KMIP Template Examples Discussion

Bruce Rich / Tim Hudson

October 4th, 2012

Specification - Context

2 Objects

2.1 Base Objects

Attribute, Credential, Key Block, Key Value, Key Wrapping Data, Key Wrapping Specification, Transparent Key Structures, Template-Attribute Structures, Extension Information

2.2 Managed Objects

Certificate, Symmetric Key, Public Key, Private Key, Split Key, Template, Secret Data, Opaque Object

3 Attributes

Unique Identifier, Name, Object Type, Cryptographic Algorithm, Cryptographic Length, Cryptographic Parameters, Cryptographic Domain Parameters, Certificate Type, Certificate Length, X.509 Certificate Identifier, X.509 Certificate Subject, X.509 Certificate Issuer, Digital Signature Algorithm, Digest, Operation Policy Name, Cryptographic Usage Mask, Lease Time, Usage Limits, State, Initial Date, Activation Date, Process Start Date, Process Stop Date, Deactivation Date, Destroy Date, Compromise Occurrence Date, Compromise Date, Revocation Reason, Archive Date, Object Group, Fresh, Link, Application Specific Information, Contact Information, Last Change Date, Custom Attribute

Specification - Template

2.2 Managed Objects

Managed Objects are objects that are the subjects of key management operations, which are described in Sections 4 and 5. Managed Cryptographic Objects are the subset of Managed Objects that contain cryptographic material (e.g. certificates, keys, and secret data).

2.2.6 Template

A Template is a named Managed Object containing the client-settable attributes of a Managed Cryptographic Object (i.e., a stored, named list of attributes). A Template is used to specify the attributes of a new Managed Cryptographic Object in various operations. It is intended to be used to specify the cryptographic attributes of new objects in a standardized or convenient way. None of the client-settable attributes specified in a Template except the Name attribute apply to the template object itself, but instead apply to any object created using the Template.

The Template MAY be the subject of the Register, Locate, Get, Get Attributes, Get Attribute List, Add Attribute, Modify Attribute, Delete Attribute, and Destroy operations.

An attribute specified in a Template is applicable either to the Template itself or to objects created using the Template.

(cont)

Specification - Template

2.2.6 Template (cont)

Attributes applicable to the Template itself are: Unique Identifier, Object Type, Name, Initial Date, Archive Date, and Last Change Date.

Attributes applicable to objects created using the Template are:

- Cryptographic Algorithm
- Cryptographic Length
- Cryptographic Domain Parameters
- Cryptographic Parameters
- Certificate Length
- Operation Policy Name
- Cryptographic Usage Mask
- Digital Signature Algorithm
- Usage Limits
- Activation Date
- Process Start Date
- Protect Stop Date
- Deactivation Date
- Object Group
- Application Specific Information
- Contact Information
- Custom Attribute

Observations

Parsing the list of Attributes from section 3 and comparing it to the table in 2.2.6 and attributes in red are applicable to the Template and attributes in blue are applicable to the objects created using the Template and those in black are not-specified.

Unique Identifier, Name, Object Type, Cryptographic Algorithm, Cryptographic Length, Cryptographic Parameters, Cryptographic Domain Parameters, Certificate Type, Certificate Length, X.509 Certificate Identifier, X.509 Certificate Subject, X.509 Certificate Issuer, Digital Signature Algorithm, Digest, Operation Policy Name, Cryptographic Usage Mask, Lease Time, Usage Limits, State, Initial Date, Activation Date, Process Start Date, Process Stop Date, Deactivation Date, Destroy Date, Compromise Occurrence Date, Compromise Date, Revocation Reason, Archive Date, Object Group, Fresh, Link, Application Specific Information, Contact Information, Last Change Date, Custom Attribute

Why is Fresh not in the list? Bruce: Perhaps "Fresh" similar to "State"?
Why can Templates not have an Object Group? Bruce: Perhaps no use case?
Why can Templates not have a Custom Attribute? Bruce: Perhaps no use case?
Tim: Object Group and Custom Attributes are being used as managed object "collection" mechanisms currently and Templates are "different".

Why is Link not in the list?
[note: Link defines the object types for which it is valid in a restrictive manner – but conceptually why can it be neither applicable to a Template or objects created referencing a Template]

Observations

None of the client-settable attributes **specified in a Template** except the Name attribute apply to the template object itself, but instead apply to any object created using the Template.

An attribute **specified in a Template** is applicable either to the Template itself or to objects created using the Template

Tim: "specified in a Template" is clear in that it means in the Template structure itself however the second sentence is contradictory.

