekmi:any="true" xsi:nil="true"/>
[j64]                    <ekmi:PermittedUses ekmi:any="true" xsi:nil="true"/>
[j65]                 </ekmi:Permissions>
[j66]             </ekmi:KeyUsePolicy>
[j67]             <ekmi:EncryptionMethod
[j68]                     Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[j69]             <xenc:CipherData>
[j70]                 <xenc:CipherValue>
[j71]                 qUiQXGOca8EU871zBoXBjDoDxmlNxTuxSffMNwCJmt1dIqzQHBnpdQ81g6DKdkCF
[j72]                 hQhywCx9sfYjv9h5FDqUiQXGOca8EU871zBoXBjDxjfg1pU8tGFbpWZcd/ATpJD/
[j73]                 UJow/qimxi8+ huUYJMtaGHtXuLlWtx27STRcRpIsY=
[j74]                 </xenc:CipherValue>
[j75]              </xenc:CipherData>
[j76]            </ekmi:Symkey>
[j77]            <ekmi:Symkey>
[j78]              <ekmi:GlobalKeyID>10514-4-3795</ekmi:GlobalKeyID>
                       ...
[j79]                    <ekmi:KeyClass>EHR-DEF</ekmi:KeyClass>
                       ...
[j80]            </ekmi:Symkey>
[j81]            <ekmi:Symkey>
[j82]              <ekmi:GlobalKeyID>10514-4-3797</ekmi:GlobalKeyID>
                       ...
```

```
[j83]                   <ekmi:KeyClass>EHR-EMT</ekmi:KeyClass>
                      ...
[j84]          </ekmi:Symkey>
[j85]          <ekmi:Symkey>
[j86]            <ekmi:GlobalKeyID>10514-4-3798</ekmi:GlobalKeyID>
                      ...
[j87]                   <ekmi:KeyClass>EHR-HOS</ekmi:KeyClass>
                      ...
[j88]          </ekmi:Symkey>
[j89]          <ekmi:Symkey>
[j90]            <ekmi:GlobalKeyID>10514-4-3799</ekmi:GlobalKeyID>
                      ...
[j91]                   <ekmi:KeyClass>EHR-INS</ekmi:KeyClass>
                      ...
[j92]          </ekmi:Symkey>
[j93]          <ekmi:Symkey>
[j94]            <ekmi:GlobalKeyID>10514-4-3801</ekmi:GlobalKeyID>
                      ...
[j95]                   <ekmi:KeyClass>EHR-NUR</ekmi:KeyClass>
                      ...
[j96]          </ekmi:Symkey>
[j97]          <ekmi:SymkeyError>
[j98]            <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
[j99]            <ekmi:RequestedKeyClass>EHR-PAT</ekmi:RequestedKeyClass>
[j100]           <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
[j101]           <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
[j102]        </ekmi:SymkeyError>
[j103]        <ekmi:SymkeyError>
[j104]           <ekmi:RequestedGlobalKeyID>10514-0-0</ekmi:RequestedGlobalKeyID>
[j105]           <ekmi:RequestedKeyClass>EHR-PHY</ekmi:RequestedKeyClass>
[j106]           <ekmi:ErrorCode>SKS-100004</ekmi:ErrorCode>
[j107]           <ekmi:ErrorMessage>Unauthorized request for key</ekmi:ErrorMessage>
[j108]        </ekmi:SymkeyError>
[j109]    </ekmi:SymkeyResponse>
```

[j01] – **[j96]** is no different from the response shown in ***Section 3.8 Response with multiple new symmetric keys***.

[j97] - **[j102]** 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***. However, there is one difference in this element on line [j79]. This element includes the ***RequestedKeyClass*** of the request.

[j103] - **[j108]** identifies the second ***SymkeyError*** returned by the SKS server. It is similar to the error described in lines [j77] through [j82] including the ***RequestedKeyClass*** of the request.

[j109] is the closing tag of the ***SymkeyResponse*** tag.

3.11 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:

```
[k01]  <ekmi:KeyCachePolicyRequest
[k02]      xmlns:ekmi="http://docs.oasis-open.org/ekmi/2008/01"/>
```

[k01] is the start of the ***KeyCachePolicyRequest*** element.

[k02] identifies the namespace to which this XML conforms, and the location of its XML Schema Definition (XSD).

The ***KeyCachePolicyRequest*** is an "empty" request. It does not need to specify any requesting parameter or element, since the SKS server only needs to know who requested the policy. The server derives this information from the SOAP Header and consequently has everything it needs to know – the digital signature to establish the identity of the requester, as well as to to ensure message integrity in the request.

While the ***KeyCachePolicyRequest*** element is very simple, the Web Service Security (WSS) envelope – which provides security for all SKSML messages – expands the size of the message. The same request shown above, is displayed below in its entirety, with its WSS envelope. Please note that some content – such as Base64-encoded binary content - has been reformatted for aesthetics and clarity of the XML elements. The actual elements and data-types have been preserved from actual SKSML messages.

For an interpretation of the XML elements shown below, please refer to **[WSS]**.

For the sake of brevity, this specification will dispense with showing the SOAP envelope and the WSS elements in all other examples, when discussing SKSML.


