TITLE: Revised Text of Public-key and Attribute Certificate Enhancements FPDAM 4

SOURCE: Collaborative ITU and ISO/IEC meeting on the Directory, London, England, February 2003,
Geneva, Switzerland, September 2003 and March 2004 and Orlando,FL USA 2004

STATUS: For your information

Introduction

Note - This clause provides an introduction to this FPDAM. The text in this clause is not intended for inclusion in ISO/IEC 9594-8.

This FPDAM provides enhancements for public key certificate and attribute certificate frameworks extensions and certificate revocation. Mechanisms have been developed that require changes to ISO/IEC 9594-2, ISO/IEC 9594-6, ISO/IEC 9594-7 and ISO/IEC 9594-8

Specific changes set forth in this document are as follows:

Final Proposed Draft Amendment 4 to ITU Rec. X.500 (2001) I ISO/IEC 9594-1: 2001

No changes at this time.

Final Proposed Draft Amendment 4 to ITU Rec. X.501 (2001) I ISO/IEC 9594-2: 2001

Added "**entityAuthentication**" object identifier in support of the public-key and attribute certificate enhancements.

Corrected and added missing object identifier definitions.

Final Proposed Draft Amendment 4 to ITU Rec. X.511 (2001) I ISO/IEC 9594-3: 2001

No changes at this time.

Final Proposed Draft Amendment 4 to ITU Rec. X.518 (2001) I ISO/IEC 9594-4: 2001

No changes at this time.

Final Proposed Draft Amendment 4 to ITU Rec. X.519 (2001) I ISO/IEC 9594-5: 2001

No changes at this time.

Final Proposed Draft Amendment 4 to ITU Rec. X.520 (2001) I ISO/IEC 9594-6: 2001

Corrected object identifiers (commented out) in support of the pubic-key and attribute certificate enhancements.

Final Proposed Draft Amendment 4 to ITU Rec. X.521 (2001) I ISO/IEC 9594-7: 2001

Corrected object identifiers (commented out) in support of the pubic-key and attribute certificate enhancements.

Final Proposed Draft Amendment 4 to ITU Rec. X.509 (2000) I ISO/IEC 9594-8: 2001

Adds the following enhancements for public key certificate and attribute certificate frameworks extensions and certificate revocation:

- Notice of future revocation
- Notice of group of certificates
- Subject directory attributes criticality
- Enhanced certificate matching rules
- Initializing permitted subtrees, excluded subtrees and required name forms
- Defines XML privilege information
- Self issued certificates
- OCSP inclusion
- Indirect CRLs
- Certification path and naming enhancements
- Enhanced name matching
- CPs and CPSs

- SOA identifier extension and cross-certification
- Privilege policy attribute certificate
- Additional revocation reason codes
- Expired certificates on CRLs
- CRL scope extension
- Examples of name constraint use
- Name form hierarchies
- Additional definitions

Final Proposed Draft Amendment 4 to ITU Rec. X.525 (2001) I ISO/IEC 9594-9: 2001

No changes at this time.

Final Proposed Draft Amendment 4 to ITU Rec. X.530 (2001) I ISO/IEC 9594-10: 2001

No changes at this time.

Final Proposed Draft Amendment 4 to ITU Rec. X.501 (2001) | ISO/IEC 9594-2: 2001

ISO/IEC 9594-2 Information Technology — Open Systems Interconnection — The Directory: Models

FPDAM 4: Enhancements to Public-key and attribute certificate framework

Add the following definition to the end of the information object definitions in Annex A.

```
keyPurposes          ID       ::=      {ds 38}
```

Add the following to the synonyms definitions in Annex A.

```
id-kp     ID   ::= keyPurposes
```

Final Proposed Draft Amendment 4 to ITU Rec. X.520 (2001) I ISO/IEC 9594-6: 2001

ISO/IEC 9594-6 Information Technology — Open Systems Interconnection — The Directory: Selected attribute types

FPDAM 4: Enhancements to Public-key and attribute certificate framework

Add the following definitions to the end of the Attribute declarations in the ASN.1 module of Annex A.

```
-- id-at-xMLPrivilegeInfo          OBJECT IDENTIFIER   ::=   {id-at 75}
-- id-at-protPrivPolicy            OBJECT IDENTIFIER   ::=   {id-at 74}
```

Add the following definition to the end of the object class declarations in in the ASN.1 module Annex A.

```
-- id-oc-protectedPrivilegePolicy   OBJECT IDENTIFIER   ::=   {id-oc-34}
```

Add the following two definitions to the end of the Matching Rule declarations in the ASN.1 module of Annex A.

```
-- id-mr-enhancedCertificateMatch   OBJECT IDENTIFIER   ::=   {id-mr 65}
-- id-mr-sOAIdentifierMatch         OBJECT IDENTIFIER   ::=   {id-mr 66}
-- id-mr-indirectIssuerMatch        OBJECT IDENTIFIER   ::=   {id-mr 67}
```

Final Proposed Draft Amendment 4 to ITU Rec. X.521 (2001) | ISO/IEC 9594-7: 2001

ISO/IEC 9594-7 Information Technology - Open Systems Interconnection - The Directory: Selected object classes

FPDAM 4: Enhancements to Public-key and attribute certificates framework

Add the following definition to the end of the object class declarations in the ASN.1 module of Annex A.

```
-- id-oc-protectedPrivilegePolicy          OBJECT IDENTIFIER     ::=        {id-oc-34}
```

Final Proposed Draft Amendment 4 to ITU Rec. X.509 (2000) I ISO/IEC 9594-8: 2001

ISO/IEC 9594-8 Information Technology - Open systems Interconnection - The Directory: Public-key and attribute certificates framework

FPDAM 4: Enhancements to Public-key and attribute certificates frameworks

3.3 Definitions

Add the following definition and renumber definitions according to alphabetical listings.

Certification Practice Statement (CPS): A statement of the practices that a Certification Authority employs in issuing certificates.

self-issued certificate: A public-key certificate where the issuer and the subject are the same CA. A CA might use self -issued certificates, for example, during a key rollover operation to provide trust from the old key to the new key.

self-signed certificate: A special case of self-issued certificates where the private key used by the CA to sign the certificate corresponds to the public key that is certified within the certificate. A CA might use a self-signed certificate, for example, to advertise their public key or other information about their operations.

> **Note: Use of self-issued certificates and self-signed certificates issued by other than CAs are outside the scope of this standard/recommendation.**

cross certificate: A public-key or attribute certificate where the issuer and the subject/holder are different CAs or AAs respectively. CAs and AAs issue cross certificates to other CAs or AAs respectively as a mechanism to authorize the subject CA's existence (e.g. in a strict hierarchy) or to recognize the existence of the subject CA or holder AA (e.g. in a distributed trust model). The cross-certificate structure is used for both of these.

trust anchor: A trust anchor is a set of the following information in addition to the public key: algorithm identifier, public key parameters (if applicable), distinguished name of the holder of the associated private key (i.e., the subject CA) and optionally a validity period. The trust anchor may be provided in the form of a self-signed certificate. A trust anchor is trusted by a certificate using system and used for validating certificates in certification paths.

self-issued AC: An attribute certificate where the issuer and the subject are the same Attribute Authority. An Attribute Authority might use a self-issued AC, for example, to publish policy information.

3.3.1 attribute certificate

Change "3.3.1 attribute certificate:"to "3.3.1 attribute certificate (AC):"

3.3.12 certificate user

Replace the existing definition with the following new definition:

"An entity that needs to know, with certainty, the attributes and or public key of another entity".

3.3.13 certificate serial number

Replace the existing definition with the following new definition:

" An integer value, unique within the issuing authority, which is unambiguously associated with a certificate issued by that authority."

3.3.18 certificate path

Replace the existing definition with the following new definition:

"An ordered sequence of public key certificates of objects in the DIT which, together with the public key of the initial object in the path, can be processed to obtain that of the final object in the path."

3.3.25 end entity

Replace the existing definition with the following new definition:

"Either a public key certificate subject that uses its private key for purposes other than signing certificates, or an attribute certificate holder that uses its attributes to gain access to a resource, or an entity that is a relying party."

3.3.34 attribute certificate

Change "3.3.44 public key certificate:"to "3.3.44 public key certificate (PKC):"

4 Abbrieviations

Add the following new abbreviation:

AC	Attribute Certificate
OCSP	On-line Certificate Status Protocol
PKC	Public Key Certificate

7 Public-keys and public-key certificates

Add the following at the beginning of the paragraph that currently begins with "A certification path logically forms an unbroken chain …".

The **issuer** and **subject** fields of each certificate are used, in part, to identify a valid path. For each pair of adjacent certificates in a valid certification path, the value of the **subject** field in one certificate shall match the value of the **issuer** field in the subsequent certificate. In addition, the value of the **issuer** field in the first certificate shall match the DN of the trust anchor. Only the names in these fields are used when checking validity of a certification path. Names in certificate extensions are not used for this purpose.

Add the following sentence to the end of the paragraph that currently begins with "A certification path logically forms an unbroken chain …". Note this paragraph was just updated as specified above.

The **distinguishedNameMatch** matching rule, defined in 13.5.2 in ITU-T Rec. X.501|ISO/IEC 9594-2, should be used to compare the Distinguished Name (DN) in the **issuer** field of one certificate with the DN in the **subject** field of another.

Add the following at the end of the third bulleted list item "Cross-certificate:

In some situations, conflicting or overlapping requirements for constraints, such as name constraints, may require a CA to issue more than one cross-certificate to another CA.

7.3 Certificate validity

In the third bullet of the first bulleted list, replace "authorizes a different authority" with "authorizes a different entity".

In the second paragraph (begins with "Authorities that do revoke…", add the following immediately after "does not preclude the use of alternative mechanisms."

One such alternative mechanism is the Online Certificate Status Protocol (OCSP) specified in IETF RFC 2560. Using this protocol, a relying party (client) requests the revocation status of a certificate from an OCSP server. The server may use **CRLs**, or other mechanisms to check the status of the certificate and respond to the client accordingly. If OCSP can be used by relying parties to check the status of a certificate, IETF RFC 3280 contains a certificate extension (Authority Info Access) that would be included in such certificates and would provide sufficient information to access an appropriate OCSP server.

Add the following footnotes:

IETF RFC 2560, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol (OCSP), June 1999.

IETF RFC 3280, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, April 2002.

Add the following to the end of the 3rd paragraph (begins with "Certificates, including public-key…'):

Invalidity date is the date/time at which it is known or suspected that the private key was compromised or that the certificate should otherwise be considered invalid. This date may be earlier that the revocation date. In the **CRL**, invalidity date is the value contained in the **invalidityDate** entry extension.

Add a new paragraph immediately before the paragraphs that begins with "Certificates, including public-key …", as follows:

Only a CA that is authorized to issue CRLs, may choose to delegate that authority to another entity. If this delegation is done, it shall be verifiable at the time of certificate/CRL verification. The **cRLDistributionPoints** extension can be used for this purpose. The **cRLIssuer** field of this extension would be populated with the name(s) of any entities, other than the certificate issuer itself, that have been authorized to issue CRLs concerning the revocation status of the certificate in question.

Clause 8 Public-key certificate and CRL extensions

8.1 Policy handling

8.1.5 Self-issued certificates

Replace a) with the following:

a) as a convenient way of encoding the public key associated with the private key used to sign the certificate, so that it can be communicated to, and stored as trust anchors by, its certificate using systems.

Replace b) with the following:

b) for certifying additional public keys of the CA used for purposes other than those covered by category a (such as OCSP and possibly CRL signing); and

Delete this note – editing error this should have been an editor's note.

Replace c) with the following:

c) as a convenient way of encoding the public key associated with the private key used to sign the certificate, so that it can be communicated to, and stored as trust anchors by, its certificate using systems

In the paragraph following list item c: replace "self-issued certificates issued in category a) are verified" *with* "self-issued certificates of category a) are self-signed certificates and are therefore verified…".

In the 3rd paragraph after the bulleted items (4th paragraph of 8.1.5) replace the 3rd sentence (starting with "Nevertheless …" with the following:

Nevertheless, if self-issued certificates of this category are encountered in the path, they shall be processed as intermediate certificates, with the following exception: they do not contribute to the path length for purposes of processing the **pathLenConstraint** component of the **basicConstraints** extension and the *skip-certificates* values associated with the *policy-mapping-inhibitpending* and *explicit-policy-pending indicators*."

Add the following as a new paragraphs and note at the end 8.1.5:

If an authority uses the same key to sign certificates and CRLs, a single self-issued certificate of category a) shall be used. If an authority uses a different key to sign CRLs than that used to sign certificates, the authority may choose to issue two self-issued certificates of category a), one for each of the keys. In this situation, certificate users would need access to both self-issued certificates to establish separate trust anchors for certificates and CRLs signed by that authority. Alternatively, an authority may issue one self-issued certificate of category a) for certificate signing and one self-issued certificate of category b) for CRL signing. In this situation, certificate users use the key certified in the certificate of category a) as their single trust anchor for both certificates and CRLs signed by that authority. In this case, if the self-issued of category b) were to be used to verify signatures on CRLs, there is no means defined in this standard to check the validity of that certificate.

If self-issued certificates of category b) are encountered within a path, they shall be ignored.

Note: Other mechanisms for distributing CA public keys are outside the scope of this specification.

8.2 Key and policy information extensions

8.2.2 Public-key certificate and CRL extension fields

8.2.2.4 Extended key usage

Add the following to the end of clause:

This specification defines the following key purpose that can be included in the extended key usage extension. Other purposes that can also be included are defined in other specifications, such as IETF RFC 3280.

```
keyPurposes                OBJECT IDENTIFIER  ::=    {ds 38 1}
```

8.3 Subject and issuer information extensions

8.3.2 Certificate and CRL extension fields

8.3.2.3 Subject directory attributes extension

Remove the restriction that **subjectDirectoryAttributes** *extension always be non-critical*

Replace the sentence "This extension is always non-critical." with the following:

This extension may, at the option of the certificate issuer, be either critical or non-critical. A certificate using system processing this extension is not required to understand all attribute types included in the extension. If the extension is flagged critical, at least one of the attribute types contained in the extension shall be understood for the certificate to be accepted. If the extension is flagged critical and none of the contained attribute types are understood, the certificate shall be rejected.