Bruce: The phrase "template structure" does not appear in this section because it was not the intent. A Template is a blank managed object to which attributes are attached. At the completion of the registration, a Template has some attributes, which may be added to, changed, or deleted, as per rules of attribute modification for Managed Objects. The phrase "specified in a Template" could have been rendered "via a Template reference". The second sentence explains that some of the attributes are for the Template itself (like InitialDate) and some are to be applied to other objects via Template reference (which is by the Template's Name).

Both: this wording in the specification needs to be "clarified"

Register

4.3 Register

This operation requests the server to register a **Managed Object** that was created by the client or obtained by the client through some other means, allowing the server to manage the object. The arguments in the request are similar to those in the Create operation, but also MAY contain the object itself for storage by the server. Optionally, objects that are not to be stored by the key management system MAY be omitted from the request (e.g., private keys).

Request Payload		
Object	**REQUIRED**	**Description**
Object Type, see 3.3	Yes	Determines the type of object being registered.
Template-Attribute, see 2.1.8	Yes	Specifies desired object attributes using templates and/or individual attributes.
Certificate, Symmetric Key, Private Key, Public Key, Split Key, Template Secret Data or Opaque Object, see 2.2	No	The object being registered. The object and attributes MAY be wrapped. Some objects (e.g., Private Keys), MAY be omitted from the request.

Register - Observations

4.3 Register – without the third field – what does that mean?

It hints the object itself is optional:
"MAY contain the object itself for storage by the server"
"Optionally, objects that are not to be stored by the key management system MAY be omitted from the request (e.g., private keys)"
"REQUIRED: No"
"Description: Some objects (e.g., Private Keys), MAY be omitted from the request."

Tim: this makes little sense in that allowing registration without specification of the object indicates (by the text):

1. The object is not stored by the server
2. The result of the register cannot be a managed object as there is no mechanism within section 2.2 that allows for a managed object where the definition allows for the object itself to be not required (i.e. REQUIRED: No for the first line).

If the object is not stored how can it be the subject of an operation?
Managed cryptographic objects are also explicitly required to contain cryptographic material.

Bruce: 1) The Attributes of a Cryptographic object can be Registered, without supplying the cryptographic material, which allow the server to manage its State. 2) A new Template can be composed from other Template references or inline attributes, so one can compose a new Template without passing any object. One gets back a Unique Identifier for this composition of Attributes, resolved at time of Register. See lines 1069-1076 [see next slide] of the spec for precedence rules.

Register - Observations

[Context for reference to lines in the specification on preceding slide]

4 Client-to-Server Operations [lines 1069-1076]

Requests MAY contain attribute values to be assigned to the object. This information is specified with a Template-Attribute (see Section 2.1.8) that contains zero or more template names and zero or more individual attributes. If more than one template name is specified, and there is a conflict between the single-instance attributes in the templates, then the value in the last of the conflicting templates takes precedence. If there is a conflict between the single-instance attributes in the request and the single-instance attributes in a specified template, then the attribute values in the request take precedence. For multi-instance attributes, the union of attribute values is used when the attributes are specified more than once.

Register Template – Example 1a

```
<ObjectType type="Enumeration" value="Template"/>
<TemplateAttribute>
</TemplateAttribute>
<Template>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Algorithm"/>
    <AttributeValue type="Enumeration" value="AES"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Length"/>
    <AttributeValue type="Integer" value="256"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Usage Mask"/>
    <AttributeValue type="Integer" value="Decrypt Encrypt"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Name"/>
    <AttributeValue>
      <NameValue type="TextString" value="Template1"/>
      <NameType type="Enumeration" value="UninterpretedTextString"/>
    </AttributeValue>
  </Attribute>
</Template>
```

Register Template – Example 1b

```
<ObjectType type="Enumeration" value="Template"/>
<TemplateAttribute>
 <Attribute>
    <AttributeName type="TextString" value="Name"/>
    <AttributeValue>
      <NameValue type="TextString" value="Template1"/>
      <NameType type="Enumeration" value="UninterpretedTextString"/>
    </AttributeValue>
  </Attribute>
</TemplateAttribute>
<Template>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Algorithm"/>
    <AttributeValue type="Enumeration" value="AES"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Length"/>
    <AttributeValue type="Integer" value="256"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Usage Mask"/>
    <AttributeValue type="Integer" value="Decrypt Encrypt"/>
  </Attribute>
</Template>
```