```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header>
        <wsse:Security xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" SOAP-
ENV:mustUnderstand="1">
        <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
       EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary"
       ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509v3" wsu:Id="XWSSGID-1172790302111-1738806553">
       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=
        </wsse:BinarySecurityToken>
        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
           <ds:SignedInfo>
                  <ds:CanonicalizationMethod
                     Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                  <InclusiveNamespaces
                     xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"
                     PrefixList="wsse SOAP-ENV"/>
               </ds:CanonicalizationMethod>
              <ds:SignatureMethod
                     Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
              <ds:Reference URI="#XWSSGID-1172790300636-653454040">
               <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                  <ds:DigestValue>lU4m+rp4oebgl9g+t3nRaZYqUlE=</ds:DigestValue>
              </ds:Reference>
               <ds:Reference URI="#XWSSGID-1172790300637708871805">
```

```
                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                   <ds:DigestValue>WCpOmTCbffcEHXhGf5rlEYWLrZg=</ds:DigestValue>
               </ds:Reference>
           </ds:SignedInfo>
           <ds:SignatureValue>
           svStAvBRRrF+g2biPl7uWHkJTQPIl8t4phMbOZQsZlQcn36tcMSj/a4+4LPNfOB3Y8yO2lr1Oa1
           fGqCPAWZNuEH34VQEM196rRwV258mgp8uwpXEYJIgPJqg89w8+/NdaODccLQ2Bizu7QM/HSM2ab
           ogNJwqmbSyIazOsnOcU=
           </ds:SignatureValue>
           <ds:KeyInfo>
                   <wsse:SecurityTokenReference xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
                wsu:Id="XWSSGID-1172790300633-442423344">
               <wsse:Reference URI="#XWSSGID-1172790302111-1738806553"
                       ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3"/>
               </wsse:SecurityTokenReference>
           </ds:KeyInfo>
       </ds:Signature>
        <wsu:Timestamp xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="XWSSGID-11727903006377O8871805">
           <wsu:Created>2007-03-01T23:05:00Z</wsu:Created>
           <wsu:Expires>2007-03-01T23:05:05Z</wsu:Expires>
       </wsu:Timestamp>
   </wsse:Security>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd" wsu:Id="XWSSGID-1172790300636-653454040">
         <ekmi:KeyCachePolicyRequest xmlns:ekmi="http://docs.oasis-
open.org/ekmi/2008/01"/>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.12 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. (The SOAP message, as indicated earlier, is secured using WSS, but only the actual SKSML content is displayed and discussed here).

```
[m01]  <ekmi:KeyCachePolicyResponse
[m02]        xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
[m03]           xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
[m04]        <ekmi:KeyCachePolicy>
[m05]           <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
[m06]           <ekmi:PolicyName>No Caching Policy</ekmi:PolicyName>
[m07]           <ekmi:Description>
[m08]              This policy is for high-risk, always-connected machines on the
[m09]              network, which will never cache symmetric keys locally.  This
[m10]              policy never expires (but checks monthly for any updates).
[m11]           </ekmi:Description>
[m12]           <ekmi:KeyClass>NoCachingClass</ekmi:KeyClass>
[m13]           <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
[m14]           <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
[m15]           <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
[m16]           <ekmi:Status>Active</ekmi:Status>
[m17]        </ekmi:KeyCachePolicy>
[m18]  </ekmi:KeyCachePolicyResponse>
```

[m01] is the start of the ***KeyCachePolicyResponse*** element.

[m02] and **[m03]** identify the namespaces to which this XML conforms, and the location of their XML Schema Definitions (XSD).

[m04] is the start of the first – and only - ***KeyCachePolicy*** element.

[m05] 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.

[m06] provides a descriptive name for this key-cache policy through the ***PolicyName*** element, which is helpful to human readers when identifying this policy.

[m07] is the start tag of the ***Description*** element.

[m08] - [m10] provides a human-readable description about this key-cache policy.

[m11] is the closing tag of the ***Description*** element.

[m12] specifies the ***KeyClass*** to which this policy applies. Only keys that belong to this key-class are subject to this caching policy.

[m13] 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.

[m14] 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.

[m15] 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.

[m16] indicates the **Status** of this ***KeyCachePolicy*** and whether it is an active policy or not.

[m17] is the closing tag of the ***KeyCachePolicy*** element.

[m18] is the closing tag of the ***KeyCachePolicyResponse*** element.

