

Certificate Management over CMS (CMC) Updates

Abstract

This document contains a set of updates to the base syntax for CMC, a Certificate Management protocol using the Cryptographic Message Syntax (CMS). This document updates [RFC 5272](#), [RFC 5273](#), and [RFC 5274](#).

The new items in this document are: new controls for future work in doing server side key generation, definition of a Subject Information Access value to identify CMC servers, and the registration of a port number for TCP/IP for the CMC service to run on.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in [Section 2 of RFC 5741](#).

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <http://www.rfc-editor.org/info/rfc6402>.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Terminology	3
1.2. Abbreviations	3
2. Updates to RFC 5272 - "Certificate Management over CMS (CMC)"	3
2.1. New Section 1.3 - "Updates Made by RFC 6402"	3
2.2. Update Section 6 - "Controls"	4
2.3. Replace Section 6.3 - "Linking Identity and POP Information"	4
2.4. Replace Section 6.3.3 - "Renewal and Rekey Messages"	5
2.5. New Section 6.20 - "RA Identity Proof Witness Control"	5
2.6. New Section 6.21 - "Response Body Control"	7
2.7. New Section 7 - "Other Attributes"	8
2.8. New Section 7.1 - "Change Subject Name Attribute"	8
2.9. New Section 9 - "Certificate Requirements"	10
2.10. New Section 9.1 - "Extended Key Usage"	10
2.11. New Section 9.2 - "Subject Information Access"	11
2.12. Update Section 8 - "Security Considerations"	11
3. Updates to RFC 5273 - "Certificate Management over CMS (CMC): Transport Protocols"	12
3.1. Update Section 5 - "TCP-Based Protocol"	12
3.2. New Section 6 - "IANA Considerations"	12
4. Updates to RFC 5274 - "Certificate Management Message over CMS (CMC): Compliance Requirements"	13
4.1. Update to Section 4.2 - "Controls"	13
5. IANA Considerations	13
6. Security Considerations	13
7. References	13
7.1. Normative References	13
7.2. Informative References	14
Appendix A. ASN.1 Modules	15
A.1. 1988 ASN.1 Module	15
A.2. 2008 ASN.1 Module	24

1. Introduction

While dealing with the Suite B profile of CMC [RFC6403], a number of deficiencies were noted in the current base CMC specification. This document has a set of updates to [RFC5272], [RFC5273], and [RFC5274] to deal with those issues.

1.1. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2. Abbreviations

The following abbreviations are used in this document. Terms are used as defined in [Section 2.1 of RFC 5272](#).

CA - Certification Authority
CRL - Certificate Revocation List
CRMF - Certificate Request Message Format
EE - End-Entity
MAC - Message Authentication Code
PKI - Public Key Infrastructure
RA - Registration Authority

2. Updates to [RFC 5272](#) - "Certificate Management over CMS (CMC)"

2.1. New [Section 1.3](#) - "Updates Made by [RFC 6402](#)"

Insert this section before the current [Section 1.3](#).

The following updates were made by [RFC 6402](#).

- o Add new controls:

RA Identity Witness allows for an RA to perform identity checking using the identity and shared-secret, and then tell any following servers that the identity check was successfully performed.

Response Body allows for an RA to identify a nested response for an EE to process.

- o Create a new attribute, Change Subject Name, that allows a client to request a change in the subject name and subject alternate name fields in a certificate.

- o Add Extended Key Usages for CMC to distinguish server types.
- o Define a new Subject Information Access type to hold locations to contact the CMC server.
- o Clarify that the use of a pre-existing certificate is not limited to just renewal and rekey messages and is required for support. This formalizes a requirement for the ability to do renewal and rekey that previously was implicit.

2.2. Update [Section 6](#) - "Controls"

Update Table 1 by adding the following rows:

Identifier	Description	OID	ASN.1 Structure	Section
id-cmc-raIdentityWitness		id-cmc 35	BodyPartPath	6.20
id-cmc-responseBody		id-cmc 37	BodyPartPath	6.21

Addition to Table 1: CMC Control Attributes

2.3. Replace [Section 6.3](#) - "Linking Identity and POP Information"

Replace the text of the section with the following text.

In a CMC Full PKI Request, identity proof information about the client is carried in the certificate associated with the signature of the SignedData containing the certification requests, one of the two identity proof controls or the MAC computed for the AuthenticatedData containing the certification requests. Proof-of-possession (POP) information for key pairs, however, is carried separately for each PKCS #10 or CRMF certification request. (For keys capable of generating a digital signature, the POP is provided by the signature on the PKCS #10 or CRMF request. For encryption-only keys, the controls described in [Section 6.7](#) are used.) In order to prevent substitution-style attacks, the protocol must guarantee that the same entity supplied both the POP and proof-of-identity information.

We describe three mechanisms for linking identity and POP information: witness values cryptographically derived from a shared-secret ([Section 6.3.1](#)), shared-secret/subject name matching ([Section 6.3.2](#)), and subject name matching to an existing certificate ([Section 6.3.3](#)). Clients and servers MUST support the witness value and the certificate linking techniques. Clients and servers MAY support shared-secret/name matching or MAY support other bilateral techniques

of similar strength. The idea behind the first two mechanisms is to force the client to sign some data into each certification request that can be directly associated with the shared-secret; this will defeat attempts to include certification requests from different entities in a single Full PKI Request.

2.4. Replace [Section 6.3.3](#) - "Renewal and Rekey Messages"

Make the new section title "Existing Certificate Linking". Replace all text in this section with this text.

Linking between the POP and an identity is easy when an existing certificate is used. The client copies all of the naming information from the existing certificate (subject name and subject alternative name) into the new certification request. The POP on the new public key is then performed by using the new key to sign the identity information (linking the POP to a specific identity). The identity information is then tied to the POP information by signing the entire enrollment request with the private key of the existing certificate.

Existing certificate linking can be used in the following circumstances:

When replacing a certificate by doing a renewal or rekey certification request.

Using an existing certificate to get a new certificate. An example of this would be to get a key establishment certificate after having gotten a signature certificate.

Using a third-party certificate to get a new certificate from a CA. An example of this would be using a certificate and key pair distributed with a device to prove an identity. This requires that the CA have an out-of-band channel to map the identity in the device certificate to the new EE identity.

2.5. New [Section 6.20](#) - "RA Identity Proof Witness Control"

Insert this section.

The RA Identity Proof Witness control allows an RA to indicate to subsequent control processors that all of the identity proof requirements have been met. This permits the identity proof to be performed at a location closer to the end-entity. For example, the identity proof could be done at multiple physical locations, while the CA could operate on a company-wide basis. The RA performs the identity proof, and potentially other tasks that require the secret

to be used, while the CA is prevented from knowing the secret. If the identity proof fails, then the RA returns an error to the client denoting that fact.

The relevant ASN.1 for the RA Identity Proof Witness control is as follows:

```
cmc-raIdentityWitness CMC-CONTROL ::=
  { BodyPartPath IDENTIFIED BY id-cmc-raIdentityWitness }

id-cmc-raIdentityWitness OBJECT IDENTIFIER ::= {id-cmc 35}
```

The above ASN.1 defines the following items:

`cmc-raIdentityWitness` is a CMC-CONTROL associating the object identifier `id-cmc-raIdentityWitness` and the type `BodyPartPath`. This object is omitted from the 1988 module. The object is added to the object set `Cmc-Control-Set`. The control is permitted to appear only in the control sequence of a `PKIData` object. It MUST NOT appear in the control sequence of a `PKIResponse`. The control is permitted to be used only by an RA. The control may appear multiple times in a control sequence with each occurrence pointing to a different object.