8.4 Certification path constraint extensions

8.4.2 Certificate extension fields

8.4.2.2 Name constraints extension

Add the following as a new paragraph immediately following the paragraph that begins with "The maximum field specifies the lower bound …"

For the **directoryName** name form, a **certificate** is considered subordinate to the **base** (and therefore a candidate to be within the subtree) if the **SEQUENCE** of **RDN**s, which forms the full **DN** in **base**, is identical to the initial **SEQUENCE** of the same number of **RDN**s which forms the first part of the **DN** in the **subject** field of the **certificate**. The **DN** in the **subject** field of the **certificate** may have additional trailing **RDN**s in its sequence that do not appear in the **DN** in **base**. The **distinguishedNameMatch** matching rule is used to compare the value of **base** with the initial sequence of **RDN**s in the **DN** in the **subject** field of the certificate."

Add the following note immediately after the 4th paragraph of 8.4.2.2 as revised by TC1:

> **Note**: This example is for illustrative purposes only. How to handle names that are in the name forms of the **GeneralName** type, except the **directoryName** name form, in their hierarchical structure, is not defined in this international standard | recommendation.

Add the following new paragraphs at the end of this clause:

Note that in some cases it may be required that more than one certificate be issued from a CA to another CA in order to achieve the desired results if some of the name constraints requirements conflict. For example, assume the Acme Corporation has 20 branches in the U.S.

The Widget Corporation wants to cross-certify the central CA of Acme Corporation, but only wants the Widget community to use Acme certificates for the subjects that meets the following criteria:

- Branch1 to Branch19 of Acme Corporation, all sections are acceptable as subject;
- Branch20 of Acme Corporation, all sections are unacceptable as subject except for subject in Purchasing Section.

This could be achieved by issued two certificates as follows; the first certificate would have a **permittedSubtrees** of {base: C=US, O=Acme} and an **excludedSubtrees** of {base: C=US, O=Acme, OU=branch20}. The second certificate would have a **permittedSubtrees** of {base: C=US, O=Acme, OU=branch20, OU=Purchasing}.

Annex G contains examples of use of the name constraints extension.

8.5 Basic CRL extensions

8.5.1 Requirements

Add the following new requirements:

h) In addition to CRLs publishing notification that certificates have been revoked, there is a requirement to publish notification that certificates will be revoked as of a specified date and time in the future.

i) There is a requirement to provide more efficient ways to indicate in a CRL that a set of certificates has been revoked.

8.5.2 CRL and CRL entry extension fields

8.5.2.2 Reason code extension

*Add the following immediately below the asn.1 for the **reasonCode** extension:*

- **unspecified** can be used to revoke certificates for reasons other than the specific codes.

8.5.2.5 CRL scope extension

Delete the CRL scope extension section 8.5.2.5 entirely.

8.5.2.6 Status referral extension

Add the following to the end of the first paragraph:

Any CRL containing this extension shall not be used as the source for a relying party to check revocation status of any certificate. Rather, a CRL containing this extension may be used by a relying party as an additional tool to locate the appropriate CRLs for checking revocation status.

Add the following two new CRL extensions:

8.5.2.10 To be revoked certificates extension

This CRL extension allows for notification that certificates will be revoked as of a specified date and time in the future. The **toBeRevoked** extension is used to specify the reason for the certificate revocation, the date and time at which the certificate will be revoked, and the group of certificates to be revoked. Each list can contain a single certificate serial number, a range of certificate serial numbers or a named **subtree**. These certificates may be public-key certificates or attribute certificates.

```
toBeRevoked  EXTENSION  ::= {
  SYNTAX            ToBeRevokedSyntax
  IDENTIFIED BY     id-ce-toBeRevoked }

ToBeRevokedSyntax::=      SEQUENCE SIZE(1..MAX) OF ToBeRevokedGroup

ToBeRevokedGroup ::=      SEQUENCE {
  certificateIssuer  [0]  GeneralName OPTIONAL,
  reasonInfo         [1]  ReasonInfo    OPTIONAL,
  revocationTime          GeneralizedTime,
  certificateGroup        CertificateGroup }

ReasonInfo                ::=     SEQUENCE {
  reasonCode              CRLReason,
  holdInstructionCode     HoldInstruction OPTIONAL }

CertificateGroup     ::=  CHOICE {
  serialNumbers           [0]     CertificateSerialNumbers,
  serialNumberRange       [1]     CertificateGroupNumberRange,
  nameSubtree             [2]     GeneralName }

CertificateGroupNumberRange ::= SEQUENCE {
  startingNumber     [0]  INTEGER,
  endingNumber       [1]  INTEGER }

CertificateSerialNumbers ::= SEQUENCE SIZE(1..MAX) OF CertificateSerialNumber
```

The **certificateIssuer** field, if present, identifies the authority (CA or AA) that issued all the certificates listed in this **ToBeRevokedGroup**. If **certificateIssuer** is omitted, it defaults to the CRL issuer name.

The **reasonInfo** field, if present, identifies the reason for the certificate revocations. If present, this field indicates that all certificates identified in **ToBeRevokedGroup** will be revoked for the reason indicated in this field. If **reasonCode** contains the value **certificateHold**, the **holdInstructionCode** may also be present. If present, **holdInstructionCode** indicates the action to be taken on encountering any of the certificates identified in **RevokedGroup**. This action should only be taken, after the revocation time indicated in the **revocationTime** field has passed.

The **revocationTime** field indicates the date and time at which this group of certificates will be revoked and should therefore be considered invalid. This date shall be later than the **thisUpdate** time of the CRL containing this extension. If **revocationTime** is before the **nextUpdate** time of the CRL containing this extension, the certificates shall be considered revoked between the **revocationTime** and the **nextUpdate** time by a relying party using a CRL containing this extension. Otherwise this is a notice that at specified time in the future these certificates will be revoked. Once the revocation time has passed either the CA has revoked the certificate or not. If it has revoked the certificate, future CRLs shall include this on the list of revoked certificates, at least until the certificate expires. If the CA has not revoked the certificate, but still intends to revoke it in the future, it may include the certificate in this extension on subsequent CRLs with a revised **revocationTime**. If the CA no longer intends to revoke the certificate, it may be excluded from all subsequent CRLs and the certificate shall not be considered revoked.

The **certificateGroup** field lists the set of certificates to be revoked. This field identifies the certificates issued by the authority identified in **certificateIssuer** to be revoked at the date/time identified in **revocationTime**. This set of certificates is not further refined by any outside controls (e.g. **issuingDistributionPoint**).

If **serialNumbers** is present, the certificate(s) with serial numbers indicated in this field, and issued by the identified certificate issuer, will be revoked at the specified time.

If **serialNumberRange** is present, all certificates in the range beginning with the starting serial number and ending with the ending serial number and issued by the identified certificate issuer will be revoked at the specified time.

If **nameSubtree** is present, all certificates with a subject/holder name that is subordinate to the specified name and issued by the identified certificate issuer will be revoked at the specified time. If the **nameSubtree** contains a DN then all DNs associated with the subject of a public-key certificate (i.e. **subject** field and **subjectAltNames** extension) or **holder** field of an attribute certificate need to be considered. For other name forms the **subjectAltNames** extension of public-key certificates and the **holder** field of attribute certificates need to be considered. If at least one of the names associated with the subject/holder, contained in the certificate, is within the subtree specified in **nameSubtree**, that certificate will be revoked at the specified time. As with the **nameConstraints** extension, not all name forms are appropriate for **subtree** specification. Only those that have recognized subordination rules should be used in this extension.

This extension may, at the option of the CRL issuer, be flagged critical or non-critical. As the information provided in this extension applies to revocations, which will occur in the future, it is recommended that it be flagged non-critical, reducing the risk of problems with interoperability and backward compatibility.

8.5.2.11 Revoked group of certificates extension

A set of certificates that has been revoked can be published using the following CRL extension. Each list of certificates to be revoked is associated with a specific certificate issuer and revocation time. Each list can contain a range of certificate serial numbers or a named subtree. These certificates may be public-key certificates or attribute certificates.

```
revokedGroups  EXTENSION  ::= {
  SYNTAX            RevokedGroupsSyntax
  IDENTIFIED BY     id-ce-RevokedGroups }

RevokedGroupsSyntax ::=   SEQUENCE SIZE (1..MAX) OF RevokedGroup

RevokedGroup ::=    SEQUENCE {
  certificateIssuer    [0]     GeneralName OPTIONAL,
  reasonInfo           [1]     ReasonInfo OPTIONAL,
  invalidityDate       [2]     GeneralizedTime OPTIONAL,
  revokedcertificateGroup      RevokedCertificateGroup }

RevokedCertificateGroup ::= CHOICE {
  serialNumberRange            NumberRange,
  nameSubtree                  GeneralName }
```

The **certificateIssuer** field, if present, identifies the authority (CA or AA) that issued all the certificates listed in this **RevokedGroup**. If **certificateIssuer** is omitted, it defaults to the CRL issuer name.

The **reasonInfo** field, if present, identifies the reason for the certificate revocations. If present, this field indicates that all certificates identified in **RevokedGroup** were revoked for the reason indicated in this field. If **reasonCode** contains the value **certificateHold**, the **holdInstructionCode** may also be present. If present, **holdInstructionCode** indicates the action to be taken on encountering any of the certificates identified in **RevokedGroup**.

The **invalidityDate** field, if present, indicates the time from which all certificates identified in **RevokedGroup** should be considered invalid. This date shall be earlier than the date contained in **thisUpdate** field of the CRL. If omitted, all certificates identified in **RevokedGroup** should be considered invalid at least from the time indicated in the **thisUpdate** field of the CRL. If the status of the certificate prior to the **thisUpdate** time is critical to a certificate using system (e.g. to determine whether a digital signature that was created prior this CRL issuance occurred while the certificate was still valid or after it had been revoked), additional revocation status checking techniques will be required to determine the actual date/time from which a given certificate should be considered invalid.

The **revokedCertificateGroup** field lists the set of certificates that have been revoked. This field identifies the certificates issued by the authority identified in **certificateIssuer** revoked under the specified conditions. This set of certificates is not further refined by any outside controls (e.g. **issuingDistributionPoint**).

If **serialNumberRange** is present, all certificates containing certificate serial numbers within the specified range, issued by the identified certificate issuer are applicable.

If **nameSubtree** is present, all certificates with a subject/holder name that is subordinate to the specified name and issued by the identified certificate issuer will be revoked at the specified time. If the **nameSubtree** contains a DN then all DNs associated with the subject of a public-key certificate (i.e. **subject** field and **subjectAltNames** extension) or **holder** field of an attribute certificate need to be considered. For other name forms the **subjectAltNames** extension of public-key certificates and the **holder** field of attribute certificates need to be considered. If at least one of the names associated with the subject/holder, contained in the certificate, is within the subtree specified in **nameSubtree**, that certificate has been revoked. As with the **nameConstraints** extension, not all name forms are appropriate for subtree specification. Only those that have recognized subordination rules should be used in this extension.

This extension is always flagged critical. Otherwise a certificate using system may incorrectly assume that certificates, identified as revoked within this extension, are not revoked. When this extension is present it may be the only indication of revoked certificates in a CRL (i.e. the **revokedCertificates** may be empty) or it may list revoked certificates that are in addition to those indicated in the **revokedCertificates** field. A revoked certificate shall not be listed both in the **revokedCertificates** field and in this extension.

Add a new CRL extension for expired certificates on CRL as follows:

8.5.2.12 Expired certificates on CRL extension

This CRL extension field indicates that the CRL includes revocation notices for expired certificates.

```
expiredCertsOnCRL EXTENSION  ::= {
  SYNTAX       ExpiredCertsOnCRL
  IDENTIFIED BY      id-ce-expiredCertsOnCRL }

ExpiredCertsOnCRL :: = GeneralizedTime
```

This extension is always non-critical.

The scope of a CRL containing this extension is extended to include the revocation status of certificates that expired at the exact time specified in the extension or after that time. If limitations in the CRL's scope are specified (by either reason codes or by distribution points), that applies to expired certificates as well. The revocation status of a certificate shall not be updated once the certificate has expired.

8.6 CRL distribution points and delta-CRL extensions

8.6.2 CRL distribution point and delta-CRL extension fields

8.6.2.1 CRL distribution points extension

Add the following sentence to the end of the paragraph that begins "The **cRLIssuer** *component …":*

If any entity, other than the certificate issuer, is authorized E.g. possesses a certificate with **cRLSign** set in **KeyUsage**, to issue CRLs that address revocation status for this certificate, the name(s) of each such entity, as provided by the CA, shall be included in this field.

> Note: Although this standard does not specify a method to provide a secure identification of the delegated CRL issuer, one example is a process where the certificate issuing authority issues to the indirect CRL issuer a certificate that indicates the delegation of issuing CRLs and that certifies the CRL issuer's public key as being used to sign the indirectly issued CRLs by setting the cRLSign bit in the keyUsage extension.

8.6.2.2 Issuing distribution point extension

Add the following as a new paragraph immediately following the paragraph that begins with "The ***distributionPoint*** *component contains…":*

If the CRL contains an **issuingDistributionPoint** extension with the **distributionPoint** field present, at least one name for the distribution point in the certificate (e.g., **cRLDistributionPoints**, **freshestCRL**, **issuer**) shall match a name for the distribution point in the CRL. Also, it may be the case that only the **nameRelativeToCRLIssuer** field is present. In that case, a name comparison would be done on the full DN, constructed by appending the value of the **nameRelativeToCRLIssuer** to the DN found in the **issuer** field of the CRL. If the names being compared are DNs (as opposed to names of other forms within the **GeneralNames** construct), the **distinguishedNameMatch** matching rule is used to compare the two DNs for equality.

*Add a new paragraph immediately following the one that begins with "*If **onlyContainsUserCerts** *is true…" as follows:*

If CRLs are partitioned by reason code, and the reason code changes for a revoked certificate (causing the certificate to move from one CRL stream to another), it is necessary to continue to include the certificate on the CRL stream for the old revocation reason until the **nextUpdate** times of all CRLs, that do not list the certificate, on the CRL stream for the new reason code have been reached.