3.13 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.

```
[m01]  <ekmi:KeyCachePolicyResponse
[m02]            xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
[m03]            xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
[m04]     <ekmi:KeyCachePolicy>
[m05]         <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
[m06]          <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
[m07]          <ekmi:Description>
[m08]             This policy defines how company-issued laptops will manage symmetric
[m09]             keys used for file/disk encryption in their local cache.  This
[m10]             policy must be used by all laptops that use the company EKMI.
[m11]         </ekmi:Description>
[m12]         <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
[m13]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
[m14]         <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
[m15]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
[m16]         <ekmi:Status>Active</ekmi:Status>
[m17]         <ekmi:NewKeysCacheDetail>
```

```
[m18]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
[m19]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
[m20]          </ekmi:NewKeysCacheDetail>
[m21]          <ekmi:UsedKeysCacheDetail>
[m22]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
[m23]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
[m24]          </ekmi:UsedKeysCacheDetail>
[m25]       </ekmi:KeyCachePolicy>
[m26]   </ekmi:KeyCachePolicyResponse>
```

[m01] is the start of the ***KeyCachePolicyResponse*** element.

[m02] and **[m03]** identify the namespaces to which this XML conforms, and the location of their XML Schema Definitions (XSD).

[m04] is the start of the first – and only - ***KeyCachePolicy*** element.

[m05] identifies the ***KeyCachePolicyID*** (KCPID) assigned by the SKS server for the key-caching policy being returned.

[m06] provides a descriptive name for this key-cache policy through the ***PolicyName*** element, which is helpful to human readers when identifying this policy.

[m07] is the start tag of the ***Description*** element.

[m08] - [m10] provides a human-readable description about this key-cache policy.

[m11] is the closing tag of the ***Description*** element.

[m12] specifies the ***KeyClass*** to which this policy applies. Only keys that belong to this key-class are subject to this caching policy.

[m13] 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.

[m14] 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.

[m15] 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.

[m16] indicates the **Status** of this ***KeyCachePolicy*** and whether it is an active policy or not.

[m17] 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.

[m18] 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.

[m19] 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.

[m20] is the closing tag of the ***NewKeysCacheDetail*** element.

[m21] 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.

[m22] 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.

[m23] 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.

[m24] is the closing tag of the ***UsedKeysCacheDetail*** element.

[m25] is the closing tag of the ***KeyCachePolicy*** element.

[m26] is the closing tag of the ***KeyCachePolicyResponse*** element.

3.14 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.

```
[m01]  <ekmi:KeyCachePolicyResponse
[m02]            xmlns:ekmi='http://docs.oasis-open.org/ekmi/2008/01'
[m03]            xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'>
[m04]      <ekmi:KeyCachePolicy>
[m05]            <ekmi:KeyCachePolicyID>10514-1</ekmi:KeyCachePolicyID>
[m06]            <ekmi:PolicyName>No Caching Policy</ekmi:PolicyName>
[m07]            <ekmi:Description>
[m08]               This policy is for high-risk, always-connected machines on the
[m09]               network, which will never cache symmetric keys locally.  This
[m10]               policy never expires (but checks monthly for any updates).
[m11]            </ekmi:Description>
[m12]            <ekmi:KeyClass>NoCachingClass</ekmi:KeyClass>
[m13]            <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
[m14]            <ekmi:EndDate>1969-01-01T00:00:00.0</ekmi:EndDate>
[m15]            <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
[m16]            <ekmi:Status>Active</ekmi:Status>
[m17]      </ekmi:KeyCachePolicy>
[m18]      <ekmi:KeyCachePolicy>
[m19]         <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
[m20]         <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
[m21]         <ekmi:Description>
[m22]            This policy defines how company-issued laptops will manage symmetric
[m23]            keys used for file/disk encryption in their local cache.  This
[m24]            policy must be used by all laptops that use the company EKMI.
[m25]         </ekmi:Description>
[m26]         <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
[m27]         <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
[m28]         <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
[m29]         <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
[m30]         <ekmi:Status>Active</ekmi:Status>
[m31]         <ekmi:NewKeysCacheDetail>
[m32]            <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
[m33]            <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
[m34]         </ekmi:NewKeysCacheDetail>
[m35]         <ekmi:UsedKeysCacheDetail>
[m36]            <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
[m37]            <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
[m38]         </ekmi:UsedKeysCacheDetail>
[m39]      </ekmi:KeyCachePolicy>
[m40]      <ekmi:KeyCachePolicy>
[m41]         <ekmi:KeyCachePolicyID>10514-17</ekmi:KeyCachePolicyID>
```

```
[m42]          <ekmi:PolicyName>Corporate Laptop Key Caching Policy</ekmi:PolicyName>
[m43]          <ekmi:Description>
[m44]             This policy defines how company-issued laptops will manage
[m45]             symmetric keys used for file/disk encryption in their local
[m46]             cache.  This policy must be used by all laptops.
[m47]          </ekmi:Description>
[m48]          <ekmi:KeyClass>LaptopKeysCachingClass</ekmi:KeyClass>
[m49]          <ekmi:StartDate>2008-01-01T00:00:01.0</ekmi:StartDate>
[m50]          <ekmi:EndDate>2008-12-31T00:00:01.0</ekmi:EndDate>
[m51]          <ekmi:PolicyCheckInterval>2592000</ekmi:PolicyCheckInterval>
[m52]          <ekmi:Status>Active</ekmi:Status>
[m53]          <ekmi:NewKeysCacheDetail>
[m54]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
[m55]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
[m56]          </ekmi:NewKeysCacheDetail>
[m57]          <ekmi:UsedKeysCacheDetail>
[m58]             <ekmi:MaximumKeys>3</ekmi:MaximumKeys>
[m59]             <ekmi:MaximumDuration>7776000</ekmi:MaximumDuration>
[m60]          </ekmi:UsedKeysCacheDetail>
[m61]      <ekmi:KeyCachePolicy>
[m62]  </ekmi:KeyCachePolicyResponse>
```