`id-cmc-raIdentityWitness` is the object identifier used to identify this CMC control.

`BodyPartPath` is the type structure associated with the control. The syntax of `BodyPartPath` is defined in [Section 3.2.2](#). The path contains a sequence of body part identifiers leading to one of the following items:

`Identity Proof control` if the RA verified the identity proof in this control.

`Identity Proof Version 2 control` if the RA verified the identity proof in this control.

`Full PKI Request` if the RA performed an out-of-band identity proof for this request. The request SHOULD NOT contain either `Identity Proof control`.

`Simple PKI Request` if the RA performed an out-of-band identity proof for this request.

The RA Identity Proof Witness control will frequently be associated with a `Modify Certification Request control`, which changes the name fields in the associated certification requests. This is because the

RA knows the actual name to be assigned to the entity requesting the certificate, and the end-entity does not yet have the details of the name. (The association would be set up by the operator at the time the shared-secret was generated by the RA.)

When this control is placed in a message, it is RECOMMENDED that the Control Processed control be placed in the body sequence as well. Using the explicit new control, rather than implicitly relying on the Control Processed control is important due to the need to know explicitly which identity proofs have been performed. The new control also allows an RA to state that out-of-band identity proofs have been performed.

When the identity proof is performed by an RA, the RA also MUST validate the linking between the identity proof and the name information wrapped inside of the key proof-of-possession.

2.6. New Section 6.21 - "Response Body Control"

Insert this section.

The Response Body Control is designed to enable an RA to inform an EE that there is an embedded response message that MUST be processed as part of the processing of this message. This control is designed to be used in a couple of different cases where an RA has done some additional processing for the certification request, e.g., as key generation. When an RA performs key generation on behalf of an EE, the RA MUST respond with both the original response message from the certificate issuer (containing the certificate issuance) as part of the response generated by the RA (containing the new key). Another case where this is useful is when the secret is shared between the RA and the EE (rather than between the CA and the EE) and the RA returns the Publish Trust Anchors control (to populate the correct trust points).

The relevant ASN.1 for the Response Body Control is as follows:

```
cmc-responseBody CMC-CONTROL ::= {  
    BodyPartPath IDENTIFIED BY id-cmc-responseBody  
}  
  
id-cmc-responseBody OBJECT IDENTIFIER ::= {id-cmc 37}
```

The above ASN.1 defines the following items:

`cmc-responseBody` is a CMC-CONTROL associating the object identifier `id-cmc-responseBody` with the type `BodyPartPath`. This object is omitted from the 1988 module. The object is added to the object set `Cmc-Control-Set`. The control is permitted to appear only in the control sequence of a `PKIResponse`. The control MUST NOT appear in the control sequence of a `PKIData`. It is expected that only an intermediary RA will use this control; a CA generally does not need the control as it is creating the original innermost message.

`id-cmc-responseBody` is the object identifier used to identify this CMC control.

`BodyPartPath` is the type structure associated with the control. The syntax of `BodyPartPath` is defined in [Section 3.2.2](#). The path contains a sequence of body part identifiers leading to a `cmsSequence` item which contains a `PKIResponse` within it.

2.7. New [Section 7](#) - "Other Attributes"

Insert this section before the current [Section 7](#).

There are a number of different locations where various types of attributes can be placed in either a CMC request or a CMC response message. These places include the attribute sequence of a PKCS #10 request, controls in CRMF ([Section 6 of \[RFC4211\]](#)), and the various CMS attribute sequences.

2.8. New [Section 7.1](#) - "Change Subject Name Attribute"

Insert this section.

The Client Name Change Request attribute is designed for a client to ask for a change in its name as part of a certification request. Because of security issues, this cannot be done in the simple way of just changing the requested subject name in the certificate template. The name in the certification request MUST match the name in the certificate used to verify the request, in order that identity and possession proofs are correctly applied.

The relevant ASN.1 for the Client Name Change Request attribute is as follows:

```
at-cmc-changeSubjectName ATTRIBUTE ::=
    { ChangeSubjectName IDENTIFIED BY id-cmc-changeSubjectName }

id-cmc-changeSubjectName OBJECT IDENTIFIER ::= {id-cmc 36}

ChangeSubjectName ::= SEQUENCE {
    subject          Name OPTIONAL,
    subjectAlt       SubjectAltName OPTIONAL
}
(WITH COMPONENTS {..., subject PRESENT} |
 COMPONENTS {..., subjectAlt PRESENT} )
```

The attribute is designed to be used as an ATTRIBUTE object. As such, the attribute is placed in one of the following two places:

The attributes field in a CertificationRequest.

The controls field of a CertRequest for a CRMF certification request.

The control is identified by the Object Identifier id-cmc-changeSubjectName.

The ASN.1 type associated with control is ChangeSubjectName. The fields of the structure are configured as follows:

subject contains the requested subject name for the new certificate.

subjectAlt contains the requested subject alternative name for the new certificate.

At least one of the fields in the sequence MUST be present when encoding the structure.

When the CA processes this attribute in a certification request, it will do the following:

1. If present, the subject field is copied to the name field of the template. If the subject field is absent, the name field of the template will be set to a empty sequence.
2. If present, the subjectAlt field is used as the content of a SubjectAltName extension in the certificate. If the subjectAlt field is absent, the subjectAltName extension is removed from the certificate template.

2.9. New Section 9 - "Certificate Requirements"

Insert this section before the current Section 8.

Certificates for servers used in the CMC protocol SHOULD conform to the profile defined in [RFC5280]. This document defines some additional items that MAY appear in CMC server certificates. Section 9.1 defines some additional values for the Extended Key Usage extension. Section 9.2 defines a new Subject Information Access value that allows for a CMC certificate to publish information on how to contact the services it provides.

2.10. New Section 9.1 - "Extended Key Usage"

Insert this section.

The Extended Key Usage (EKU) extension is used to restrict the use of a certificate to specific applications. We define three different EKUs in this document. The ASN.1 to define these EKUs is:

```
id-kp-cmcCA OBJECT IDENTIFIER ::= { id-kp 27 }
id-kp-cmcRA OBJECT IDENTIFIER ::= { id-kp 28 }
id-kp-cmcArchive OBJECT IDENTIFIER ::= { id-kp 29 }
```

The usage description for each of the EKUs is as follows:

CMC Certification Authorities are identified by the id-kp-cmcCA extended key usage. The certificate may be the same as or different than the CA certificate. If a different certificate is used, the certificates containing the id-kp-cmcCA extended key usage SHOULD have the same name as the certificate used for issuing the certificates. (Using a separate key pair for CMC protocol operations and for issuing certificates and CRLs decreases the number of operations for which the private key used to sign certificates and CRLs would be used.)

CMC Registration Authorities are identified by the id-kp-cmcRA extended key usage. This usage is placed into RA certificates.

CMC Archive Servers are identified by the id-kp-cmcArchive extended key usage. CMC Archive Servers and the associated protocol are to be defined in a future document.

2.11. New [Section 9.2](#) - "Subject Information Access"

Insert this section.