10 Certification path processing procedure

10.1 Path processing inputs

Add new subclauses h), i), and j) as follows:

h) an *initial-permitted-subtrees-set* containing an initial set of subtree specifications defining subtrees within which subject names (of the name form used to specify the subtrees) are permitted. In the certificates in the certification path all subject names of a given name form, for which initial permitted subtrees are defined, shall fall within the permitted subtrees set for that given name form. This input may also contain the special value *unbounded* to indicate that initially all subject names are acceptable. For clause 10, subject names are those name values appearing in the **subject** field or the **subjectAltName** extension;

i) an *initial-excluded-subtrees-set* containing an initial set of subtree specifications defining subtrees within which the subject names in the certificates in the certification path cannot fall. This input may also be an empty set to indicate that initially no subtree exclusions are in effect;

j) an *initial-required-name-forms* containing an initial set of name forms indicating that all certificates in the path must include a subject name of at least one of the specified name forms. This input may also be an empty set to indicate that no specific name forms are required for subject names in the certificates.

10.4 Initialization step

Replace bullet b), c), and d) with the following:

b) Initialize the *permitted-subtrees* variable to the *initial-permitted-subtrees-set* value;

c) Initialize the *excluded-subtrees* variable to the *initial-excluded-subtrees-set* value;

d) Initialize the *required-name-forms* variable to the *initial-required-name-forms* value; *Editor's note — this is a replacement for the item d) stated in Technical Corrigendum 1 to the 4th edition of this specification.*

11 PKI directory schema

11.2 PKI directory attributes

11.2.9 Certificate policy attribute

Add the following note to the end of this clause:

Note: The option to include a hash in this attribute is purely to perform an integrity check against data located from a source other than the directory. The HASH stored in the Directory needs to be protected. Directory security services, including strong authentication, access control and/or signed attributes could be used for this purpose. In addition, even if the HASH matches the original CP/CPS document, there are additional security requirements to ensure that the original specification itself is the correct document (e.g. the document is signed by an appropriate authority).

11.3 PKI directory matching rules

Add the following to the end of the first paragraph:

The enhanced certificate matching rule provides the ability to perform more sophisticated matching against certificates held in directory entries.

Add the following new matching rule, which includes the basic matching capability of the ***certificateMatch*** *matching rule with the extensions described below.*

11.3.10 Enhanced certificate match

The enhanced certificate match rule compares a presented value with an attribute value of type **Certificate**. It selects one or more certificates on the basis of various characteristics.

```
enhancedCertificateMatch MATCHING-RULE  ::= {
  SYNTAX      EnhancedCertificateAssertion
  ID          id-mr-enhancedCertificateMatch }

EnhancedCertificateAssertion  ::= SEQUENCE {
  serialNumber              [0]    CertificateSerialNumber    OPTIONAL,
  issuer                    [1]    Name                       OPTIONAL,
  subjectKeyIdentifier      [2]    SubjectKeyIdentifier       OPTIONAL,
  authorityKeyIdentifier    [3]    AuthorityKeyIdentifier     OPTIONAL,
  certificateValid          [4]    Time                       OPTIONAL,
  privateKeyValid           [5]    GeneralizedTime            OPTIONAL,
  subjectPublicKeyAlgID     [6]    OBJECT IDENTIFIER          OPTIONAL,
  keyUsage                  [7]    KeyUsage                   OPTIONAL,
  subjectAltName            [8]    AltName                    OPTIONAL,
  policy                    [9]    CertPolicySet              OPTIONAL,
  pathToName                [10]   GeneralNames               OPTIONAL,
  subject                   [11]   Name                       OPTIONAL,
  nameConstraints           [12]    NameConstraintsSyntax     OPTIONAL
}

(ALL EXCEPT ({ --none; at least one component shall be present -- }))

AltName  ::= SEQUENCE {
  altnameType        AltNameType,
  altNameValue       GeneralName OPTIONAL }
```

The directory search operation allows for multiple values of **EnhancedCertificateAssertion** to be combined in filter specifications, including and/or logic. This matching rule returns TRUE if all of the components that are present in the presented value match the corresponding components of the attribute value, as follows:

Matching for **serialNumber**; issuer; **subjectKeyIdentifier**; **authorityKeyIdentifier**; **certificateValid**, **privateKeyValid**, **policy**, **subject**, and **nameConstraints** components is as defined for the same components in the **certificateMatch** matching rule.

subjectAltName component contains an **altNameType** and optional **altNameValue** fields. If **altNameValue** is present, the value shall of the same name form as indicated in **altNameType**.

subjectAltName matches if at least one of the following conditions is true:

- The presented value contains only the **altNameType** component and the stored attribute value contains the subject alternative name extension with an **AltNames** component of the same type as indicated in the presented value;
- The presented value contains both the **altNameType** and **altNameValue** components and the stored attribute value contains the subject alternative name extension with an **AltNames** component of the same type and value indicated in the presented value.

subjectAltName match fails if at least one of the following conditions is true:

- The stored attribute value does not contain the subject alternative name extension;
- The stored attribute value contains the subject alternative name extension but the **altNames** component does not include the same type as identified in the presented value;
- The presented value contains both the **altNameType** and **altNameValue** components and the stored attribute value contains the subject alternative name extension with an **AltNames** component of the same type indicated in the presented value, but the stored value does not contain the same value of that type as in the presented value;

subjectAltName match is undefined under the following condition:

- If the presented value contains both the **altNameType** and **altNameValue** components and the stored attribute value contains the subject alternative name extension with an **AltNames** component of the same type indicated in the presented value, but the type is one for which the directory is unable to compare values for purposes of determining a match. This may be because the name form is not appropriate for matching or because the directory is unable to perform the required comparisons;

pathToName matches unless the certificate has a name constraints extension which inhibits the construction of a certification path to any of the presented name values. For example, if attempting to retrieve certificates that form a path to a user certificate which has a subject value of “dc=com; dc=corporate; cn=john.smith”, it may be useful to include an assertion in the search operation containing this DN in the **pathToName** component. A stored certificate that contained a name constraints extension that excluded the complete subtree below base “dc=com; dc=company A” would fail in certification path validation to that user certificate and would therefore not be a matched value for this sample assertion.

13 Attribute Authority, SOA and Certification Authority relationship

Add the following to the end of clause:

The delegation model that is fully described in this specification is one where the privilege delegator is itself both claimant for the privilege and an AA that can issue certificates delegating that privilege to others. The other model that is permitted, but not fully described in this Specification, allows for an independent Delegation Service (DS) in which the entity issuing certificates that delegate privilege cannot itself act as a claimant for that privilege. The DS model is particularly relevant to environments that wish to maintain some central management over the set of privileges delegated within their domain. For example a set of one or more DS servers performing delegation, rather than individual privilege holders, allows the total set of privileges delegated within an environment to be determined from a centralized facility and enables policy and management decisions to be modified accordingly. Two distinct deployment models are possible. In one model, privilege is assigned by an SOA to privilege holders and those holders are authorized to delegate that privilege to others.

Two delegation models are described in this specification. The first delegation model is one where the privilege delegator is itself both claimant for the privilege and an AA that can issue certificates delegating that privilege to others. The second model allows for an independent Delegation Service (DS) in which the entity issuing certificates that delegate privilege cannot itself act as a claimant for that privilege. The DS model is particularly relevant to environments that wish to maintain some central management over the set of privileges delegated within their domain. For example a set of one or more DS servers performing delegation, rather than individual privilege holders, allows the total set of privileges delegated within an environment to be determined from a centralized facility and enables policy and management decisions to be modified accordingly. Two distinct deployment models are possible for DS servers. In one model, a privilege is assigned by an SOA to privilege holders and those holders are authorized to delegate that privilege to others. However, rather than issue the attribute certificates that delegate the privilege themselves, the privilege holder requests the DS to delegate that privilege on their behalf. The DS does not itself hold that privilege and therefore cannot act as a claimant for that privilege, however the DS is authorized by the SOA to issue attribute certificates on behalf of other privilege holders. The second deployment model is similar to the first with the following exception. The DS is actually a holder that is assigned the privilege to be delegated, but the DS is not authorized to act as a claimant for the privilege, only as a delegator. In this case, the noAssertion extension must be set in the AC issued to the DS by the SOA. The DS is termed an indirect issuer.

In both deployment models the SOA issues attributes/privileges to subordinate AAs. The AAs then request the DS to issue a subset of these privilege attributes to other holders. In the second deployment model the DS can check that an AA is delegating within the overall scope set by the SOA, in the first deployment model the DS cannot check and the relying party will have to check that delegation was performed correctly.

14 PMI models

Add a new attribute type for XML encoded privilege information as follows:

14.5 XML privilege information attribute

The specification of privileges is generally an application-specific issue that is outside the scope of this Specification. While this attribute does not define any specific privilege information, it provides a container attribute in which XML-encoded privileges can be conveyed in attribute certificates.

```
xmlPrivilegeInfo     ATTRIBUTE ::= {
  WITH SYNTAX        UTF8String –contains XML-encoded privilege information
  ID                 id-at-xMLPrivilegeInfo }
```

The XML schema for the role attribute type can be defined either with ASN.1 or with XSD.

The XML contained within the **UTF8String** needs to be self-identifying.

The following is an ASN.1 schema defining an XML role attribute type. It is followed by an XSD specification for the same attribute type, and by an example XML instance. The example instance is a valid instance for both the ASN.1 and the XSD schema instances, and can be validated by either ASN.1 or XSD tools.

The example schema defines a role attribute with an ID, an issuing authority and the name of the role.

```
CERTIFICATE-ATTRIBUTE DEFINITIONS ::=
BEGIN
  Role ::= [UNCAPITALIZED] SEQUENCE {
      id            [ATTRIBUTE] XML-ID,
      authorities     SEQUENCE (1..MAX) OF
                        authority UTF8String,
      name            UTF8String }

  XML-ID  ::= UTF8String
END
```

The following XSD schema is an alternative (exactly equivalent) definition:

```
<schema xmlns="http://www.w3.org/2000/08/XMLSchema">
 <element name="role">
  <attribute name="id" type="ID"/>
  <complexType>
   <sequence>
    <element name="authorities">
     <complexType>
      <sequence>
       <element name="authority" type="string" minOccurs="1" maxOccurs="*"/>
      </sequence>
     </complexType>
    </element>
    <element name="name" type="string"/>
   </sequence>
  </complexType>
 </element>
</schema>
```

An example of an instance conforming to the above schema definitions, that would be a value of the **xMLPrivilegeInfo** attribute type would be:

```
<role id="123" xmlns="http://www.example.org/certificates/attribute">
<authorities>
<authority>Fictitious Organization</authority>
</authorities>
<name>manager</name>
</role>
```

15 Privilege management certificate extensions

15.1 Basic privilege management extensions

15.1.2 Basic privilege management extension fields

Add the following two new extension to the list:

e) Indirect issuer

f) No assertion.

Add the following extension field:

15.1.2.5 Indirect issuer extension

In some environments, privilege may be delegated indirectly. In such cases, the delegator requests that an AA issue a certificate delegating privilege on their behalf to another entity. The indirect issuer field is used in either an attribute certificate or a public-key certificate issued to an AA by an SOA. Presence of this extension means that the subject AA is authorized by that SOA to act as a proxy and issue certificates that delegate privilege, on behalf of other delegators.

```
indirectIssuer EXTENSION ::=
  {
  SYNTAX            BOOLEAN
  IDENTIFIED BY     id-ce-indirectIssuer
  }
```

This extension is always non-critical.

The indirect issuer matching rule compares for equality a presented value with an attribute value of type **AttributeCertificate.**

```
indirectIssuerMatch MATCHING-RULE  ::= {
SYNTAX  BOOLEAN
ID       id-mr-indirectIssuerMatch }
```

This matching rule returns TRUE if the stored value contains the **indirectIssuer** extension and if the value that is present in the presented value match the stored value.

15.1.2.6 No assertion extension.

If present, this extension indicates that the AC holder cannot assert the privileges indicated in the attributes of the AC. This field can only be inserted into AA ACs, and not into end entity ACs. If present, this extension shall always be marked critical.

```
noAssertion EXTENSION ::=

{

SYNTAX NULL

IDENTIFIED BY { id-ce-noAssertion }

}
```

15.3 Source of Authority extensions

15.3.2 SOA extension fields

15.3.2.1 SOA identifier extension

Add the following as a new paragraph immediately before the sentence "This extension is always non-critical."

Cross-certification applies only to public-key certificates and not to attribute certificates. Therefore a cross-certificate issued to the CA that is the issuer of a certificate containing the SOA identifier extension does not provide transitive trust to the SOA identified in this extension.

Add the following as a new clause

15.3.2.1.1 SOA identifier match

The SOA identifier matching rule compares a presented value with an attribute value of type **Certificate**.

```
sOAIdentifierMatch MATCHING-RULE ::= {
  SYNTAX      NULL
  ID          id-mr-sOAIdentifierMatch }
```

This matching rule returns TRUE if the stored value contains an SOA Identifier extension.

15.5 Delegation extensions

15.5.1 Requirements

Add the following new requirement to the list:

e) There is a requirement for an independent Delegation Service (DS) to issue certificates that delegate privileges, whilst the DS server cannot itself act as a claimant for those privileges.

15.5.2 Delegation extension fields

Add the following extension field:

e) Indirect Issuer

Add the following new section:

15.5.2.5 Indirect issuer extension

In some environments, privilege may be delegated indirectly. In such cases, the delegator requests that a DS server issue a certificate delegating privilege on their behalf to another entity. The indirect issuer field is used in either an attribute certificate or a public-key certificate issued to a DS server by an SOA. Presence of this extension means that the subject AA (the DS server) is authorized by that SOA to act as a proxy and issue certificates that delegate privilege, on behalf of other delegators.

```
indirectIssuer EXTENSION ::=
  {
  SYNTAX            NULL
  IDENTIFIED BY     id-ce-indirectIssuer
  }
```

This extension is always non-critical.

The indirect issuer matching rule compares for equality a presented value with an attribute value of type **AttributeCertificate.**

```
indirectIssuerMatch MATCHING-RULE  ::= {
SYNTAX  NULL
ID        id-mr-indirectIssuerMatch }
```

This matching rule returns TRUE if the stored value contains the **indirectIssuer** extension and if the value that is present in the presented value match the stored value.

15.5.x) **Issued on Behalf Of**

This extension is inserted into an AC by an indirect issuer (DS server). It indicates the AA that has requested the DS server to issue the AC, and allows the delegation chain to be constructed and validated.

```
issuedOnBehalfOf EXTENSION ::= {
        SYNTAX GeneralName
        ID id-ce-issuedOnBehalfOf }
```

The GeneralName is the name of the AA who has asked the indirect issuer (DS server) to issue this AC

The issuer of this AC must have been granted the privilege to issue ACs on behalf of other AAs by an SOA, through the IndirectIssuer extension in its AC.