[m01] is the start of the ***KeyCachePolicyResponse*** element.

[m02] and **[m03]** identify the namespaces to which this XML conforms, and the location of their XML Schema Definitions (XSD).

[m04] is the start of the first of three ***KeyCachePolicy*** elements in this response.

[m05] 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.

[m06] provides a descriptive name for this key-cache policy through the ***PolicyName*** element, which is helpful to human readers when identifying this policy.

[m07] is the start tag of the ***Description*** element.

[m08] - [m10] provides a human-readable description about this key-cache policy.

[m11] is the closing tag of the ***Description*** element.

[m12] specifies the ***KeyClass*** to which this policy applies. Only keys that belong to this key-class are subject to this caching policy.

[m13] 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.

[m14] 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.

[m15] 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.

[m16] indicates the **Status** of this ***KeyCachePolicy*** and whether it is an active policy or not.

[m17] is the closing tag of the first ***KeyCachePolicy*** element.

[m18] is the start of the second of three ***KeyCachePolicy*** elements in this response.

[m19] identifies the ***KeyCachePolicyID*** (KCPID) assigned by the SKS server for the key-caching policy being returned.

[m20] provides the descriptive name for this key-cache policy through the ***PolicyName*** element.

[m21] is the start tag of the ***Description*** element.

[m22] - **[m24]** provides a human-readable description about this key-cache policy.

[m25] is the closing tag of the ***Description*** element.

[m26] specifies the ***KeyClass*** to which this policy applies. Only keys that belong to this key-class are subject to this caching policy.

[m27] 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.

[m28] 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.

[m29] 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.

[m30] indicates the **Status** of this ***KeyCachePolicy*** and whether it is an active policy or not.

[m31] 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.

[m32] 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.

[m33] 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.

[m34] is the closing tag of the ***NewKeysCacheDetail*** element.

[m35] 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.

[m36] 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.

[m37] 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.

[m38] is the closing tag of the ***UsedKeysCacheDetail*** element.

[m39] is the closing tag of the second ***KeyCachePolicy*** element.

[m40] is the start of the third ***KeyCachePolicy*** element in this response.

[m41] identifies the ***KeyCachePolicyID*** (KCPID) assigned by the SKS server for the key-caching policy being returned.

[m42] provides the descriptive name for this key-cache policy through the ***PolicyName*** element.

[m43] is the start tag of the ***Description*** element.

[m44] - [m46] provides a human-readable description about this key-cache policy.

[m47] is the closing tag of the ***Description*** element.

[m48] specifies the ***KeyClass*** to which this policy applies. Only keys that belong to this key-class are subject to this caching policy.

[m49] specifies the date and time that this ***KeyCachePolicy*** is effective.

[m50] specifies the date and time that this ***KeyCachePolicy*** becomes invalid.

[m51] specifies the frequency at which this client must check with the SKS server for updates to the key-caching policy.

[m52] indicates the **Status** of this ***KeyCachePolicy*** and whether it is an active policy or not.

[m53] 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.

[m54] 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.

[m55] 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.

[m56] is the closing tag of the ***NewKeysCacheDetail*** element.

[m57] 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.

[m58] 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.

[m59] 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.

[m60] is the closing tag of the ***UsedKeysCacheDetail*** element.

[m61] is the closing tag of the third and final ***KeyCachePolicy*** element in this response.

[m62] is the closing tag of the ***KeyCachePolicyResponse*** element.

Appendix A. Acknowledgments

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

Participants:

Appendix B. Revision History

Version	Date	Author	Notes
DRAFT 5	June 17, 2008	Arshad Noor	Initial version

Appendix C. Non-Normative Text