The subject information access extension indicates how to access information and services for the subject of the certificate. We define a new value for use in this extension, to identify the different locations that CMC services will be available. If this value is placed in a certificate, an appropriate extended key usage defined in [Section 9.1](#) MUST be included in the certificate as well.

The id-ad-cmc OID is used when the subject offers certification services using the CMC protocol. If the CMC services are available via HTTP or FTP, accessLocation MUST be a uniformResourceIdentifier. If the CMC services are available via electronic mail, accessLocation MUST be an rfc822Name. If CMC services are available using TCP/IP, the dNSName or iPAddress name forms MUST be used. Since the GeneralName data structure does not permit the inclusion of a port number, in the absence of other external configuration information, the value of 5318 should be used. (The port registration is in [Section 3.2](#).) The semantics of other name forms of accessLocation (when accessMethod is id-ad-cmc) are not defined by this specification.

The ASN.1 type for this extension is GeneralName (see [Section 4.2.1.8 of \[RFC5280\]](#)).

id-ad-cmc OBJECT IDENTIFIER ::= { id-ad 12 }

2.12. Update [Section 8](#) - "Security Considerations"

Add the following paragraphs to the end of [Section 8](#).

A number of controls such as the RA Identity Proof Witness control exist for an RA to either make assertions about or modify a certification request. Any upstream request processor, such as a CA, MUST verify that the RA is fully identified and authorized to make the assertion or modification it is claiming. If it is not identified or authorized, then any request MUST be rejected.

CMC servers, both RAs and CAs, need to perform due diligence in checking the contents of a certification request. At an absolute minimum, all fields should be checked to ensure that the policies of the CA/RA are correctly enforced. While all fields need to be checked, special care should be taken with names, name forms, algorithm choices, and algorithm parameters.

3. Updates to RFC 5273 - "Certificate Management over CMS (CMC): Transport Protocols"

3.1. Update Section 5 - "TCP-Based Protocol"

Replace paragraph 3 in Section 5 with the following.

CMC requires a registered port number to send and receive CMC messages over TCP. The title of this IP Protocol number is "pkix-cmc". The value of this TCP port is 5318.

Prior to this update, CMC did not have a registered port number and used an externally configured port from the Private Port range. Client implementations MAY want to continue to allow for this to occur. Servers SHOULD change to use the new port. It is expected that HTTP will continue to be the primary transport method used by CMC installations.

3.2. New Section 6 - "IANA Considerations"

Insert this new section before the current Section 6.

IANA has assigned a TCP port number in the Registered Port Number range for the use of CMC.

Service name: pkix-cmc
Port Number: 5318
Transport protocol: TCP
Description: PKIX Certificate Management using CMS (CMC)
Reference: RFC 6402
Assignee: iesg@ietf.org
Contact: chair@ietf.org

4. Updates to RFC 5274 - "Certificate Management Message over CMS (CMC): Compliance Requirements"

4.1. Update to Section 4.2 - "Controls"

Add the following lines to the end of Table 1.

The following table lists the name and level of support required for each control.

Control	EE	RA	CA
RA Identity Proof Witness	N/A	MUST	(2)
Response Body	(6)	(6)	N/A

Addition to Table 1: CMC Control Attributes

The following note should be added.

6. EE's SHOULD implement if designed to work with RAs and MUST implement if intended to be used in environments where RAs are used for identity validation or key generation. RAs SHOULD implement and validate responses for consistency.

5. IANA Considerations

This document contains a new IANA Considerations section to be added to [RFC5273] as part of this update.

6. Security Considerations

No changes are made to the existing security considerations of RFC 5273 and RFC 5274. The security considerations for RFC 5272 have been slightly modified (Section 2.12).

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, June 2008.

- [RFC5273] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC): Transport Protocols", [RFC 5273](#), June 2008.
- [RFC5274] Schaad, J. and M. Myers, "Certificate Management Messages over CMS (CMC): Compliance Requirements", [RFC 5274](#), June 2008.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", [RFC 5280](#), May 2008.

7.2. Informative References

- [CMS] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, [RFC 5652](#), September 2009.
- [RFC6403] Ziegler, L., Turner, S., and M. Peck, "Suite B Profile of Certificate Management over CMS", [RFC 6403](#), November 2011.
- [RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", [RFC 4211](#), September 2005.
- [RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", [RFC 5912](#), June 2010.

Appendix A. ASN.1 Modules

A.1. 1988 ASN.1 Module

This section contains the updated ASN.1 module for [RFC5272]. This module replaces the module in Appendix A of that document. Although a 2008 ASN.1 module is provided, this remains the normative module as per the policy of the PKIX working group.

EnrollmentMessageSyntax-2011-v88

```
{ iso(1) identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-enrollMsgSyntax-2011-88(76) }
```

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- EXPORTS All --

-- The types and values defined in this module are exported for use
-- in the other ASN.1 modules. Other applications may use them for
-- their own purposes.

IMPORTS

-- PKIX Part 1 - Implicit From [RFC5280]

GeneralName, CRLReason, ReasonFlags, GeneralNames

FROM PKIX1Implicit88 {iso(1) identified-organization(3) dod(6)
 internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
 id-pkix1-implicit(19)}

-- PKIX Part 1 - Explicit From [RFC5280]

AlgorithmIdentifier, Extension, Name, CertificateSerialNumber,
id-ad, id-kp

FROM PKIX1Explicit88 {iso(1) identified-organization(3) dod(6)
 internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
 id-pkix1-explicit(18)}

-- Cryptographic Message Syntax FROM [CMS]

ContentInfo, Attribute, IssuerAndSerialNumber

FROM CryptographicMessageSyntax2004 { iso(1) member-body(2)
 us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
 modules(0) cms-2004(24)}

-- CRMF FROM [RFC4211]