This extension may be critical or non-critical as necessary to ensure delegation path validation.

16 Privilege path processing procedure

16.3 Delegation processing procedure

Add the following new *text to the end of this clause:*

In the case where attribute certificates are issued by an indirect issuer (DS) the relying party should fully validate the delegation chain as follows.

i) Starting with the end entity AC, the RP extracts the issuer name and the issuedOnBehalfOf name.

ii) The RP retrieves the AC of the issuer and validates that the issuer is an indirect issuer of the SOA (i.e. has the indirectIssuer extension).

iii) The RP retrieves the AC of the issuedOnBehalfOf AA and validates that the AA has a superset of the privilege attributes issued to the end entity.

The RP recurses to step ii) using the AC of the AA, and thereby moves up the chain until it arrives at the AC of an AA that is issued by the SOA.

17 PMI directory schema

17.1 PMI directory object classes

Add the following new policy object class as follows:

17.1.7 Protected privilege policy object class

The protected privilege policy object class is used in defining entries for objects that contain privilege policies protected within attribute certificates.

```
protectedPrivilegePolicy        OBJECT-CLASS        ::= {
        SUBCLASS OF             {top}
        KIND                    auxiliary
        MAY CONTAIN             {protPrivPolicy }

        ID                              id-oc-protectedPrivilegePolicy }
```

17.2 PMI Directory attributes

Add the following two new policy attributes as follows:

17.2.8 Protected privilege policy attribute

The protected privilege policy attribute contains privilege policies, protected within attribute certificates.

```
protPrivPolicy      ATTRIBUTE    ::= {
      WITH SYNTAX                       AttributeCertificate
      EQUALITY MATCHING RULE            attributeCertificateExactMatch
      ID                                id-at-protPrivPolicy }
```

Note that unlike typical attribute certificates, those within the **protPrivPolicy** attribute contain privilege policies, not privileges. The issuer and holder components of these attribute certificates identify the same entity. The attribute that is included in the attribute certificate contained within the **protPrivPolicy** attribute is either the **privPolicy** attribute or the **xmlPrivPolicy** attribute.

17.2.9 Protected privilege policy attribute

The XML protected privilege policy attribute contains XML encoded privilege policy information.

```
xmlPrivPolicy       ATTRIBUTE    ::= {
      WITH SYNTAX         UTF8String –contains XML-encoded privilege policy information
      ID                  id-at-xMLPprotPrivPolicy }
```

Annex A

Add the following new object class definitions to the end of the "object identifier assignments" in Annex A.2: Certificate extension module.

```
id-ce-toBeRevoked                      OBJECT IDENTIFIER ::=     {id-ce 58}
id-ce-RevokedGroups                    OBJECT IDENTIFIER ::=     {id-ce 59}
id-ce-expiredCertsOnCRL                OBJECT IDENTIFIER ::=     {id-ce 60}
```

Add the following new object class definitions to the end of the "attribute certificate extensions" in Annex A.3: Attribute certificate framework module.

```
id-ce-indirecIssuer                   OBJECT IDENTIFIER ::=    {id-ce 61}

id-ce-noAssertion                      OBJECT IDENTIFIER ::=     {id-ce 62}
```

Add the following new object class definitions to the end of the "matching rule OIDs" in Annex A.2: Certificate extension module.

```
    id-mr-enhancedCertificateMatch     OBJECT IDENTIFIER   ::   {id-mr 65}
```

Add the following new object class definitions to the end of the "PMI matching rules" in Annex A.3: Attribute certificate framework module.

```
    id-mr-sOAIdentifierMatch          OBJECT IDENTIFIER   ::     {id-mr 66}

    id-mr-indirectIssuerMatch        OBJECT IDENTIFIER    ::=   {id-mr 67}
```

Add the following new object class definitions to the end of the "directory attributes" in Annex A.3: Attribute certificate framework module.

```
id-at-protPrivPolicy                  OBJECT IDENTIFIER ::=     {id-at 74}

id-at-xMLPrivilegeInfo                 OBJECT IDENTIFIER  ::=     {id-at 75}
```

Add the following new object class definitions to the end of the "object classes" in Annex A.3: Attribute certificate framework module.

```
 id-oc-protectedPrivilegePolicy          OBJECT IDENTIFIER ::=  {id-oc 34}
```

An amended Annex A appears at the end of this document. Note that the Annex also contains the ASN,1 statements of several approved Technical Corrigenda.

Replace the existing Annex B with the following:

Annex B

CRL Generation and Processing Rules

(This annex forms an integral part of this Recommendation | International Standard)

B.1 Introduction

A relying party (certificate user), needs the ability to check the revocation status of a certificate in order to determine whether or not to trust that certificate. Certificate Revocation Lists (CRL) are one mechanism for relying parties to obtain the revocation information. Other mechanisms may also be used, but are outside the scope of this Specification.

This annex addresses the use of CRLs for certificate revocation status checking by relying parties. Various authorities may have different policies regarding their issuance of revocation lists. For instance, in some cases the certificate issuing authority may authorize a different authority to issue a certificate revocation list for the certificates it issues. Some authorities may combine revocation of end-entity and CA-certificates into a single list while other authorities may split these into separate lists. Some authorities may partition their certificate population onto CRL fragments and some authorities may issue delta updates to a revocation list between regular CRL intervals. As a result, relying parties need to be able to determine the scope of the CRLs they retrieve to enable them to ensure they have the complete set of revocation information covering the scope of the certificate in question for the revocation reasons of interest, given the policy under which they are working. This annex provides a mechanism for the relying parties to determine the scope of retrieved CRLs.

This annex is written for revocation status checking of public-key certificates using CRLs, Full and Complete End-Entity CRLs (EPRLs) and Certification Authority Revocation Lists (CARLs). However, this description can also be applied to revocation status checking of attribute certificates using Attribute Certificate Revocation Lists (ACRL) and Attribute Authority Revocation Lists (AARL). For purposes of this annex, ACRL can be considered in place of CRL, EPRL can be full and complete end-entity ACRL, and AARL in place of CARL; Similarly, the directory attributes identified in B.4 shall be mapped to those for the AARL and ACRL and the fields identifying certificate types in the Issuing Distribution Point extension can be mapped to those applicable to PMI.

B.1.1 CRL types

CRLs of one or more of the following types may be available to a relying party, based on the revocation aspects of the policy of the certificate issuing authority:

- Full and complete CRL;
- Full and complete end-entity CRL (EPRL);
- Full and complete Certification Authority Revocation List (CARL);
- Distribution Point CRL, EPRL or CARL;
- Indirect CRL, EPRL or CARL (ICRL);
- Delta CRL, EPRL or CARL;
- Indirect dCRL, EPRL or CARL.

A full and complete CRL is a list of all revoked end-entity and CA-certificates issued by an authority for any and all reasons.

A full and complete EPRL is a list of all revoked end-entity certificates issued by an authority for any and all reasons.

A full and complete CARL is a list of revoked CA-certificates issued by an authority for any and all reasons.

A distribution point CRL, EPRL or CARL is one that covers all or a subset of certificates issued by an authority. The subset could be based on a variety of criteria.

An indirect CRL, EPRL or CARL (ICRL) is a CRL that contains a list of revoked certificates, in which some or all of those certificates were not issued by the authority signing and issuing the CRL.

A delta CRL, EPRL or CARL is a CRL that only contains changes to a CRL that is complete for the given scope at the time of the CRL referenced in the dCRL. Note that the referenced CRL might be one that is complete for the given scope or it might be a dCRL that is used to locally construct a CRL that is complete for the given scope.

All of the above CRL types (except for the dCRL) are CRL types that are complete for their given scope. A dCRL shall be used in conjunction with an associated CRL that is complete for the same scope in order to form a complete picture of the revocation status of certificates.

An indirect delta-CRL, EPRL or CARL is a CRL which only contains changes to a set of one or more CRLs, that are complete for their given scopes and in which some or all of those certificates may not have been issued by the authority signing and issuing this CRL.

Within this annex as well as this Specification, "Scope of a CRL" is defined by two independent dimensions. One dimension is the set of certificates covered by the CRL. Another dimension is the set of reason codes covered by the CRL. The scope of a CRL can be determined in one or more of the following ways:

- Issuing Distribution Point (IDP) extension in the CRL; or
- Other means, outside the scope of this Specification.

B.1.2 CRL processing

If a relying party is using CRLs as the mechanism to determine if a certificate is revoked, they shall use the appropriate CRL(s) for that certificate. This annex describes a procedure for obtaining and processing appropriate CRLs by walking through a number of specific steps. An implementation functionally equivalent to the external behaviour resulting from this procedure shall also be considered compliant with this annex and the associated Specification. The algorithm used by a particular implementation to derive the correct output (i.e. revocation status for a certificate) from the given inputs (the certificate itself and input from local policy) is not standardized. For example, although this procedure is described as a sequence of steps to be processed in order, an implementation may use CRLs which are in its local cache rather than retrieving CRLs each time it processes a certificate, provided those CRLs are complete for the scope of the certificate and do not violate any of the parameters of the certificate or policy.

The following general steps are described in B.2 through B.5 below:

1) Determine Parameters for CRLs;
2) Determine CRLs Required;
3) Obtain the CRLs;
4) Process the CRLs.

Step 1) identifies the parameters from the certificate and elsewhere that will be used to determine which types of CRLs are required.

Step 2) applies the values of the parameters to make the determination.

Step 3) identifies the directory attributes from which the CRL types can be retrieved.

Step 4) describes the processing of appropriate CRLs.

B.2 Determine parameters for CRLs

Information located in the certificate itself, as well information from the policy under which the relying party is operating, provide the parameters for determining the appropriateness of candidate CRLs. The following information is required to determine which CRL types are appropriate:

- Certificate type (i.e. end-entity or CA);
- Critical CRL Distribution Point;
- Critical Freshest CRL;
- Reason codes of interest.

The certificate type can be determined from the basic constraints extension in the certificate. If the extension is present, it indicates whether the certificate is a CA-certificate or an end-entity certificate. If the extension is absent, the certificate type is considered to be end-entity. This information is required to determine if a CRL, EPRL or CARL can be used to check the certificate for revocation.

If the certificate contains a critical CRL Distribution Point extension, the relying party certificate processing system shall understand this extension and obtain and use the CRL(s) pointed to by the CRL Distribution Point extension for the reason codes of interest in order to determine revocation status of the certificate. Reliance on a full CRL, for instance, would not be sufficient.

If the certificate contains a critical Freshest CRL extension the relying party cannot use the certificate without first retrieving and checking the freshest CRL.

The reason codes of interest are determined by policy and are generally supplied by the application. It is recommended that these should include all reason codes. This information is required to determine which CRLs are sufficient in terms of reason codes.

Note that policy may also dictate whether or not a relying party is expected to check dCRLs for revocation status, when the **freshestCRL** extension is flagged non-critical or is absent from the certificate. Though excluded from this step, the processing of these optional dCRLs is described in step 4).

B.3 Determine CRLs required

The values of the parameters described in B.2 determine the criteria upon which the CRL types required to check revocation status of a given certificate is determined. The determination of CRL types can be done based on the following sets of criteria as described in B.3.1 through B.3.4 below.

- End-entity certificate with critical CRL DP asserted;
- End-entity certificate with no critical CRL DP asserted;
- CA-certificate with critical CRL DP asserted;
- CA-certificate with no critical CRL DP asserted.

Handling of the remaining parameters (critical freshest CRL extension and set of reason codes of interest) is done within each of the subclauses.

Note that in each case, more than one CRL type can satisfy the requirements. Where there is a choice of CRL types, the relying party may select any of the appropriate types to use.

B.3.1 End-entity with critical CRL DP

If the certificate is an end-entity certificate and **cRLDistributionPoints** extension is present in the certificate and flagged critical, the following CRLs shall be obtained:

- A CRL from one of the nominated distribution Point CRLs that covers one or more of the reason codes of interest;
- If all the reason codes of interest are not covered by that CRL, revocation status for the remaining reason codes may be satisfied by any combination of the following CRLs:
  - Additional distribution point CRLs;
  - Additional complete CRLs;
  - Additional complete EPRLs.

If the freshest CRL extension is also present in the certificate and if flagged critical, one or more CRLs shall also be obtained from one or more of the nominated distribution points in that extension, ensuring that freshest revocation information for all reason codes of interest is checked.

B.3.2 End-entity with no critical CRL DP

If the certificate is an end-entity certificate and the **cRLDistributionPoints** extension is absent from the certificate or present and not flagged critical, revocation status for the reason codes of interest may be satisfied by any combination of the following CRLs:

- Distribution point CRLs (if present);
- Complete CRLs;
- Complete EPRLs.

If the freshest CRL extension is also present in the certificate and if flagged critical, one or more CRLs shall also be obtained from one or more of the nominated distribution points in that extension, ensuring that freshest revocation information for all reason codes of interest is checked.

B.3.3 CA with critical CRL DP

If the certificate is a CA and the **cRLDistributionPoints** extension is present in the certificate and flagged critical, the following CRLs/CARLs shall be obtained:

- A CRL or CARL from one of the nominated distribution Points that covers one or more of the reason codes of interest;

- If all the reason codes of interest are not covered by that CRL/CARL, revocation status for the remaining reason codes may be satisfied by any combination of the following CRLs/CARLs:
  - Additional distribution point CRLs/CARLs;
  - Additional complete CRLs;
  - Additional complete CARLs.

If the freshest CRL extension is also present in the certificate and if flagged critical, one or more CRLs/CARLs shall also be obtained from one or more of the nominated distribution points in that extension, ensuring that freshest revocation information for all reason codes of interest is checked.

B.3.4 CA with no critical CRL DP

If the certificate is a CA certificate and the **cRLDistributionPoints** extension is absent from the certificate or present and not flagged critical, revocation status for the reason codes of interest may be satisfied by any combination of the following CRLs:

- Distribution point CRLs/CARLs (if present);
- Complete CRLs;
- Complete CARLs.

If the freshest CRL extension is also present in the certificate and if flagged critical, one or more CRLs/CARLs shall also be obtained from one or more of the nominated distribution points in that extension, ensuring that freshest revocation information for all reason codes of interest is checked.

B.4 Obtain CRLs

If the relying party is retrieving appropriate CRLs from the Directory, these CRLs are obtained from the CRL DP or certificate issuer directory entry by retrieving the appropriate attributes, i.e. one or more of the following attributes:

- Certificate Revocation List;
- Authority Revocation List;
- Delta Revocation List.