Get Template – Option 1

You get back only exactly what you placed in the Template attribute at Register

```
<Template>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Algorithm"/>
    <AttributeValue type="Enumeration" value="AES"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Length"/>
    <AttributeValue type="Integer" value="256"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Usage Mask"/>
    <AttributeValue type="Integer" value="Decrypt Encrypt"/>
  </Attribute>
</Template>
```

Get Template – Option 2

You get back what you placed in the Template attribute at Register and some other Attributes

```
<Template>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Algorithm"/>
    <AttributeValue type="Enumeration" value="AES"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Length"/>
    <AttributeValue type="Integer" value="256"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Usage Mask"/>
    <AttributeValue type="Integer" value="Decrypt Encrypt"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Object Type"/>
    <AttributeValue type="Enumeration" value="Template"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Name"/>
    <AttributeValue>
      <NameValue type="TextString" value="Template1"/>
      <NameType type="Enumeration" value="UninterpretedTextString"/>
    </AttributeValue>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Unique Identifier"/>
    <AttributeValue type="TextString" value="e176df8d-16de-4bed-8031-0ef7932f8740"/>
  </Attribute>
</Template>
```

Get Template – Option 3

You get back what you placed in the Template attribute at Register and some other different list of Attributes

```
<Template>
  <Attribute>
    <AttributeName type="TextString" value="Object Type"/>
    <AttributeValue type="Enumeration" value="Template"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Algorithm"/>
    <AttributeValue type="Enumeration" value="AES"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Length"/>
    <AttributeValue type="Integer" value="256"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Name"/>
    <AttributeIndex type="Integer" value="0"/>
    <AttributeValue>
      <NameValue type="TextString" value="Template1"/>
      <NameType type="Enumeration" value="UninterpretedTextString"/>
    </AttributeValue>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Initial Date"/>
    <AttributeValue type="DateTime" value="2012-04-24T09:54:08+00:00"/>
  </Attribute>
</Template>
```

Get Template – Option 4

You get back what you placed in the Template attribute at Register and yet another different list of Attributes

```
<Template>
  <Attribute>
    <AttributeName type="TextString" value="Object Type"/>
    <AttributeValue type="Enumeration" value="Template"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Algorithm"/>
    <AttributeValue type="Enumeration" value="AES"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Length"/>
    <AttributeValue type="Integer" value="256"/>
  </Attribute>
  . . .
  <Attribute>
    <AttributeName type="TextString" value="Name"/>
    <AttributeIndex type="Integer" value="0"/>
    <AttributeValue>
      <NameValue type="TextString" value="Template1"/>
      <NameType type="Enumeration" value="UninterpretedTextString"/>
    </AttributeValue>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Initial Date"/>
    <AttributeValue type="DateTime" value="2012-04-24T09:54:08+00:00"/>
  </Attribute>
</Template>
```

Register Example - Observations

4.3 Register – it is under-specified

There are at least three (major) interpretations of how a Register of a Template should be interpreted. These implementations are in shipping product.

These products are non-interoperable in both how Register accepts values and how Get returns values.

Should the Name be included in the Template structure during registration?
The definition of Template in 2.2.6 has
> "None of the client-settable attributes specified in a Template except the Name attribute apply to the template object itself, but instead apply to any object created using the Template."

Tim: The Name should be specified in the Template-Attribute list and not in the Template structure – and if it was done this way (as it is by some vendors) then it makes the handling clear and consistent.
Bruce: The Name can be picked out of the set of incoming Attributes and servers can tolerate different client expressions. I agree that servers should be precise in what they return, but should be tolerant in what they accept.

Can the Template structure itself be left out of the Register request?

Tim: Only if you accept that it won't be stored as a managed object ... and ignore that Template is noted as "a **stored**, named list of attributes" which tends to indicate the original (mixed) intent.
Bruce: As noted before, the spec permits amalgamation of Attributes, and I think it unwise to change this.
Tim: Amalgamation is for attributes of managed objects – not for managed objects themselves
Bruce: And a Template is a managed object, which when referenced will contribute certain Attributes to other Managed Objects. It's like a view of a database table that selects a subset of values in the table. A Template reference pulls certain Attributes from the Template.