CertReqMsg, PKIPublicationInfo, CertTemplate

FROM PKIXCRMF-2005 {iso(1) identified-organization(3) dod(6)
 internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
 id-mod-crmf2005(36)};

```
-- Global Types
-- UTF8String ::= [UNIVERSAL 12] IMPLICIT OCTET STRING
-- The content of this type conforms to RFC 3629.

id-pkix OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
  dod(6) internet(1) security(5) mechanisms(5) pkix(7) }

id-cmc OBJECT IDENTIFIER ::= {id-pkix 7}    -- CMC controls
id-cct OBJECT IDENTIFIER ::= {id-pkix 12}   -- CMC content types

-- The following controls have the type OCTET STRING

id-cmc-identityProof OBJECT IDENTIFIER ::= {id-cmc 3}
id-cmc-dataReturn OBJECT IDENTIFIER ::= {id-cmc 4}
id-cmc-regInfo OBJECT IDENTIFIER ::= {id-cmc 18}
id-cmc-responseInfo OBJECT IDENTIFIER ::= {id-cmc 19}
id-cmc-queryPending OBJECT IDENTIFIER ::= {id-cmc 21}
id-cmc-popLinkRandom OBJECT IDENTIFIER ::= {id-cmc 22}
id-cmc-popLinkWitness OBJECT IDENTIFIER ::= {id-cmc 23}

-- The following controls have the type UTF8String

id-cmc-identification OBJECT IDENTIFIER ::= {id-cmc 2}

-- The following controls have the type INTEGER

id-cmc-transactionId OBJECT IDENTIFIER ::= {id-cmc 5}

-- The following controls have the type OCTET STRING

id-cmc-senderNonce OBJECT IDENTIFIER ::= {id-cmc 6}
id-cmc-recipientNonce OBJECT IDENTIFIER ::= {id-cmc 7}

-- This is the content type used for a request message
--   in the protocol

id-cct-PKIData OBJECT IDENTIFIER ::= { id-cct 2 }

PKIData ::= SEQUENCE {
  controlSequence    SEQUENCE SIZE(0..MAX) OF TaggedAttribute,
  reqSequence        SEQUENCE SIZE(0..MAX) OF TaggedRequest,
  cmsSequence        SEQUENCE SIZE(0..MAX) OF TaggedContentInfo,
  otherMsgSequence   SEQUENCE SIZE(0..MAX) OF OtherMsg
}

bodyIdMax INTEGER ::= 4294967295

BodyPartID ::= INTEGER(0..bodyIdMax)
```



```
TaggedAttribute ::= SEQUENCE {
    bodyPartID      BodyPartID,
    attrType        OBJECT IDENTIFIER,
    attrValues      SET OF AttributeValue
}

AttributeValue ::= ANY

TaggedRequest ::= CHOICE {
    tcr              [0] TaggedCertificationRequest,
    crm              [1] CertReqMsg,
    orm              [2] SEQUENCE {
        bodyPartID      BodyPartID,
        requestMessageType OBJECT IDENTIFIER,
        requestMessageValue ANY DEFINED BY requestMessageType
    }
}

TaggedCertificationRequest ::= SEQUENCE {
    bodyPartID      BodyPartID,
    certificationRequest CertificationRequest
}

CertificationRequest ::= SEQUENCE {
    certificationRequestInfo SEQUENCE {
        version          INTEGER,
        subject           Name,
        subjectPublicKeyInfo SEQUENCE {
            algorithm      AlgorithmIdentifier,
            subjectPublicKey BIT STRING },
        attributes        [0] IMPLICIT SET OF Attribute },
    signatureAlgorithm     AlgorithmIdentifier,
    signature              BIT STRING
}

TaggedContentInfo ::= SEQUENCE {
    bodyPartID      BodyPartID,
    contentInfo      ContentInfo
}

OtherMsg ::= SEQUENCE {
    bodyPartID      BodyPartID,
    otherMsgType     OBJECT IDENTIFIER,
    otherMsgValue     ANY DEFINED BY otherMsgType }

-- This defines the response message in the protocol
id-cct-PKIResponse OBJECT IDENTIFIER ::= { id-cct 3 }
```

```
ResponseBody ::= PKIResponse
```

```
PKIResponse ::= SEQUENCE {  
    controlSequence SEQUENCE SIZE(0..MAX) OF TaggedAttribute,  
    cmsSequence     SEQUENCE SIZE(0..MAX) OF TaggedContentInfo,  
    otherMsgSequence SEQUENCE SIZE(0..MAX) OF OtherMsg  
}
```

```
-- Used to return status state in a response
```

```
id-cmc-statusInfo OBJECT IDENTIFIER ::= {id-cmc 1}
```

```
CMCStatusInfo ::= SEQUENCE {  
    cMCStatus CMCStatus,  
    bodyList SEQUENCE SIZE (1..MAX) OF BodyPartID,  
    statusString UTF8String OPTIONAL,  
    otherInfo CHOICE {  
        failInfo CMCFailInfo,  
        pendInfo PendInfo } OPTIONAL  
}
```

```
PendInfo ::= SEQUENCE {  
    pendToken OCTET STRING,  
    pendTime GeneralizedTime  
}
```

```
CMCStatus ::= INTEGER {  
    success (0),  
    failed (2),  
    pending (3),  
    noSupport (4),  
    confirmRequired (5),  
    popRequired (6),  
    partial (7)  
}
```

```
-- Note:
-- The spelling of unsupportedExt is corrected in this version.
-- In RFC 2797, it was unsupportedExt.
```

```
CMCFailInfo ::= INTEGER {
    badAlg          (0),
    badMessageCheck (1),
    badRequest      (2),
    badTime         (3),
    badCertId       (4),
    unsupportedExt   (5),
    mustArchiveKeys (6),
    badIdentity     (7),
    popRequired     (8),
    popFailed       (9),
    noKeyReuse      (10),
    internalCAError (11),
    tryLater        (12),
    authDataFail    (13)
}
```

```
-- Used for RAs to add extensions to certification requests
id-cmc-addExtensions OBJECT IDENTIFIER ::= {id-cmc 8}
```

```
AddExtensions ::= SEQUENCE {
    pkiDataReference    BodyPartID,
    certReferences      SEQUENCE OF BodyPartID,
    extensions          SEQUENCE OF Extension
}
```

```
id-cmc-encryptedPOP OBJECT IDENTIFIER ::= {id-cmc 9}
id-cmc-decryptedPOP OBJECT IDENTIFIER ::= {id-cmc 10}
```

```
EncryptedPOP ::= SEQUENCE {
    request      TaggedRequest,
    cms          ContentInfo,
    thePOPAlgID  AlgorithmIdentifier,
    witnessAlgID AlgorithmIdentifier,
    witness      OCTET STRING
}
```

```
DecryptedPOP ::= SEQUENCE {
    bodyPartID    BodyPartID,
    thePOPAlgID   AlgorithmIdentifier,
    thePOP        OCTET STRING
}
```

```
id-cmc-lraPOPWitness OBJECT IDENTIFIER ::= {id-cmc 11}

LraPopWitness ::= SEQUENCE {
    pkiDataBodyid    BodyPartID,
    bodyIds           SEQUENCE OF BodyPartID
}

--
id-cmc-getCert OBJECT IDENTIFIER ::= {id-cmc 15}

GetCert ::= SEQUENCE {
    issuerName        GeneralName,
    serialNumber      INTEGER }

id-cmc-getCRL OBJECT IDENTIFIER ::= {id-cmc 16}

GetCRL ::= SEQUENCE {
    issuerName        Name,
    cRLName           GeneralName OPTIONAL,
    time              GeneralizedTime OPTIONAL,
    reasons           ReasonFlags OPTIONAL }

id-cmc-revokeRequest OBJECT IDENTIFIER ::= {id-cmc 17}

RevokeRequest ::= SEQUENCE {
    issuerName        Name,
    serialNumber      INTEGER,
    reason            CRLReason,
    invalidityDate    GeneralizedTime OPTIONAL,
    passphrase        OCTET STRING OPTIONAL,
    comment           UTF8String OPTIONAL }

id-cmc-confirmCertAcceptance OBJECT IDENTIFIER ::= {id-cmc 24}

CMCCertId ::= IssuerAndSerialNumber

-- The following is used to request V3 extensions be added to a
-- certificate

id-ExtensionReq OBJECT IDENTIFIER ::= {iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) 14}

ExtensionReq ::= SEQUENCE SIZE (1..MAX) OF Extension

-- The following exists to allow Diffie-Hellman Certification
-- Request Messages to be well-formed

id-alg-noSignature OBJECT IDENTIFIER ::= {id-pkix id-alg(6) 2}
```

```
NoSignatureValue ::= OCTET STRING

-- Unauthenticated attribute to carry removable data.
-- This could be used in an update of "CMC Extensions: Server
-- Side Key Generation and Key Escrow" (February 2005) and in
-- other documents.

id-aa OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
    rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2)}
id-aa-cmc-unsignedData OBJECT IDENTIFIER ::= {id-aa 34}

CMCUnsignedData ::= SEQUENCE {
    bodyPartPath      BodyPartPath,
    identifier         OBJECT IDENTIFIER,
    content            ANY DEFINED BY identifier
}

-- Replaces CMC Status Info
--

id-cmc-statusInfoV2 OBJECT IDENTIFIER ::= {id-cmc 25}

CMCStatusInfoV2 ::= SEQUENCE {
    cMCStatus          CMCStatus,
    bodyList           SEQUENCE SIZE (1..MAX) OF
                        BodyPartReference,
    statusString       UTF8String OPTIONAL,
    otherInfo          CHOICE {
        failInfo       CMCFailInfo,
        pendInfo       PendInfo,
        extendedFailInfo SEQUENCE {
            failInfoOID OBJECT IDENTIFIER,
            failInfoValue AttributeValue
        }
    } OPTIONAL
}

BodyPartReference ::= CHOICE {
    bodyPartID      BodyPartID,
    bodyPartPath    BodyPartPath
}

BodyPartPath ::= SEQUENCE SIZE (1..MAX) OF BodyPartID

-- Allow for distribution of trust anchors
--

id-cmc-trustedAnchors OBJECT IDENTIFIER ::= {id-cmc 26}
```

```
PublishTrustAnchors ::= SEQUENCE {
    seqNumber      INTEGER,
    hashAlgorithm  AlgorithmIdentifier,
    anchorHashes   SEQUENCE OF OCTET STRING
}

id-cmc-authData OBJECT IDENTIFIER ::= {id-cmc 27}

AuthPublish ::= BodyPartID

-- These two items use BodyPartList
id-cmc-batchRequests OBJECT IDENTIFIER ::= {id-cmc 28}
id-cmc-batchResponses OBJECT IDENTIFIER ::= {id-cmc 29}

BodyPartList ::= SEQUENCE SIZE (1..MAX) OF BodyPartID

--
id-cmc-publishCert OBJECT IDENTIFIER ::= {id-cmc 30}

CMCPublicationInfo ::= SEQUENCE {
    hashAlg          AlgorithmIdentifier,
    certHashes       SEQUENCE OF OCTET STRING,
    pubInfo           PKIPublicationInfo
}

id-cmc-modCertTemplate OBJECT IDENTIFIER ::= {id-cmc 31}

ModCertTemplate ::= SEQUENCE {
    pkiDataReference  BodyPartPath,
    certReferences    BodyPartList,
    replace           BOOLEAN DEFAULT TRUE,
    certTemplate      CertTemplate
}

-- Inform follow-on servers that one or more controls have already
-- been processed

id-cmc-controlProcessed OBJECT IDENTIFIER ::= {id-cmc 32}

ControlsProcessed ::= SEQUENCE {
    bodyList          SEQUENCE SIZE(1..MAX) OF BodyPartReference
}

-- Identity Proof control w/ algorithm agility

id-cmc-identityProofV2 OBJECT IDENTIFIER ::= { id-cmc 34 }
```

```
IdentifyProofV2 ::= SEQUENCE {
    proofAlgID      AlgorithmIdentifier,
    macAlgId        AlgorithmIdentifier,
    witness         OCTET STRING
}

id-cmc-popLinkWitnessV2 OBJECT IDENTIFIER ::= { id-cmc 33 }
PopLinkWitnessV2 ::= SEQUENCE {
    keyGenAlgorithm AlgorithmIdentifier,
    macAlgorithm     AlgorithmIdentifier,
    witness          OCTET STRING
}

--

id-cmc-raIdentityWitness OBJECT IDENTIFIER ::= {id-cmc 35}

--
-- Allow for an End-Entity to request a change in name.
-- This item is added to RegControlSet in CRMF.
--

id-cmc-changeSubjectName OBJECT IDENTIFIER ::= {id-cmc 36}

ChangeSubjectName ::= SEQUENCE {
    subject          Name OPTIONAL,
    subjectAlt       GeneralNames OPTIONAL
}
-- (WITH COMPONENTS {..., subject PRESENT} |
-- WITH COMPONENTS {..., subjectAlt PRESENT} )

--
-- Embedded response from a third party for processing
--

id-cmc-responseBody OBJECT IDENTIFIER ::= {id-cmc 37}

--
-- Key purpose identifiers are in the Extended Key Usage extension
--

id-kp-cmcCA OBJECT IDENTIFIER ::= { id-kp 27 }
id-kp-cmcRA OBJECT IDENTIFIER ::= { id-kp 28 }
id-kp-cmcArchive OBJECT IDENTIFIER ::= { id-kp 28 }
```

```
--  
-- Subject Information Access identifier  
--  
  
id-ad-cmc OBJECT IDENTIFIER ::= { id-ad 12 }
```