B.5 Process CRLs

After considering the parameters discussed in B.2, identifying appropriate CRL types as described in B.3 and retrieving an appropriate set of CRLs as described in B.4, a relying party is ready to process the CRLs. The set of CRLs will contain at least one base CRL and may also contain one or more dCRLs. For each CRL being processed, the relying party shall ensure that the CRL is accurate with respect to its scope. The relying party has already determined that the CRL is appropriate for the scope of the certificate of interest, through the process of B.2 and B.3 above. In addition, validity checks shall be conducted on the CRLs and they shall be checked to determine whether or not the certificate has been revoked. These checks are described in B.5.1 through B.5.4 below.

B.5.1 Validate base CRL scope

As described in B.3, there can be more than one type of CRL that can be used as the base CRL for checking revocation status of a certificate. Depending on the policy of issuing authority with respect to CRL issuance, the relying party may have one or more of the following base CRL types available to them.

- Complete CRL for all entities;
- Complete EPRL;
- Complete CARL;
- Distribution Point Based CRL/EPRL/CARL.

Subclauses B.5.1.1 through B.5.1.4 provide the set of conditions which shall be true in order for a relying party to use a CRL of each type as the base CRL for certificate revocation status checking for reason codes of interest.

Indirect base CRLs are addressed within each of the subclauses.

B.5.1.1 Complete CRL

In order to determine that a CRL is a complete CRL for end-entity and CA-certificates for which the CRL issuer is responsible, for all reason codes of interest, the following shall be true:

– Delta CRL indicator extension shall be absent; and

– Issuing distribution point extension may be present; and

– Either the issuing distribution point extension shall not contain distribution point field or one of the names in the distribution point field shall match the issuer field in the CRL; and

– Issuing distribution point extension shall either not contain any of the following fields or if it contains any of the following fields, none of the fields present shall be set to TRUE: containsUserPublicKeyCerts, containsCACerts, containsUserAttributeCerts, containsAACerts, and/or containsSOAPublicKeyCerts; and

– If the **reasonCodes** field is present in the issuing distribution point extension, the reasons code field shall include all the reasons of interest to the application; and

– Issuing distribution point extension may or may not contain **indirectCRL** field (hence, this field need not be checked).

B.5.1.2 Complete EPRL

In order to determine that a CRL is a complete EPRL for reason codes of interest, all of the following shall be true:

– Delta CRL indicator extension shall be absent; and

– Issuing distribution point extension shall be present; and

- Either the issuing distribution point extension shall not contain distribution point field or one of the names in the distribution point field shall match the issuer field in the CRL; and

- Issuing distribution point extension shall contain **containsUserPublicKeyCerts** field. This field shall be set to **TRUE**; and

– If the **reasonCodes** field is present in the issuing distribution point extension, the reasons code field shall include all the reasons of interest to the application; and

– Issuing distribution point extension may or may not contain **indirectCRL** field (hence, this field need not be checked); and

This CRL may be only used if the relying party has determined the subject certificate to be an end entity certificate. Thus, if the subject certificate contains the **basicConstraints** extension, its value shall be **cA=FALSE**.

B.5.1.3 Complete CARL

In order to determine that a CRL is a complete CARL for reason codes of interest, all of the following conditions shall be true:

– Delta CRL indicator extension shall be absent; and

– Issuing point distribution shall be present; and

– Either the issuing distribution point extension shall not contain distribution point field or one of the names in the distribution point field shall match the issuer field in the CRL; and

- Issuing distribution point shall contain **containsCACerts** field. This field shall be set to **TRUE**; and
- If the **reasonCodes** field is present in the issuing distribution point extension, the reasons code field shall include all the reasons of interest to the application; and
- Issuing distribution point may or may not contain **indirectCRL** field (hence, this field need not be checked); and

This CARL may be only used if the subject certificate is a CA-certificate. Thus, the subject certificate shall contain the **basicConstraints** extension with the value **cA=TRUE**.

B.5.1.4 Distribution point based CRL/EPRL/CARL

In order to determine that a CRL is one of the CRLs indicated by a CRL distribution point extension or freshest CRL Extension in the certificate, all of the following conditions shall be true:

- Either the distribution point field in the CRL's issuing distribution point extension shall be absent (only when not looking for a critical CRL DP), or one of the names in the distribution point field in the CRL distribution point extension or freshest CRL extension of the certificate shall match one of the names in the distribution point field in the issuing distribution point extension of the CRL. Alternatively, one of the names in the **cRLIssuer** field of the certificate's CRL DP or freshest CRL extension can match one of the names in DP of the IDP; and
- Issuing distribution point extension shall either not contain any of the following fields, or if it contains any of the following fields, none of the fields present shall be set to TRUE: containsUserPublicKeyCerts, containsCACerts, containsUserAttributeCerts, containsAACerts, and/or containsSOAPublicKeyCerts, or the field appropriate for the certificate type shall be set to TRUE (See Table B-1 for field type for each certificate type; and
- If the reasons code field is present in the CRL distribution point extension or freshest CRL extension of the certificate, this field shall be either absent from the issuing distribution point extension of the CRL or contain at least one of the reason codes asserted in the CRL distribution point extension of the certificate; and
- If the **cRLIssuer** field is absent from the CRL distribution point extension of the certificate, the CRL shall be signed by the same CA that signed the certificate; and
- If the **cRLIssuer** field is present in the relative extension (CRL distribution point or freshest CRL extension) of the certificate, the CRL shall be signed by the CRL Issuer identified in the CRL distribution point extension or freshest CRL extension of the certificate and the CRL shall contain the **indirectCRL** field in the issuing distribution point extension.

Note: When testing the reasons and cRLIssuer field for presence, the test succeeds only if the field is present in the same DistributionPoint of the CRL DP or freshest CRL extension for which there is a name match in the distribution point field of the IDP extension in the corresponding CRL.

TABLE B-1: Certificate Type and Issuing Distribution Point Field

Certificate Type	Issuing Distribution Point Field
End Entity (public key)	containsUserPublicKeyCerts
CA	containsCACerts
End Entity (attribute)	containsUserAttributeCerts

AA	containsAACerts
SOA	containsSOAPublicKeyCerts

B.5.2 Validate delta CRL scope

The relying party may also be checking dCRLs, either because required to through a critical **freshestCRL** extension in the certificate or CRL, or because the policy under which the relying party is operating requires dCRL checking.

A relying party can always be sure that it has the appropriate CRL information for a certificate if all of the following conditions are met:

– The base CRL the relying party is using is appropriate for the certificate (in terms of the scope); and

– The delta CRL the relying party is using is appropriate for the certificate (in terms of the scope); and

– The base CRL was issued at the time or later than the base CRL referenced by the dCRL.

In order to determine that the dCRL is appropriate for the certificate, all of the following conditions shall be true:

– Delta CRL indicator extension shall be present; and

– The dCRL shall be issued after the base CRL. One way to ensure this is to check that the CRL number in the **crlNumber** extension of the dCRL is greater than the CRL number in the **crlNumber** extension of the base CRL the relying party is using and the **cRLStreamIdentifier** fields in the base and the dCRL match. This approach may require additional logic to account for number wrapping. Another way is to compare the **thisUpdate** fields in the base and dCRLs the relying party has; and

– The base CRL the relying party is using shall be the one the dCRL is issued for or a later one. One way to ensure this is to check that the CRL number in the **deltaCRLIndicator** extension of the dCRL is less than or equal to the CRL number in the **crlNumber** extension of the base CRL the relying party is using and the **cRLStreamIdentifier** fields in the base and the dCRL match. This approach may require additional logic to account for number wrapping. Another way is to compare the **thisUpdate** fields of the base CRL the relying party has and the base CRL pointed to by the dCRL. Yet another way is to compare the **thisUpdate** field in the base CRL the relying party has and the **baseUpdateTime** extension in the dCRL the relying party has; and

  NOTE – A relying party can always construct a base CRL by applying a dCRL to a base CRL as long as the above two rules are satisfied using the **crlNumber** and **cRLStreamIdentifier** checks. In that case, the new base CRL's **crlNumber** extension and **thisUpdate** field are those of the dCRL. The relying party does not know the **nextUpdate** field of the new base CRL and does not need to know for the purpose of associating it with another dCRL.

– If the dCRL contains an Issuing Distribution Point extension, then the scope of the issuing distribution point shall be consistent with the certificate as described in B.5.1.4 above; and

– If the dCRL does not contain any of the following extensions: **streamIdentifier** and **issuingDistributionPoint**, it shall be used only in conjunction with a full and complete base CRL.

B.5.3 Validity and currency checks on the base CRL

In order to verify that a base CRL is accurate and has not been modified since its issuance, all of the following conditions shall be satisfied:

- The relying party shall be able to obtain the public key of the issuer identified in the CRL using authenticated means (See Section B.6 for additional requirements to obtain the authenticated public key); and
- The signature on the base CRL shall be verified using this authenticated public key; and
- If the **nextUpdate** field is present, the current time should be prior to the **nextUpdate** field; and
- The issuer name in the CRL shall match the issuer name in the certificate that is being checked for revocation, unless the CRL is retrieved from the CRL DP in the certificate and the CRL DP extension contains the CRL issuer component. In that case, one of the names in CRL issuer component in the CRL DP extension shall match issuer name in the CRL.

B.5.4 Validity and checks on the delta CRL

In order to verify that a dCRL is accurate and has not been modified since its issuance, all of the following conditions shall be satisfied:

- The relying party shall be able to obtain the public key of the issuer identified in the CRL using authenticated means (See Section B.6 for additional requirements to obtain the authenticated public key), and
- The signature on the dCRL shall be verified using this authenticated public key; and
- If the **nextUpdate** field is present, the current time should be prior to the **nextUpdate** field; and
- The issuer name in the dCRL shall match the issuer name in the certificate which is being checked for revocation, unless the Delta CRL is retrieved from the CRL DP in the certificate and the CRL DP extension contains the CRL issuer component. In that case, one of the names in CRL issuer component in the CRL DP extension shall match issuer name in the CRL.

Annex G

Delete the existing text of clause G.2.

Renumber existing clause G.3 to be the new G.2

Add the following as a new G.3

G.3 Use of Name Constraints Extension

G.3.1 Examples of Certificate Format with Name Constraints Extension

The CAs can put various restrictions to the subject names (in the **subject** filed or **subjectAltName** extension) of the certificates that they issue and the subsequent certificates in the certification path, by inclusion of the Name Constraints extension to their CA-certificates. This section describes the examples of certificate format with Name Constraints extension.

Making the examples simple, required name forms (**requiredNameForms**) of Name Constraints extension in these examples indicates rfc822 name (**rfc822Name**) and DN (**directoryName**) only.

G.3.1.1 Examples of *permittedSubtrees*

(1-1) If the CA-certificate contains the following Name Constraints extension, for all subsequent certificates in the certification path, each subject name (in the **subject** field or **subjectAltName** extension) in DN name form, if exists, shall be equal to or subordinate to the Acme Inc. in U.S. (i.e. {C=US, O=Acme Inc}).

nameConstraints extension

permittedSubtrees	**excludedSubtrees**	**requiredNameForms**
{{**base**(**directoryName**) {C=US, O=Acme Inc}}}	(void)	(void)

(1-2) If the CA-certificate contains the following Name Constraints extension, for all subsequent certificates in the certification path, each subject name (in the **subject** field or **subjectAltName** extension) in DN name form, if exists, shall be equal to or immediately subordinate to the Acme Inc. in U.S. (i.e. {C=US, O=Acme Inc}).

nameConstraints extension		
permittedSubtrees	**excludedSubtrees**	**requiredNameForms**
{{**base**(**directoryName**) {C=US, O=Acme Inc}, **maximum 1**}}	(void)	(void)

(1-3) If the CA-certificate contains the following Name Constraints extension, for all subsequent certificates in the certification path, each subject name (in the **subject** field or **subjectAltName** extension) in DN name form, if exists, shall be subordinate to the Acme Inc. in U.S. (i.e. {C=US, O=Acme Inc}).

nameConstraints extension		
permittedSubtrees	**excludedSubtrees**	**requiredNameForms**
{{**base**(**directoryName**) {C=US, O=Acme Inc}, **minimum 1**}}	(void)	(void)

(1-4) If the CA-certificate contains the following Name Constraints extension, for all subsequent certificates in the certification path, each subject name (in the **subject** field or **subjectAltName** extension) in DN name form, if exists, shall be equal to or subordinate to the Acme Inc. in U.S. (i.e. {C=US, O=Acme Inc}), or equal to or subordinate to the Acme Ltd. in U.K. (i.e. {C=UK, O=Acme Ltd}).

nameConstraints extension		
permittedSubtrees	**excludedSubtrees**	**requiredNameForms**
{{**base**(**directoryName**) {C=US, O=Acme Inc}}, {**base**(**directoryName**) {C=UK, O=Acme Ltd}}}	(void)	(void)

G.3.1.2 Examples of *excludedSubtrees*

(2-1) If the CA-certificate contains the following Name Constraints extension, for all subsequent certificates in the certification path, each subject name (in the **subject** field or **subjectAltName** extension) in DN name form, if exists, shall not be equal to nor subordinate to the Acme Corp. in Canada. (i.e. {C=CA, O=Acme Corp}).

nameConstraints extension		
permittedSubtrees	**excludedSubtrees**	**requiredNameForms**
(void)	{{**base**(**directoryName**) {C=CA, O=Acme Corp}}}	(void)

(2-2) If the CA-certificate contains the following Name Constraints extension, for all subsequent certificates in the certification path, each subject name (in the **subject** field or **subjectAltName** extension) in DN name form, if exists, shall not be subordinate to each immediately subordinate of the Acme Corp. in Canada (i.e. {C=CA, O=Acme Corp}).

nameConstraints extension		
permittedSubtrees	**excludedSubtrees**	**requiredNameForms**
(void)	{{**base**(**directoryName**) {C=CA, O=Acme Corp}, **minimum 2**}}	(void)

(2-3) If the CA-certificate contains the following Name Constraints extension, for all subsequent certificates in the certification path, each subject name (in the **subject** field or **subjectAltName** extension) in DN name form, if exists, shall not be equal to the Acme Corp. in Canada (i.e. {C=CA, O=Acme Corp}).

nameConstraints extension		
permittedSubtrees	**excludedSubtrees**	**requiredNameForms**
(void)	{{**base**(**directoryName**) {C=CA, O=Acme Corp}, **maximum 0**}}	(void)