Register Template – Example 2a

```
<ObjectType type="Enumeration" value="Template"/>
<TemplateAttribute>
 <Attribute>
    <AttributeName type="TextString" value="Name"/>
    <AttributeValue>
      <NameValue type="TextString" value="Template1"/>
      <NameType type="Enumeration" value="UninterpretedTextString"/>
    </AttributeValue>
  </Attribute>
 <Attribute>
    <AttributeName type="TextString" value="Cryptographic Algorithm"/>
    <AttributeValue type="Enumeration" value="AES"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Length"/>
    <AttributeValue type="Integer" value="256"/>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Usage Mask"/>
    <AttributeValue type="Integer" value="Decrypt Encrypt"/>
  </Attribute>
</TemplateAttribute>

NOTE: THERE IS NO <Template> ... </Template> here
```

Register Template – Example 2b

```
<ObjectType type="Enumeration" value="Template"/>
<TemplateAttribute>
 <Name>
    <NameValue type="TextString" value="ExistingAESTemplate128"/>
    <NameType type="Enumeration" value="UninterpretedTextString"/>
 </Name>
 <Attribute>
    <AttributeName type="TextString" value="Name"/>
    <AttributeValue>
      <NameValue type="TextString" value="NewAESTemplate256"/>
      <NameType type="Enumeration" value="UninterpretedTextString"/>
    </AttributeValue>
  </Attribute>
  <Attribute>
    <AttributeName type="TextString" value="Cryptographic Length"/>
    <AttributeValue type="Integer" value="256"/>
  </Attribute>
</TemplateAttribute>
```

NOTE: THERE IS NO <Template> ... </Template> here, just a reference to an existing Template, plus the Name we want for the new Template, and an override of an Attribute from the existing Template.

Register Example 2 - Observations

4.3 Register – Example 2a and Example 2b

Should it be allowed?
Some vendors allow it, some require it, some reject it. Non-interoperable!

Tim: it should not be allowed

Bruce: it should be allowed – the specification says that the third parameter is not required. This was intentional and not an oversight.

Specification - Operations

4.11 Get
This operation requests that the server returns the Managed Object specified by its Unique Identifier.
4.12 Get Attributes
This operation requests one or more attributes of a Managed Object.
4.13 Get Attribute List
This operation requests a list of the attribute names associated with a Managed Object.
4.14 Add Attribute
This request adds a new attribute instance to a Managed Object and sets its value.
4.15 Modify Attribute
This request modifies the value of an existing attribute instance associated with a Managed Object.
4.16 Delete Attribute
This request deletes an attribute associated with a Managed Object.

None of the above wording makes any distinction between the attributes of a Template as a Managed Object (“associated with”) and the attributes that a Template provides when referenced by Name in the Template-Attribute lists in Create, CreateKeyPair, Register, Certify, DeriveKey, ReKey, ReKeyKeyPair, ReCertify.

Register Example 2- Observations

General

What about AddAttribute? **Can you AddAttribute to a Template and have that change the result of a Get?** Some vendors allow it, some ignore it, some reject it. Non-interoperable!
Bruce: A Template must obey the contract for ManagedObjects, which says you can permute Attributes and Attribute values.

What about GetAttributeList? **What should GetAttributeList return?** Some vendors return a combined list of attributes, some just those that apply to the Template. Non-interoperable.

Tim: nothing in the specification text of the operations which handle attributes supports the concept of modifying the managed object – it is all about attributes associated with the managed object. To support this interpretation you have to take a leap of faith beyond the specification text.

Bruce: there is a list in 2.2.6 which makes it clear which attributes apply to the template and which apply to objects created referencing the template so an implementation can handle sorting out allowing those which apply to objects created from a template and allow those to be handled in the attribute operations

Tim: if that is the case then do Get and GetAttributeList return different values or the same values?

Bruce: Good question. GetAttributeList is clear that the names of all the attributes are returned (and GetAttributes with no names specified means return all attributes). The vague part of the specification is what Get of a Template should return. Since the spec did not clearly eliminate any attributes from consideration, our implementation returns them all.

Summary

Tim: Templates should be handled just like all the other Managed Objects and the text in 2.2.6 should be updated to remove the contradictory statements and Register should require specification of the managed object

Bruce: Templates are different for good reason and the Get operation needs to be updated to note which set of attributes should be returned (all, template-only, template-contributing)

We have an interoperability issue that needs to be resolved.
The specification needs to be updated to make the areas of ambiguity clear.

It is likely that vendors will alter existing behaviour of products under KMIP 1.0 and KMIP 1.1 to be consistent with the outcome of whatever goes into KMIP 1.2 so it is important to resolve this issue with that context in mind.

Tim: The specification should not be open to such wide interpretation – interoperability should be the guide for decisions on approaches

Bruce: When amending an existing specification, we should be judicious in making incompatible changes. Where possible, implementations (and hence the specification) should be flexible in what is accepted and strict in what is returned. So clarifying what the server does is good, but restricting the client is not so good.