END

A.2. 2008 ASN.1 Module

An updated 2008 ASN.1 module has been provided as part of this update. The module contains those changes that were done to update the current ASN.1 standards (done for [RFC5912]) as well as changes made for this document.

```
EnrollmentMessageSyntax-2011-v08  
  {iso(1) identified-organization(3) dod(6) internet(1)  
    security(5) mechanisms(5) pkix(7) id-mod(0)  
    id-mod-enrollMsgSyntax-2011-08(76)}  
DEFINITIONS IMPLICIT TAGS ::=  
BEGIN  
  EXPORTS ALL;  
  IMPORTS  
  
  AttributeSet{}, Extension{}, EXTENSION, ATTRIBUTE  
  FROM PKIX-CommonTypes-2009  
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)  
      mechanisms(5) pkix(7) id-mod(0) id-mod-pkixCommon-02(57)}  
  
  AlgorithmIdentifier{}, DIGEST-ALGORITHM, KEY-WRAP, KEY-DERIVATION,  
    MAC-ALGORITHM, SIGNATURE-ALGORITHM, PUBLIC-KEY  
  FROM AlgorithmInformation-2009  
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)  
      mechanisms(5) pkix(7) id-mod(0)  
      id-mod-algorithmInformation-02(58)}  
  
  CertificateSerialNumber, GeneralName, CRLReason, ReasonFlags,  
    CertExtensions, GeneralNames  
  FROM PKIX1Implicit-2009  
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)  
      mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-implicit-02(59)}  
  
  Name, id-pkix, PublicKeyAlgorithms, SignatureAlgorithms, id-ad, id-kp  
  FROM PKIX1Explicit-2009  
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)  
      mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-explicit-02(51)}
```



```
ContentInfo, IssuerAndSerialNumber, CONTENT-TYPE
FROM CryptographicMessageSyntax-2010
{ iso(1) member-body(2) us(840) rsadsi(113549)
  pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) }

CertReqMsg, PKIPublicationInfo, CertTemplate
FROM PKIXCRMF-2009
{iso(1) identified-organization(3) dod(6) internet(1) security(5)
 mechanisms(5) pkix(7) id-mod(0) id-mod-crmf2005-02(55)}

mda-sha1
FROM PKIXAlgs-2009
{ iso(1) identified-organization(3) dod(6)
  internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-pkix1-algorithms2008-02(56)}

kda-PBKDF2, maca-hMAC-SHA1
FROM CryptographicMessageSyntaxAlgorithms-2009
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
  smime(16) modules(0) id-mod-cmsalg-2001-02(37) }

mda-sha256
FROM PKIX1-PSS-OAEP-Algorithms-2009
{ iso(1) identified-organization(3) dod(6)
  internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-pkix1-rsa-pkalgs-02(54) } ;

-- CMS content types defined in this document

CMC-ContentTypes CONTENT-TYPE ::= { ct-PKIData | ct-PKIResponse, ... }

-- Signature Algorithms defined in this document

SignatureAlgs SIGNATURE-ALGORITHM ::= { sa-noSignature }

-- CMS Unsigned Attributes

CMC-UnsignedAtts ATTRIBUTE ::= { aa-cmc-unsignedData }

--
--

id-cmc OBJECT IDENTIFIER ::= {id-pkix 7}    -- CMC controls
id-cct OBJECT IDENTIFIER ::= {id-pkix 12}    -- CMC content types
```