(2-4) If the CA-certificate contains the following Name Constraints extension, for all subsequent certificates in the certification path, each subject name (in the **subject** field or **subjectAltName** extension) in DN name form, if exists, shall not be equal to nor subordinate to the Acme Corp. in Canada (i.e. {C=CA, O=Acme Corp}), nor equal to nor subordinate to the Asia Acme in Japan (i.e. {C=JP, O=Asia Acme}).

nameConstraints extension		
permittedSubtrees	**excludedSubtrees**	**requiredNameForms**
(void)	{{**base**(**directoryName**) {C=CA, O=Acme Corp}}, {**base**(**directoryName**) {C=JP, O=Asia Acme}}}	(void)

G.3.1.3 Examples of permittedSubtrees and excludedSubtrees

(3-1) If the CA-certificate contains the following Name Constraints extension, for all subsequent certificates in the certification path, each subject name (in the **subject** field or **subjectAltName** extension) in DN name form, if exists, shall be equal to or subordinate to the Acme Inc. in U.S. (i.e. {C=US, O=Acme Inc}) except the R&D organization unit of Acme Inc. and the R&D organization's subordinates.

nameConstraints extension		
permittedSubtrees	**excludedSubtrees**	**requiredNameForms**
{{**base**(**directoryName**) {C=US, O=Acme Inc}}}	{{**base**(**directoryName**) {C=US, O=Acme Inc, OU=R&D}}}	(void)

(3-2) If the CA-certificate contains the following Name Constraints extension, for all subsequent certificates in the certification path, each subject name (in the **subject** field or

subjectAltName extension) in DN name form, if exists, shall be equal to one of immediately subordinates to the Acme Inc. in U.S. (i.e. {C=US, O=Acme Inc}) except the Purchasing organization unit (i.e. {C=US, O=Acme Inc, OU=Purchasing}).

nameConstraints extension		
permittedSubtrees	**excludedSubtrees**	**requiredNameForms**
{{**base**(**directoryName**) {C=US, O=Acme Inc}, **minimum 1**, **maximum 1**}}	{{**base**(**directoryName**) {C=US, O=Acme Inc, OU=Purchasing}}}	(void)

G.3.1.4 Examples of permittedSubtrees and excludedSubtrees with requiredNameForms

(4-1) If the CA-certificate contains the following Name Constraints extension, for all subsequent certificates in the certification path, at least one of subject names (in the **subject** field or **subjectAltName** extension) of certificate shall be in the DN name form. However, each subject name is not constrained by any name spaces.

nameConstraints extension			
permittedSubtrees	**excludedSubtrees**	**requiredNameForms**	
		rfc822 name	DN
(void)	(void)	OFF	ON

(4-2) If the CA-certificate contains the following Name Constraints extension, for all subsequent certificates in the certification path, at least one of subject names (in the **subject** field or **subjectAltName** extension) shall be in the DN name form. Moreover, each subject name in DN name form shall satisfy the name spaces constrained by the **permittedSubtrees** and **excludedSubtrees**.

nameConstraints extension			
permittedSubtrees	**excludedSubtrees**	**requiredNameForms**	
		rfc822 name	DN
{{**base**(**directoryName**) {C=JP, O=Asia Acme}}}	{{**base**(**directoryName**) {C=JP, O=Asia Acme, OU=Marketing}}}	OFF	ON

(4-3) If the CA-certificate contains the following Name Constraints extension, for all subsequent certificates in the certification path, each subject name (in the **subject** field or **subjectAltName** extension) in DN name form, if exists, shall satisfy the name spaces constrained by the **permittedSubtrees** and **excludedSubtrees**.

nameConstraints extension			
permittedSubtrees	**excludedSubtrees**	**requiredNameForms**	
		rfc822 name	DN
{{**base**(**directoryName**) {C=JP, O=Asia Acme}}}	{{**base**(**directoryName**) {C=JP, O=Asia Acme, OU=Marketing}}}	OFF	OFF

Note: The above example of CA-certificate is compatible with the following CA-certificate with Name Constraints extension without **requiredNameForms** element.

nameConstraints extension		
permittedSubtrees	**excludedSubtrees**	**requiredNameForms**
{{**base**(**directoryName**) {C=JP, O=Asia Acme}}}	{{**base**(**directoryName**) {C=JP, O=Asia Acme, OU=Marketing}}}	(void)

(4-4) If the CA-certificate contains the following Name Constraints extension, for all subsequent certificates in the certification path, each subject name (in the **subject** field or **subjectAltName** extension) in DN name form, if exists, shall satisfy the name spaces constrained by the **permittedSubtrees** and **excludedSubtrees**. Moreover, at least one **subjectAltName** in **rfc822Name** name form shall be present, though its name is not constrained by any name spaces.

nameConstraints extension			
permittedSubtrees	**excludedSubtrees**	**requiredNameForms**	
		rfc822 name	DN
{{**base**(**directoryName**) {C=JP, O=Asia Acme}}}	{{**base**(**directoryName**) {C=JP, O=Asia Acme, OU=Marketing}}}	ON	OFF

(4-5) If the CA-certificate contains the following Name Constraints extension, for all subsequent certificates in the certification path, at least one of subject names (in the **subject** field or **subjectAltName** extension) of certificate shall be in the DN name form or in the rfc822 name form.
Each subject name in DN name form, if exists, shall satisfy the name spaces constrained by the **permittedSubtrees** and **excludedSubtrees**.
Each subject name in **rfc822Name** name form is not constrained by any name spaces.

nameConstraints extension			
permittedSubtrees	**ExcludedSubtrees**	**requiredNameForms**	
		rfc822 name	DN
{{**base**(**directoryName**) {C=JP, O=Asia Acme}}}	{{**base**(**directoryName**) {C=JP, O=Asia Acme, OU=Marketing}}}	ON	ON

G.3.2 Examples of Certificate Handling with Name Constraint Extension

This section describes the examples of how subject name (in the **subject** field or **subjectAltName** extension) are validated in the certificate processing with the path processing state variables, namely *permitted-subtrees, excluded-subtrees and required-name-forms.*

Making the examples simple, the path processing state variable *required-name-forms* in these examples indicates rfc822 name (**rfc822Name**), DN (**directoryName**) and URI (**uniformResourceIdentifier**) only.

G.3.2.1 Name Spaces Constraints by *permitted-subtrees* in DN Name Form

In this case, each subject name (in the **subject** field or **subjectAltName** extension) in DN name form appeared in the certificate in question shall satisfy the constraint by path processing state variable *permitted-subtrees*.

(1-1) One permitted subtree for DN is present and DN is required in *required-name-forms*.

Path Processing State Variables				
permitted-subtrees	*excluded-subtrees*	*required-name-forms*		
		rfc822	DN	URI
{{**base**(**directoryName**) {C=US, O=Acme Inc}}}	NONE	OFF	ON	OFF

Acceptable Certificate Examples

1	**subject** = {C=US, O=Acme Inc, OU=Purchasing}

2	**subject** = {}
	subjectAltName(**directoryName**) = {C=US, O=Acme Inc, OU=Purchasing}

3	**subject** = {C=US, O=Acme Inc, OU=Purchasing} **subjectAltName** (**rfc822Name**) = manager@purchasing.acme.com

4	**subject** = {}
	subjectAltName (**directoryName**) = {C=US, O=Acme Inc, OU=Purchasing}
	subjectAltName (**rfc822Name**) = manager@purchasing.acme.com

5	**subject** = {C=US, O=Acme Inc, OU=Purchasing}
	subjectAltName (**directoryName**) = {C=US, O=Acme Inc, OU=Accounting}

Unacceptable Certificate Examples

1	**subject** = {C=US, O=*Acme Ltd*,OU=Purchasing}
2	**subject** = {} **subjectAltName** (**rfc822Name**) = manager@purchasing.acme.com Note: *DN missing*

3	**subject** = {C=US, O=Acme Inc, OU=Purchasing}
	subjectAltName (**directoryName**) = {C=US, O=*Acme Ltd*, OU=Purchasing}

4	**subject** = {C=US, O=*Acme Ltd*, OU=Purchasing}
	subjectAltName (**directoryName**) = {C=US, O=Acme Inc, OU=Purchasing}

5	**subject** = {C=US, O=*Acme Ltd*, OU=Purchasing}
	subjectAltName (**directoryName**) = {C=US, O=*Acme Ltd*, OU=Accounting}

(1-2) Two permitted subtrees for DN are present and DN is required in *required-name-forms*

Path Processing State Variables				
permitted-subtrees	*excluded-subtrees*	*required-name-forms*		
		rfc822	DN	URI

{{**base**(**directoryName**) {C=US, O=Acme Inc}}, {**base**(**directoryName**) {C=US, O=Acme Ltd}}}	NONE	OFF	ON	OFF

Acceptable Certificate Examples

1	**subject** = {C=US, O=Acme Ltd, OU=Purchasing}

2	**subject** = {}
	subjectAltName(**directoryName**) = {C=US, O=Acme Ltd, OU=Purchasing}

3	**subject** = {C=US, O=Acme Ltd, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme-ltd.com

4	**subject** = {}
	subjectAltName(**directoryName**) = {C=US, O=Acme Inc, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=Acme Ltd, OU=Purchasing}
	subjectAltName(**rfc822Name**) = manager@purchasing.acme.com

5	**subject** = {C=US, O=Acme Ltd, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=Acme Ltd, OU= Accounting}

Unacceptable Certificate Examples

1	**subject** = {C=US, O=*Acme International*, OU=Accounting}
2	**subject** = {} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme.com Note: *DN missing*

3	**subject** = {C=US, O=Acme Inc, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=*Acme International*, OU=Accounting}

4	**subject** = {C=US, O=*Acme International*, OU=Accounting}
	subjectAltName(**directoryName**) = {C=US, O=Acme Inc, OU=Purchasing}

5	**subject** = {C=US, O=*Acme International*, OU=Accounting}
	subjectAltName(**directoryName**) = {C=US, O=*Acme Corp*, OU=Accounting}

6	**subject** = {}
	subjectAltName(**directoryName**) = {C=US, O=Acme Inc, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=*Acme International*, OU=Accounting}
	subjectAltName(**rfc822Name**) = manager@purchasing.acme.com

(1-3) One permitted subtree for DN is present and *required-name-forms* is empty.

Path Processing State Variables				
permitted-subtrees	*excluded-subtrees*	*required-name-forms*		
		rfc822	DN	URI

{{**base**(**directoryName**) {C=US, O=Acme Inc}}}	NONE	Empty

Acceptable Certificate Examples

1	**subject** = {C=US, O=Acme Inc, OU=Purchasing}

2	**subject** = {}
	subjectAltName(**directoryName**) = {C=US, O=Acme Inc, OU=Purchasing}

3	**subject** = {C=US, O=Acme Inc, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme.com

4	**subject** = {}
	subjectAltName(**directoryName**) = {C=US, O=Acme Inc, OU=Purchasing}
	subjectAltName(**rfc822Name**) = manager@purchasing.acme.com

5	**subject** = {C=US, O=Acme Inc, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=Acme Inc, OU=Accounting}

6	**subject** = {} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme.com

Unacceptable Certificate Examples

1	**subject** = {C=US, O=*Acme Ltd*,OU=Purchasing}

2	**subject** = {C=US, O=Acme Inc, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=*Acme Ltd*, OU=Accounting}

3	**subject** = {C=US, O=*Acme Ltd*, OU=Accounting}
	subjectAltName(**directoryName**) = {C=US, O=Acme Inc, OU=Purchasing}

4	**subject** = {C=US, O=*Acme Ltd*, OU=Accounting}
	subjectAltName(**directoryName**) = {C=US, O=*Acme Ltd*, OU=Purchasing}

G.3.2.2 Name Spaces Constraints by *excluded-subtrees* in DN Name Form

In this case, each subject name (in the **subject** field or **subjectAltName** extension) in DN name form appeared in the certificate in question shall satisfy the constraint by path processing state variable *excluded-subtrees*.

(2-1) One excluded subtree for DN is present and DN is required in *required-name-forms*.

Path Processing State Variables				
permitted-subtrees	*excluded-subtrees*	*required-name-forms*		
		rfc822	DN	URI
NONE	{{**base**(**directoryName**) {C=US, O=Acme Ltd}}}	OFF	ON	OFF

Acceptable Certificate Examples

1	**subject** = {C=US, O=Acme Inc, OU=Purchasing}

2	**subject** = {}
	subjectAltName(**directoryName**) = {C=US, O=Acme Inc, OU=Purchasing}

3	**subject** = {C=US, O=Acme Inc, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme.com

4	**subject** = {}
	subjectAltName(**directoryName**) = {C=US, O=Acme Inc, OU=Purchasing}
	subjectAltName(**rfc822Name**) = manager@purchasing.acme.com

5	**subject** = {C=US, O=Acme Inc, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=Acme Inc, OU=Accounting}

Unacceptable Certificate Examples

1	**subject** = {C=US, O=*Acme Ltd*,OU=Purchasing}
2	**subject** = {} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme.com Note: *DN missing*

3	**subject** = {C=US, O=Acme Inc, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=*Acme Ltd*, OU=Accounting}

(2-2) Two excluded subtrees for DN are present and DN is required in *required-name-forms*.

Path Processing State Variables				
permitted-subtrees	*excluded-subtrees*	*required-name-forms*		
		rfc822	DN	URI
NONE	{{**base**(**directoryName**) {C=US, O=Acme Inc}}, {**base**(**directoryName**) {C=US, O=Acme Ltd}}}	OFF	ON	OFF

Acceptable Certificate Examples

1	**subject** = {C=US, O=Acme International, OU=Purchasing}

2	**subject** = {}
	subjectAltName(**directoryName**) = {C=US, O=Acme International, OU=Purchasing}

3	**subject** = {C=US, O=Acme International, OU=Purchasing} **subjectAltName**(**rfc822Name**) = purchasing@acme-international.com
4	**subject** = {}

	subjectAltName(**directoryName**) = {C=US, O=Acme International, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=Acme N.Y, OU=Purchasing}
	subjectAltName(**rfc822Name**) = purchasing@acme-international.com

Unacceptable Certificate Examples

1	**subject** = {C=US, O=*Acme Inc*,OU=Purchasing}
2	**subject** = {C=US, O=*Acme Ltd*,OU=Purchasing}
3	**subject** = {} **subjectAltName**(**rfc822Name**) = purchasing@acme-international.com Note: *DN missing*

4	**subject** = {C=US, O=*Acme Inc*, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=Acme International, OU=Accounting}

5	**subject** = {}
	subjectAltName(**directoryName**) = {C=US, O=*Acme Inc*, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=Acme International, OU=Purchasing}
	subjectAltName(**rfc822Name**) = purchasing@acme-international.com

(2-3) One excluded subtree for DN is present and *required-name-forms* is empty.

Path Processing State Variables				
permitted-subtrees	*excluded-subtrees*	*required-name-forms*		
		rfc822	DN	URI
NONE	{{**base**(**directoryName**) {C=US, O=Acme Inc}}}	empty		

Acceptable Certificate Examples

1	**subject** = {C=US, O=Acme Ltd, OU=Purchasing}

2	**subject** = {}
	subjectAltName(**directoryName**) = {C=US, O=Acme Ltd, OU=Purchasing}

3	**subject** = {C=US, O=Acme Ltd, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme-ltd.com

4	**subject** = {}
	subjectAltName(**directoryName**) = {C=US, O=Acme Ltd, OU=Purchasing}
	subjectAltName(**rfc822Name**) = manager@purchasing.acme-ltd.com

5	**subject** = {C=US, O=Acme Ltd, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=Acme Ltd, OU=Accounting}

6	**subject** = {}

	subjectAltName(**rfc822Name**) = manager@purchasing.acme-ltd.com

Unacceptable Certificate Examples

1	**subject** = {C=US, O=*Acme Inc*, OU=Purchasing}

2	**subject** = {C=US, O=*Acme Inc*, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=Acme Ltd, OU=Accounting}

G.3.2.3 Name Spaces Constraints only by required-name-forms

(3-1) DN is required in *required-name-forms*.

Path Processing State Variables				
permitted-subtrees	*excluded-subtrees*	*required-name-forms*		
		rfc822	DN	URI
NONE	NONE	OFF	ON	OFF

Acceptable Certificate Examples

1	**subject** = {C=US, O=Acme Ltd, OU=Purchasing}

2	**subject** = {}
	subjectAltName(**directoryName**) = {C=JP, O=Acme Inc, OU=Purchasing}

3	**subject** = {C=JP, O=Acme Ltd, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme-ltd.com

4	**subject** = {}
	subjectAltName(**directoryName**) = {C=US, O=Acme Ltd, OU=Purchasing}
	subjectAltName(**rfc822Name**) = manager@purchasing.acme-ltd.com

5	**subject** = {C=JP, O=Acme Ltd, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=Acme Ltd, OU=Accounting}

Unacceptable Certificate Examples

1	**subject** = {} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme-ltd.com Note: *DN missing*
2	**subject** = {} **subjectAltName**(**uniformResourceIdentifier**) = http://purchasing.www.acme-ltd.com Note: *DN missing*

(3-2) DN or **rfc822Name** are required in *required-name-forms*.

Path Processing State Variables				
permitted-subtrees	*excluded-subtrees*	*required-name-forms*		
		rfc822	DN	URI
NONE	NONE	ON	ON	OFF

Acceptable Certificate Examples

1	**subject** = {C=US, O=Acme Ltd, OU=Purchasing}

2	**subject** = {}
	subjectAltName(**directoryName**) = {C=JP, O=Acme Inc, OU=Purchasing}

3	**subject** = {} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme-ltd.com

4	**subject** = {}
	subjectAltName(**directoryName**) = {C=US, O=Acme Ltd, OU=Purchasing}
	subjectAltName(**rfc822Name**) = manager@purchasing.acme-ltd.com

5	**subject** = {} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme-ltd.com **subjectAltName**(**rfc822Name**) = purchasing@acme-ltd.com

Unacceptable Certificate Examples

1	**subject** = {} **subjectAltName**(**uniformResourceIdentifier**) = http://purchasing.www.acme-ltd.com Note: *DN and rfc822 missing*
2	**subject** = {} **subjectAltName**(**dNSName**) = www.acme-ltd.com Note: *DN and rfc822 missing*

G.3.2.4 Name Spaces Constraints by *permitted-subtrees* in Multiple Name Forms

In this case, each subject name (in the **subject** field or **subjectAltName** extension) in DN name form or rfc822 name form appeared in the certificate in question shall satisfy the constraint by path processing state variable *permitted-subtrees*.

(4-1) One permitted subtree for DN and another permitted subtree **rfc822Name** are present. In addition, DN is required in *required-name-forms*.

Path Processing State Variables				
permitted-subtrees	*excluded-subtrees*	*required-name-forms*		
		rfc822	DN	URI
{{**base**(**directoryName**) {C=US, O=Acme Inc}},	NONE	OFF	ON	OFF

{**base**(**rfc822Name**) .acme.com}}				

Acceptable Certificate Examples

1	**subject** = {C=US, O=Acme Inc, OU=Purchasing}
2	**subject** = {C=US, O=Acme Inc, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme.com

3	**subject** = {C=US, O=Acme Inc, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=Acme Inc, OU=Accounting}

4	**subject** = {C=US, O=Acme Inc, OU=Purchasing} **subjectAltName**(**uniformResourceIdentifier**) = http://purchasing.www.acme-inc.com

Unacceptable Certificate Examples

1	**subject** = {C=US, O=*Acme Ltd*, OU=Purchasing}
2	**subject** = {} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme.com Note: *DN missing*
3	**subject** = {C=US, O=*Acme Ltd*, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme.com
4	**subject** = {C=US, O=Acme Inc, OU=Purchasing} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme-inc.com*
5	**subject** = {C=US, O=*Acme Ltd*, OU=Purchasing} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme-ltd.com*
6	**subject** = {} **subjectAltName**(**uniformResourceIdentifier**) = http://purchasing.www.acme-inc.com Note: *DN missing*

(4-2) One permitted subtree for DN and another permitted subtree **rfc822Name** are present. In addition, at least one of DN or **rfc822Name** is required in *required-name-forms*.

Path Processing State Variables				
permitted-subtrees	*excluded-subtrees*	*required-name-forms*		
		rfc822	DN	URI
{{**base**(**directoryName**) {C=US, O=Acme Inc}}, {**base**(**rfc822Name**) .acme.com}}	NONE	ON	ON	OFF

Acceptable Certificate Examples

1	**subject** = {C=US, O=Acme Inc, OU=Purchasing}
2	**subject** = {} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme.com
3	**subject** = {C=US, O=Acme Inc, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme.com

4	**subject** = {C=US, O=Acme Inc, OU=Purchasing}
	subjectAltName(**directoryName**) = { C=US, O=Acme Inc, OU=Accounting}

5	**subject** = {C=US, O=Acme Inc, OU=Purchasing} **subjectAltName**(**uniformResourceIdentifier**) = http://purchasing.www.acme-inc.com

Unacceptable Certificate Examples

1	**subject** = {C=US, O=*Acme Ltd*, OU=Purchasing}
2	**subject** = {} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme-inc.com*
3	**subject** = {C=US, O=*Acme Ltd*, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme.com
4	**subject** = {C=US, O=Acme Inc, OU=Purchasing} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme-inc.com*
5	**subject** = {C=US, O=*Acme Ltd*, OU=Purchasing} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme-ltd.com*
6	**subject** = {} **subjectAltName**(**uniformResourceIdentifier**) = http://purchasing.www.acme-inc.com Note: *DN and rfc822 missing*

(4-3) One permitted subtree for DN and another permitted subtree **rfc822Name** are present. No name forms are required in *required-name-forms*.

Path Processing State Variables				
permitted-subtrees	*excluded-subtrees*	*required-name-forms*		
		rfc822	DN	URI
{{**base**(**directoryName**) {C=US, O=Acme Inc}}, {**base**(**rfc822Name**) .acme.com}}	NONE	empty		

Acceptable Certificate Examples

1	**subject** = {C=US, O=Acme Inc, OU=Purchasing}
2	**subject** = {} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme.com
3	**subject** = {C=US, O=Acme Inc, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme.com
4	**subject** = {} **subjectAltName**(**uniformResourceIdentifier**) = http://purchasing.www.acme.com
5	**subject** = {C=US, O=Acme Inc, OU=Purchasing} **subjectAltName**(**uniformResourceIdentifier**) = http://purchasing.www.acme.com

Unacceptable Certificate Examples

1	**subject** = {C=US, O=*Acme Ltd*, OU=Purchasing}
2	**subject** = {} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme-inc.com*
3	**subject** = {C=US, O=*Acme Ltd*, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme.com
4	**subject** = {C=US, O=Acme Inc, OU=Purchasing} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme-inc.com*
5	**subject** = {C=US, O=*Acme Ltd*, OU=Purchasing} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme-inc.com*

G.3.2.5 Name Spaces Constraints by *excluded-subtrees* in Multiple Name Forms

In this case, each subject name (in the **subject** field or **subjectAltName** extension) in DN name form or rfc822 name form appeared in the certificate in question shall satisfy the constraint by path processing state variable *excluded-subtrees*.

(5-1) One excluded subtree for DN and another excluded subtree **rfc822Name** are present. In addition, DN is required in *required-name-forms*.

Path Processing State Variables				
permitted-subtrees	*excluded-subtrees*	*required-name-forms*		
		rfc822	DN	URI
NONE	{{**base**(**directoryName**) {C=US, O=Acme Inc}}, {**base**(**rfc822Name**) .acme.com}}	OFF	ON	OFF

Acceptable Certificate Examples

1	**subject** = {C=US, O=Acme Ltd, OU=Purchasing}
2	**subject** = {C=US, O=Acme Ltd, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme-ltd.com

3	**subject** = {C=US, O=Acme Ltd, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=Acme Ltd, OU=Accounting}

4	**subject** = {C=US, O=Acme Ltd, OU=Purchasing} **subjectAltName**(**uniformResourceIdentifier**) = http://purchasing.www.acme-ltd.com

Unacceptable Certificate Examples

1	**subject** = {C=US, O=*Acme Inc*, OU=Purchasing}
2	**subject** = {} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme.com Note: *DN missing*
3	**subject** = {C=US, O=Acme Ltd, OU=Purchasing} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme.com*
4	**subject** = {C=US, O=*Acme Inc*, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme-inc.com
5	**subject** = {C=US, O=*Acme Inc*, OU=Purchasing} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme.com*

6	**subject** = {C=US, O=Acme Ltd, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=*Acme Inc*, OU=Accounting}

7	**subject** = {} **subjectAltName**(**uniformResourceIdentifier**) = http://purchasing.www.acme-inc.com Note: *DN missing*

(5-2) One excluded subtree for DN and another excluded subtree **rfc822Name** are present. In addition, at least one of DN or **rfc822Name** is required in *required-name-forms*.

Path Processing State Variables				
permitted-subtrees	*excluded-subtrees*	*required-name-forms*		
		rfc822	DN	URI
NONE	{{**base**(**directoryName**) {C=US, O=Acme Inc}}, {**base**(**rfc822Name**) .acme.com}}	ON	ON	OFF

Acceptable Certificate Examples

1	**subject** = {C=US, O=Acme Ltd, OU=Purchasing}
2	**subject** = {}

	subjectAltName(**rfc822Name**) = manager@purchasing.acme.org
3	**subject** = {C=US, O=Acme Ltd, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme-ltd.com

4	**subject** = {C=US, O=Acme Ltd, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=Acme Ltd, OU=Accounting}

5	**subject** = {C=US, O=Acme Ltd, OU=Purchasing} **subjectAltName**(**uniformResourceIdentifier**) = http://purchasing.www.acme-ltd.com

Unacceptable Certificate Examples

1	**subject** = {C=US, O=*Acme Inc*, OU=Purchasing}
2	**subject** = {} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme.com*
3	**subject** = {C=US, O=Acme Ltd, OU=Purchasing} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme.com*
4	**subject** = {C=US, O=*Acme Inc*, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme-inc.com
5	**subject** = {C=US, O=*Acme Inc*, OU=Purchasing} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme.com*

6	**subject** = {C=US, O=Acme Ltd, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=*Acme Inc*, OU=Accounting}

7	**subject** = {} **subjectAltName**(**uniformResourceIdentifier**) = http://purchasing.www.acme-inc.com Note: *DN and rfc822 missing*

(5-3) One excluded subtree for DN and another excluded subtree **rfc822Name** are present. No name forms are required in *required-name-forms*.

Path Processing State Variables				
permitted-subtrees	*excluded-subtrees*	*required-name-forms*		
		rfc822	DN	URI
NONE	{{**base**(**directoryName**) {C=US, O=Acme Inc}}, {**base**(**rfc822Name**) .acme.com}}	Empty		

Acceptable Certificate Examples

1	**subject** = {C=US, O=Acme Ltd, OU=Purchasing}
2	**subject** = {}

	subjectAltName(**rfc822Name**) = manager@purchasing.acme-ltd.com
3	**subject** = {C=US, O=Acme Ltd, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme-ltd.com
4	**subject** = {} **subjectAltName**(**uniformResourceIdentifier**) = http://purchasing.www.acme-inc.com
5	**subject** = {C=US, O=Acme Ltd, OU=Purchasing} **subjectAltName**(**uniformResourceIdentifier**) = http://purchasing.www.acme-ltd.com

Unacceptable Certificate Examples

1	**subject** = {C=US, O=*Acme Inc*, OU=Purchasing}
2	**subject** = {} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme.com*
3	**subject** = {C=US, O=Acme Ltd, OU=Purchasing} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme.com*
4	**subject** = {C=US, O=*Acme Inc*, OU=Purchasing} **subjectAltName**(**rfc822Name**) = manager@purchasing.acme-inc.com
5	**subject** = {C=US, O=*Acme Inc*, OU=Purchasing} **subjectAltName**(**rfc822Name**) = *manager@purchasing.acme.com*

6	**subject** = {C=US, O=Acme Ltd, OU=Purchasing}
	subjectAltName(**directoryName**) = {C=US, O=*Acme Inc*, OU=Accounting}

Add the following new informative Annex J that provides guidance on configuring variables that are inputs to the path validation process.

Annex J

Annex J -- Guidance on Determining for Which Policies a Certification Path is Valid

(This annex does not form an integral part of this Recommendation | International Standard)

The purpose of this annex to provide guidance for PKI enabled applications with respect to control of the certificate policy related processing of certificate path validation. Control of certificate policy related processing by the PKI through the contents of certificates is described in the Certificate Path Processing Procedure section of the standard.