-- This is the content type for a request message in the protocol

ct-PKIData CONTENT-TYPE ::=

{ TYPE PKIData IDENTIFIED BY id-cct-PKIData }

id-cct-PKIData OBJECT IDENTIFIER ::= { id-cct 2 }

PKIData ::= SEQUENCE {

controlSequence SEQUENCE SIZE(0..MAX) OF TaggedAttribute,

reqSequence SEQUENCE SIZE(0..MAX) OF TaggedRequest,

cmsSequence SEQUENCE SIZE(0..MAX) OF TaggedContentInfo,

otherMsgSequence SEQUENCE SIZE(0..MAX) OF OtherMsg

}

BodyPartID ::= INTEGER(0..4294967295)

TaggedAttribute ::= SEQUENCE {

bodyPartID BodyPartID,

attrType CMC-CONTROL.&id({Cmc-Control-Set}),

attrValues SET OF CMC-CONTROL.

&Type({Cmc-Control-Set}){@attrType})

}

Cmc-Control-Set CMC-CONTROL ::= {

cmc-identityProof | cmc-dataReturn | cmc-regInfo |

cmc-responseInfo | cmc-queryPending | cmc-popLinkRandom |

cmc-popLinkWitness | cmc-identification | cmc-transactionId |

cmc-senderNonce | cmc-recipientNonce | cmc-statusInfo |

cmc-addExtensions | cmc-encryptedPOP | cmc-decryptedPOP |

cmc-lraPOPWitness | cmc-getCert | cmc-getCRL |

cmc-revokeRequest | cmc-confirmCertAcceptance |

cmc-statusInfoV2 | cmc-trustedAnchors | cmc-authData |

cmc-batchRequests | cmc-batchResponses | cmc-publishCert |

cmc-modCertTemplate | cmc-controlProcessed |

cmc-identityProofV2 | cmc-popLinkWitnessV2, ...,

cmc-raIdentityWitness | cmc-responseBody }

OTHER-REQUEST ::= TYPE-IDENTIFIER

-- We do not define any other requests in this document.

-- Examples might be attribute certification requests.

OtherRequests OTHER-REQUEST ::= {...}

```

TaggedRequest ::= CHOICE {
    tcr          [0] TaggedCertificationRequest,
    crm          [1] CertReqMsg,
    orm          [2] SEQUENCE {
        bodyPartID          BodyPartID,
        requestMessageType   OTHER-REQUEST.&id({OtherRequests}),
        requestMessageValue  OTHER-REQUEST.&Type({OtherRequests}
                                {@.requestMessageType})
    }
}

TaggedCertificationRequest ::= SEQUENCE {
    bodyPartID          BodyPartID,
    certificationRequest CertificationRequest
}

AttributeList ATTRIBUTE ::= {at-extension-req, ...,
    at-cmc-changeSubjectName}

CertificationRequest ::= SEQUENCE {
    certificationRequestInfo SEQUENCE {
        version          INTEGER,
        subject           Name,
        subjectPublicKeyInfo SEQUENCE {
            algorithm      AlgorithmIdentifier{PUBLIC-KEY,
                                {PublicKeyAlgorithms}},
            subjectPublicKey BIT STRING
        },
        attributes        [0] IMPLICIT SET OF
                                AttributeSet{{AttributeList}}
    },
    signatureAlgorithm     AlgorithmIdentifier
                                {SIGNATURE-ALGORITHM,
                                {SignatureAlgorithms}},
    signature              BIT STRING
}

TaggedContentInfo ::= SEQUENCE {
    bodyPartID          BodyPartID,
    contentInfo          ContentInfo
}

OTHER-MSG ::= TYPE-IDENTIFIER

-- No other messages currently defined

OtherMsgSet OTHER-MSG ::= {...}

```

```
OtherMsg ::= SEQUENCE {
    bodyPartID      BodyPartID,
    otherMsgType     OTHER-MSG.&id({OtherMsgSet}),
    otherMsgValue     OTHER-MSG.&Type({OtherMsgSet}{@otherMsgType}) }

-- This defines the response message in the protocol

ct-PKIResponse CONTENT-TYPE ::=
    { TYPE PKIResponse IDENTIFIED BY id-cct-PKIResponse }
id-cct-PKIResponse OBJECT IDENTIFIER ::= { id-cct 3 }

ResponseBody ::= PKIResponse

PKIResponse ::= SEQUENCE {
    controlSequence SEQUENCE SIZE(0..MAX) OF TaggedAttribute,
    cmsSequence     SEQUENCE SIZE(0..MAX) OF TaggedContentInfo,
    otherMsgSequence SEQUENCE SIZE(0..MAX) OF OtherMsg
}

CMC-CONTROL ::= TYPE-IDENTIFIER

-- The following controls have the type OCTET STRING

cmc-identityProof CMC-CONTROL ::=
    { OCTET STRING IDENTIFIED BY id-cmc-identityProof }
id-cmc-identityProof OBJECT IDENTIFIER ::= {id-cmc 3}

cmc-dataReturn CMC-CONTROL ::=
    { OCTET STRING IDENTIFIED BY id-cmc-dataReturn }
id-cmc-dataReturn OBJECT IDENTIFIER ::= {id-cmc 4}

cmc-regInfo CMC-CONTROL ::=
    { OCTET STRING IDENTIFIED BY id-cmc-regInfo }
id-cmc-regInfo OBJECT IDENTIFIER ::= {id-cmc 18}

cmc-responseInfo CMC-CONTROL ::=
    { OCTET STRING IDENTIFIED BY id-cmc-responseInfo }
id-cmc-responseInfo OBJECT IDENTIFIER ::= {id-cmc 19}

cmc-queryPending CMC-CONTROL ::=
    { OCTET STRING IDENTIFIED BY id-cmc-queryPending }
id-cmc-queryPending OBJECT IDENTIFIER ::= {id-cmc 21}

cmc-popLinkRandom CMC-CONTROL ::=
    { OCTET STRING IDENTIFIED BY id-cmc-popLinkRandom }
id-cmc-popLinkRandom OBJECT IDENTIFIER ::= {id-cmc 22}
```

```
cmc-popLinkWitness CMC-CONTROL ::=
  { OCTET STRING IDENTIFIED BY id-cmc-popLinkWitness }
id-cmc-popLinkWitness OBJECT IDENTIFIER ::= {id-cmc 23}

-- The following controls have the type UTF8String

cmc-identification CMC-CONTROL ::=
  { UTF8String IDENTIFIED BY id-cmc-identification }
id-cmc-identification OBJECT IDENTIFIER ::= {id-cmc 2}

-- The following controls have the type INTEGER

cmc-transactionId CMC-CONTROL ::=
  { INTEGER IDENTIFIED BY id-cmc-transactionId }
id-cmc-transactionId OBJECT IDENTIFIER ::= {id-cmc 5}

-- The following controls have the type OCTET STRING

cmc-senderNonce CMC-CONTROL ::=
  { OCTET STRING IDENTIFIED BY id-cmc-senderNonce }
id-cmc-senderNonce OBJECT IDENTIFIER ::= {id-cmc 6}

cmc-recipientNonce CMC-CONTROL ::=
  { OCTET STRING IDENTIFIED BY id-cmc-recipientNonce }
id-cmc-recipientNonce OBJECT IDENTIFIER ::= {id-cmc 7}

-- Used to return status in a response

cmc-statusInfo CMC-CONTROL ::=
  { CMCStatusInfo IDENTIFIED BY id-cmc-statusInfo }
id-cmc-statusInfo OBJECT IDENTIFIER ::= {id-cmc 1}

CMCStatusInfo ::= SEQUENCE {
  cmcStatus      CMCStatus,
  bodyList       SEQUENCE SIZE (1..MAX) OF BodyPartID,
  statusString   UTF8String OPTIONAL,
  otherInfo      CHOICE {
    failInfo      CMCFailInfo,
    pendInfo      PendInfo
  } OPTIONAL
}

PendInfo ::= SEQUENCE {
  pendToken      OCTET STRING,
  pendTime       GeneralizedTime
}
```

```
CMCStatus ::= INTEGER {
    success          (0),
    failed           (2),
    pending          (3),
    noSupport        (4),
    confirmRequired  (5),
    popRequired      (6),
    partial          (7)
}

CMCFailInfo ::= INTEGER {
    badAlg           (0),
    badMessageCheck (1),
    badRequest       (2),
    badTime          (3),
    badCertId        (4),
    unsupportedExt    (5),
    mustArchiveKeys  (6),
    badIdentity      (7),
    popRequired      (8),
    popFailed        (9),
    noKeyReuse       (10),
    internalCAError  (11),
    tryLater         (12),
    authDataFail     (13)
}

-- Used for RAs to add extensions to certification requests

cmc-addExtensions CMC-CONTROL ::=
    { AddExtensions IDENTIFIED BY id-cmc-addExtensions }
id-cmc-addExtensions OBJECT IDENTIFIER ::= {id-cmc 8}

AddExtensions ::= SEQUENCE {
    pkiDataReference    BodyPartID,
    certReferences      SEQUENCE OF BodyPartID,
    extensions          SEQUENCE OF Extension{{CertExtensions}}
}

cmc-encryptedPOP CMC-CONTROL ::=
    { EncryptedPOP IDENTIFIED BY id-cmc-encryptedPOP }
cmc-decryptedPOP CMC-CONTROL ::=
    { DecryptedPOP IDENTIFIED BY id-cmc-decryptedPOP }
id-cmc-encryptedPOP OBJECT IDENTIFIER ::= {id-cmc 9}
id-cmc-decryptedPOP OBJECT IDENTIFIER ::= {id-cmc 10}
```

```
EncryptedPOP ::= SEQUENCE {
    request      TaggedRequest,
    cms          ContentInfo,
    thePOPAlgID  AlgorithmIdentifier{MAC-ALGORITHM, {POPAlgs}},
    witnessAlgID AlgorithmIdentifier{DIGEST-ALGORITHM,
                                   {WitnessAlgs}},
    witness      OCTET STRING
}
```

```
POPAlgs MAC-ALGORITHM ::= {maca-hMAC-SHA1, ...}
WitnessAlgs DIGEST-ALGORITHM ::= {mda-sha1, ...}
```

```
DecryptedPOP ::= SEQUENCE {
    bodyPartID   BodyPartID,
    thePOPAlgID  AlgorithmIdentifier{MAC-ALGORITHM, {POPAlgs}},
    thePOP       OCTET STRING
}
```

```
cmc-lraPOPWitness CMC-CONTROL ::=
    { LraPopWitness IDENTIFIED BY id-cmc-lraPOPWitness }
```

```
id-cmc-lraPOPWitness OBJECT IDENTIFIER ::= {id-cmc 11}
```

```
LraPopWitness ::= SEQUENCE {
    pkiDataBodyid BodyPartID,
    bodyIds       SEQUENCE OF BodyPartID
}
```

--

```
cmc-getCert CMC-CONTROL ::=
    { GetCert IDENTIFIED BY id-cmc-getCert }
id-cmc-getCert OBJECT IDENTIFIER ::= {id-cmc 15}
```

```
GetCert ::= SEQUENCE {
    issuerName      GeneralName,
    serialNumber    INTEGER }

```

```
cmc-getCRL CMC-CONTROL ::=
    { GetCRL IDENTIFIED BY id-cmc-getCRL }
id-cmc-getCRL OBJECT IDENTIFIER ::= {id-cmc 16}
```

```
GetCRL ::= SEQUENCE {
    issuerName      Name,
    cRLName         GeneralName OPTIONAL,
    time            GeneralizedTime OPTIONAL,
    reasons         ReasonFlags OPTIONAL }

```

```
cmc-revokeRequest CMC-CONTROL ::=
  { RevokeRequest IDENTIFIED BY id-cmc-revokeRequest }
id-cmc-revokeRequest OBJECT IDENTIFIER ::= {id-cmc 17}

RevokeRequest ::= SEQUENCE {
  issuerName          Name,
  serialNumber        INTEGER,
  reason              CRLReason,
  invalidityDate      GeneralizedTime OPTIONAL,
  passphrase          OCTET STRING OPTIONAL,
  comment             UTF8String OPTIONAL }

cmc-confirmCertAcceptance CMC-CONTROL ::=
  { CMCCertId IDENTIFIED BY id-cmc-confirmCertAcceptance }
id-cmc-confirmCertAcceptance OBJECT IDENTIFIER ::= {id-cmc 24}

CMCCertId ::= IssuerAndSerialNumber

-- The following is used to request V3 extensions be added
--   to a certificate

at-extension-req ATTRIBUTE ::=
  { TYPE ExtensionReq IDENTIFIED BY id-ExtensionReq }
id-ExtensionReq OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840)
  rsadsi(113549) pkcs(1) pkcs-9(9) 14}

ExtensionReq ::= SEQUENCE SIZE (1..MAX) OF
  Extension{{CertExtensions}}

-- The following allows Diffie-Hellman Certification Request
--   Messages to be well-formed

sa-noSignature SIGNATURE-ALGORITHM ::= {
  IDENTIFIER id-alg-noSignature
  VALUE NoSignatureValue
  PARAMS TYPE NULL ARE required
  HASHES { mda-sha1 }
}
id-alg-noSignature OBJECT IDENTIFIER ::= {id-pkix id-alg(6) 2}

NoSignatureValue ::= OCTET STRING

-- Unauthenticated attribute to carry removable data.

id-aa OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
  rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) }
```



```
aa-cmc-unsignedData ATTRIBUTE ::=
  { TYPE CMCUnsignedData IDENTIFIED BY id-aa-cmc-unsignedData }
id-aa-cmc-unsignedData OBJECT IDENTIFIER ::= {id-aa 34}

CMCUnsignedData ::= SEQUENCE {
  bodyPartPath      BodyPartPath,
  identifier         TYPE-IDENTIFIER.&id,
  content            TYPE-IDENTIFIER.&Type
}

-- Replaces CMC Status Info
--

cmc-statusInfoV2 CMC-CONTROL ::=
  { CMCStatusInfoV2 IDENTIFIED BY id-cmc-statusInfoV2 }
id-cmc-statusInfoV2 OBJECT IDENTIFIER ::= {id-cmc 25}

EXTENDED-FAILURE-INFO ::= TYPE-IDENTIFIER

ExtendedFailures EXTENDED-FAILURE-INFO ::= {...}

CMCStatusInfoV2 ::= SEQUENCE {
  cMCStatus          CMCStatus,
  bodyList           SEQUENCE SIZE (1..MAX) OF
                      BodyPartReference,
  statusString       UTF8String OPTIONAL,
  otherInfo          CHOICE {
    failInfo          CMCFailInfo,
    pendInfo          PendInfo,
    extendedFailInfo  [1] SEQUENCE {
      failInfoOID     TYPE-IDENTIFIER.&id
                      ({ExtendedFailures}),
      failInfoValue    TYPE-IDENTIFIER.&Type
                      ({ExtendedFailures}
                      {@.failInfoOID})
    }
  } OPTIONAL
}

BodyPartReference ::= CHOICE {
  bodyPartID         BodyPartID,
  bodyPartPath       BodyPartPath
}

BodyPartPath ::= SEQUENCE SIZE (1..MAX) OF BodyPartID
```

```
-- Allow for distribution of trust anchors
--

cmc-trustedAnchors CMC-CONTROL ::=
  { PublishTrustAnchors IDENTIFIED BY id-cmc-trustedAnchors }
id-cmc-trustedAnchors OBJECT IDENTIFIER ::= {id-cmc 26}

PublishTrustAnchors ::= SEQUENCE {
  seqNumber      INTEGER,
  hashAlgorithm  AlgorithmIdentifier{DIGEST-ALGORITHM,
                                     {HashAlgorithms}},
  anchorHashes   SEQUENCE OF OCTET STRING
}

HashAlgorithms DIGEST-ALGORITHM ::= {
  mda-sha1 | mda-sha256, ...
}

cmc-authData CMC-CONTROL ::=
  { AuthPublish IDENTIFIED BY id-cmc-authData }
id-cmc-authData OBJECT IDENTIFIER ::= {id-cmc 27}

AuthPublish ::= BodyPartID

-- These two items use BodyPartList

cmc-batchRequests CMC-CONTROL ::=
  { BodyPartList IDENTIFIED BY id-cmc-batchRequests }

id-cmc-batchRequests OBJECT IDENTIFIER ::= {id-cmc 28}

cmc-batchResponses CMC-CONTROL ::=
  { BodyPartList IDENTIFIED BY id-cmc-batchResponses }
id-cmc-batchResponses OBJECT IDENTIFIER ::= {id-cmc 29}

BodyPartList ::= SEQUENCE SIZE (1..MAX) OF BodyPartID

cmc-publishCert CMC-CONTROL ::=
  { CMCPublicationInfo IDENTIFIED BY id-cmc-publishCert }
id-cmc-publishCert OBJECT IDENTIFIER ::= {id-cmc 30}

CMCPublicationInfo ::= SEQUENCE {
  hashAlg      AlgorithmIdentifier{DIGEST-ALGORITHM,
                                   {HashAlgorithms}},
  certHashes   SEQUENCE OF OCTET STRING,
  pubInfo      PKIPublicationInfo
}
```

```
cmc-modCertTemplate CMC-CONTROL ::=
  { ModCertTemplate IDENTIFIED BY id-cmc-modCertTemplate }

id-cmc-modCertTemplate OBJECT IDENTIFIER ::= {id-cmc 31}

ModCertTemplate ::= SEQUENCE {
  pkiDataReference      BodyPartPath,
  certReferences         BodyPartList,
  replace               BOOLEAN DEFAULT TRUE,
  certTemplate           CertTemplate
}

-- Inform follow-on servers that one or more controls have
--   already been processed

cmc-controlProcessed CMC-CONTROL ::=
  { ControlsProcessed IDENTIFIED BY id-cmc-controlProcessed }
id-cmc-controlProcessed OBJECT IDENTIFIER ::= {id-cmc 32}

ControlsProcessed ::= SEQUENCE {
  bodyList              SEQUENCE SIZE(1..MAX) OF BodyPartReference
}

-- Identity Proof control w/ algorithm agility

cmc-identityProofV2 CMC-CONTROL ::=
  { IdentityProofV2 IDENTIFIED BY id-cmc-identityProofV2 }
id-cmc-identityProofV2 OBJECT IDENTIFIER ::= { id-cmc 33 }

IdentityProofV2 ::= SEQUENCE {
  proofAlgID            AlgorithmIdentifier{DIGEST-ALGORITHM,
                                     {WitnessAlgs}},
  macAlgId              AlgorithmIdentifier{MAC-ALGORITHM, {POPAlgs}},
  witness               OCTET STRING
}

cmc-popLinkWitnessV2 CMC-CONTROL ::=
  { PopLinkWitnessV2 IDENTIFIED BY id-cmc-popLinkWitnessV2 }
id-cmc-popLinkWitnessV2 OBJECT IDENTIFIER ::= { id-cmc 34 }

PopLinkWitnessV2 ::= SEQUENCE {
  keyGenAlgorithm       AlgorithmIdentifier{KEY-DERIVATION,
                                     {KeyDevAlgs}},
  macAlgorithm          AlgorithmIdentifier{MAC-ALGORITHM, {POPAlgs}},
  witness               OCTET STRING
}

KeyDevAlgs KEY-DERIVATION ::= {kda-PBKDF2, ...}
```

```
cmc-raIdentityWitness CMC-CONTROL ::=
  { BodyPartPath IDENTIFIED BY id-cmc-raIdentityWitness }

id-cmc-raIdentityWitness OBJECT IDENTIFIER ::= {id-cmc 35}

--
-- Allow for an End-Entity to request a change in name.
-- This item is added to RegControlSet in CRMF.
--
at-cmc-changeSubjectName ATTRIBUTE ::=
  { TYPE ChangeSubjectName IDENTIFIED BY id-cmc-changeSubjectName }

id-cmc-changeSubjectName OBJECT IDENTIFIER ::= {id-cmc 36}

ChangeSubjectName ::= SEQUENCE {
  subject          Name OPTIONAL,
  subjectAlt       GeneralNames OPTIONAL
}
(WITH COMPONENTS {..., subject PRESENT} |
 WITH COMPONENTS {..., subjectAlt PRESENT} )

--
-- Embedded response from a third party for processing
--

cmc-responseBody CMC-CONTROL ::= {
  BodyPartPath IDENTIFIED BY id-cmc-responseBody
}

id-cmc-responseBody OBJECT IDENTIFIER ::= {id-cmc 37}

--
-- Key purpose identifiers are in the Extended Key Usage extension
--

id-kp-cmcCA OBJECT IDENTIFIER ::= { id-kp 27 }
id-kp-cmcRA OBJECT IDENTIFIER ::= { id-kp 28 }
id-kp-cmcArchive OBJECT IDENTIFIER ::= { id-kp 29 }

--
-- Subject Information Access identifier
--

id-ad-cmc OBJECT IDENTIFIER ::= { id-ad 12 }

END
```

Author's Address

Jim Schaad
Soaring Hawk Consulting

EMail: jimsch@augustcellars.com