This annex addresses initialization of two of the policy related inputs in the path processing procedure: *initial-policy-set* and *initial-explicit-policy*. In addition to these, the *initial-policy-mapping-inhibit* and *initial-inhibit-any-policy* inputs to the procedure, which can also be initialized by the user, impact the processing of policy related information during path processing, however these are outside the scope of this Annex. Setting the *initial-policy-mapping-inhibit* to **TRUE** prevents policy mappings from being used in successful path validations. Setting the *initial-inhibit-any-policy* to **TRUE** prevents the special OID for **anyPolicy**, if present in a certificate, from being an acceptable match for a specific policy OID.

The terms “user” in this annex can be used to mean a “human user” or a PKI enabled “application”.

The following scenarios are envisioned:

1. The user requires that the certification path be valid for one of the policies of interest to the user.
2. The user requires that the certification path be valid for at least a policy, but the user does not care which policy it is. This scenario should (can) be used when the user intends to do additional policy processing using other contextual information and information content, to determine if one of the policies the certification path is valid for is acceptable to the user for the particular transaction.
3. The user has no policy related requirements on the certification path. In other words, the user is willing to accept a certification path that is not valid for any policy, but is otherwise valid.
4. The user desires that the certification path be valid for one of the policies of interest to the user, but failing that, wants the opportunity to reconsider paths that are not valid for the policies of interest to the user. This scenario should (can) be used when the user generally requires the certification path to be valid for a policy acceptable to the user, but based on other contextual information and information content, the user may wish to override policy failure.

The following sections describe how the user should go about obtaining the desired information from a compliant path validation engine.

J.1 Certification Path Valid for a User Specified Policy Required

In this scenario, the user requires that the certification path be valid for one of the policies of interest to the user. In order to obtain the desired information, the user should set the policy processing related certification path validation inputs as follows:

initial-policy-set = {set of policies of interest to the user}

initial-explicit-policy = **TRUE**

If the path validation is successful, the certification path is valid for at least one of the policies of interest to the user. The certification path is valid for the policies listed in the *user-constrained-policy-set* output variable.

Under this scenario, applications should not use a certification path if it is rejected by a path validation engine for certificate policy related failures[1].

J.2 Certification Path Valid for any Policy Required

In this scenario, the user requires that the certification path be valid for at least a policy, but the user does not care which policy it is. In order to obtain the desired information, the user should set the policy processing related certification path validation inputs as follows:

initial-policy-set = {**anyPolicy**}

initial-explicit-policy = **TRUE**

If the path validation is successful, the certification path is valid for at least a policy. The certification path is valid for the policies listed in the *user-constrained-policy-set* output variable.

Under this scenario, applications should not use a certification path if it is rejected by a path validation engine for certificate policy related failures.

J.3 Certification Path Valid Regardless of Policy

In this scenario, the user has no policy related requirements on the certification path. In order to obtain the desired information, the user should set the policy processing related certification path validation inputs as follows:

initial-policy-set = {**anyPolicy**}

initial-explicit-policy = **FALSE**

If the path validation is successful, the certification path is valid for the policies listed in the *user-constrained-policy-set* output variable.

Under this scenario, applications should not use a certification path if it is rejected by a path validation engine for certificate policy related failures.

It should be noted that in this scenario, the certification path can have policy related failure. An example is if the infrastructure (i.e. a CA certificate in the certification path) causes the setting of *explicit-policy-indicator*. In this case, if the path is not valid for any policy, i.e. *authorities-constrained-policy-set* is empty, a compliant path validation engine will return a failure. Applications should reject the certification path for failure of this type.

J.4 Certification Path Valid for a User Specific Policy Desired, but Not Required

In this scenario, the user desires that the certification path be valid for one of the policies of interest to the user, but does not want to reject paths that are not valid for any of the policies of interest to the user. In order to obtain the desired information, the user should set the policy processing related certification path validation inputs as follows:

initial-policy-set = {set of policies of interest to the user}

initial-explicit-policy = **FALSE**

If the path validation is successful, the certification path is valid for the policies listed in the *user-constrained-policy-set* output variable.. The *user-constrained-policy-set* is a subset of the *initial-policy-set*. Please note that *user-constrained-policy-set* could be **NULL** in this case when *explicit-*

[1] A path validation failure is a certificate policy related failure when failure is caused by the certificate policy related extension(s) or certificate policy related state variable(s). The certificate policy related extensions are: **certificatePolicies, policyMappings, policyConstraints,** and **inhibitAnyPolicy**. Certificate policy related state variables are: *authorities-constrained-policy-set, explicit-policy-indicator, policy-mapping-inhibit-indicator,* and *inhibit-any-policy-indicator,*

policy-indicator is not set. The application should examine the returned *user-constrained-policy-set* to determine if the path is acceptable to the user.

Applications should reject the certification path for policy related failure caused by the infrastructure in this scenario, (i.e. when the *authorities-constrained-policy-set* is empty and the *explicit-policy-indicator* is set).

It should be noted that in this scenario, the certification path can have policy related failure. An example is if the infrastructure (i.e. a CA certificate in the certification path) causes the setting of *explicit-policy-indicator*. In this case, if the path is not valid for any policy, i.e. *authorities-constrained-policy-set* is empty, a compliant path validation engine will return a failure. Applications should reject the certification path for failure of this type.

Another example is the combination of the user input and infrastructure causes policy related failure. This occurs when a CA certificate in the certification path causes the setting of *explicit-policy-indicator*, *authorities-constrained-policy-set* is not empty, and *user-constrained-policy-set* is empty. Compliant path validation engine will return a failure. Under these conditions, if the only reason for the path validation engine returning a failure is that the *user-constrained-policy-set* is empty, applications may choose to override that failure and accept the certification path. Authority imposed constraints are still respected, by virtue of the *authorities-constrained-policy-set* not being empty. Acceptance of this path by an application is equivalent to the application re-submitting the path to the validation engine with *initial-policy-set* equal to **anyPolicy** initial-explicit-policy equal to **FALSE**, and examining the returned *user-constrained-policy-set* to determine if the path is acceptable.

Add the following new informative Annex K that provides guidance on key usage certificate extension issues.

Annex K

(This annex does not form an integral part of this Recommendation | International Standard)

Key Usage Certificate Extension Issues

Editor's note: This text was agreed as part of the discussion on DTC 6 as the AFNOR text that was reached during a ballot process. Text for additional issues and risk is solicited from national bodies.

Combining the contentCommitment bit in the keyUsage certificate extension with other keyUsage bits may have security implications depending on the security environment in which the certificate is to be used. If the subject's environment can be fully controlled and trusted, then there are no specific security implications. For example, in cases where the subject is fully confident about exactly which data is signed or cases where the subject is fully confident about the security characteristics of the authentication protocol being used. If the subject's environment is not fully controlled or not fully trusted, then unintentional signing of commitments is possible. Examples include the use of badly formed authentication exchanges and the use of a rogue software component. If untrusted environments are used by a subject, these security implications can be limited through use of the following measures:

- to not combine contentCommitment key usage setting in certificates with any other key usage setting and to use the corresponding private key only with this certificate.

- to limit the use of private keys associated with certificates that have the contentCommitment key usage bit set, to environments which are considered adequately controlled and trustworthy

Replace Annex A with the following. Note that this replacement annex also contains the ASN.1 statements of several approved Technical Corrigenda. Formating will be corrected in the next version of this document.

-- *A.1 Public-Key and Attribute Certificates Framework module*

```
AuthenticationFramework {joint-iso-itu-t ds(5) module(1) authenticationFramework(7) 5}
DEFINITIONS ::=
BEGIN

-- EXPORTS All --

-- The types and values defined in this module are exported for use in the other ASN.1 modules contained
-- within the Directory Specifications, and for the use of other applications which will use them to access
-- Directory services. Other applications may use them for their own purposes, but this will not constrain
-- extensions and modifications needed to maintain or improve the Directory service.

IMPORTS
  id-at, id-nf, id-oc, informationFramework, upperBounds, selectedAttributeTypes,
basicAccessControl,
  certificateExtensions
        FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 5}

  Name, ATTRIBUTE, OBJECT-CLASS, NAME-FORM, top
        FROM InformationFramework informationFramework

  ub-user-password, ub-content
        FROM UpperBounds upperBounds

  UniqueIdentifier, octetStringMatch, DirectoryString, commonName
        FROM SelectedAttributeTypes selectedAttributeTypes

  certificateExactMatch, certificatePairExactMatch, certificateListExactMatch, KeyUsage,
GeneralNames,

      CertificatePoliciesSyntax, algorithmIdentifierMatch, CertPolicyId
        FROM CertificateExtensions certificateExtensions ;

-- public-key certificate definition --

Certificate                          ::=          SIGNED { SEQUENCE {
  version                  [0]       Version DEFAULT v1,
  serialNumber                                    CertificateSerialNumber,
  signature                                       AlgorithmIdentifier,
  issuer                             Name,
  validity                           Validity,
  subject                            Name,
  subjectPublicKeyInfo               SubjectPublicKeyInfo,
  issuerUniqueIdentifier   [1]       IMPLICIT UniqueIdentifier OPTIONAL,
                                                  -- if present, version shall be v2 or v3
  subjectUniqueIdentifier  [2]       IMPLICIT UniqueIdentifier OPTIONAL,
                                                  -- if present, version shall be v2 or v3
  extensions               [3]       Extensions OPTIONAL
                                                  -- If present, version shall be v3 -- } }

Version                    ::=       INTEGER { v1(0), v2(1), v3(2) }

CertificateSerialNumber    ::=   INTEGER

AlgorithmIdentifier        ::=       SEQUENCE {
  algorithm                ALGORITHM.&id ({SupportedAlgorithms}),
  parameters           ALGORITHM.&Type ({SupportedAlgorithms}{ @algorithm})
OPTIONAL }


-- Definition of the following information object set is deferred, perhaps to standardized
-- profiles or to protocol implementation conformance statements. The set is required to
-- specify a table constraint on the parameters component of AlgorithmIdentifier.
```

```
SupportedAlgorithms                 ALGORITHM          ::=     { ... }

Validity                      ::=   SEQUENCE {
  notBefore           Time,
  notAfter            Time }

SubjectPublicKeyInfo                ::=        SEQUENCE {
  algorithm                         AlgorithmIdentifier,
  subjectPublicKey              BIT STRING }

Time  ::=  CHOICE {
  utcTime                           UTCTime,
  generalizedTime               GeneralizedTime }

Extensions ::= SEQUENCE OF Extension
-- For those extensions where ordering of individual extensions within the SEQUENCE is
significant, the
-- specification of those individual extensions shall include the rules for the significance of the order
therein

Extension ::= SEQUENCE {
  extnId         EXTENSION.&id ({ExtensionSet}),
  critical       BOOLEAN DEFAULT FALSE,
  extnValue             OCTET STRING
                        -- contains a DER encoding of a value of type &ExtnType
                        -- for the extension object identified by extnId -- }

ExtensionSet EXTENSION              ::=        {  ... }

EXTENSION ::= CLASS {
  &id       OBJECT IDENTIFIER UNIQUE,
  &ExtnType }
WITH SYNTAX {
  SYNTAX                        &ExtnType
  IDENTIFIED BY                 &id }

-- other PKI certificate constructs

Certificates                        ::=                   SEQUENCE {
  userCertificate                                         Certificate,
  certificationPath                            ForwardCertificationPath OPTIONAL}

ForwardCertificationPath                       ::=                 SEQUENCE OF
CrossCertificates

CrossCertificates                   ::=                   SET OF Certificate

CertificationPath                   ::=                   SEQUENCE {
  userCertificate                                         Certificate,
  theCACertificates                            SEQUENCE OF CertificatePair OPTIONAL}

CertificatePair        ::=          SEQUENCE {
  forward              [0]          Certificate OPTIONAL,
  reverse              [1]          Certificate OPTIONAL
                                               -- at least one of the pair shall be present -- }

  (WITH COMPONENTS {..., forward PRESENT} I

  WITH COMPONENTS {..., reverse PRESENT})


-- certificate revocation list (CRL)

CertificateList        ::=          SIGNED { SEQUENCE {
  version                           Version OPTIONAL,
                                -- if present, version shall be v2
  signature                                    AlgorithmIdentifier,
  issuer                            Name,
  thisUpdate                        Time,
  nextUpdate                        Time OPTIONAL,
  revokedCertificates                                   SEQUENCE OF SEQUENCE {
        serialNumber                                          CertificateSerialNumber,
        revocationDate                                        Time,
        crlEntryExtensions                              Extensions OPTIONAL }
OPTIONAL,
  crlExtensions               [0]                       Extensions OPTIONAL }}


-- information object classes --


ALGORITHM  ::=    TYPE-IDENTIFIER

-- parameterized types --

HASH {ToBeHashed}                 ::=                   SEQUENCE {
  algorithmIdentifier                         AlgorithmIdentifier,
  hashValue                                             BIT STRING ( CONSTRAINED BY {
          -- shall be the result of applying a hashing procedure to the DER-encoded octets --

  -- of a value of --ToBeHashed } ) }

ENCRYPTED-HASH { ToBeSigned }          ::=                    BIT STRING ( CONSTRAINED
BY {
  -- shall be the result of applying a hashing procedure to the DER-encoded (see 6.1) octets --
  -- of a value of -- ToBeSigned -- and then applying an encipherment procedure to those octets --
})

ENCRYPTED { ToBeEnciphered }           ::=                    BIT STRING ( CONSTRAINED
BY {
  -- shall be the result of applying an encipherment procedure --
  -- to the BER-encoded octets of a value of -- ToBeEnciphered})


SIGNATURE { ToBeSigned }                      ::=             SEQUENCE {
  algorithmIdentifier                         AlgorithmIdentifier,
  encrypted                                             ENCRYPTED-HASH { ToBeSigned
}}


SIGNED { ToBeSigned }                         ::=             SEQUENCE {
  toBeSigned                                  ToBeSigned,
  COMPONENTS OF                               SIGNATURE { ToBeSigned }}


-- PKI object classes --

pkiUser   OBJECT-CLASS            ::= {
  SUBCLASS OF             {top}
  KIND              auxiliary
  MAY CONTAIN             {userCertificate}
  ID                      id-oc-pkiUser }


pkiCA   OBJECT-CLASS        ::= {
  SUBCLASS OF             {top}
  KIND              auxiliary
  MAY CONTAIN             {cACertificate |
                          certificateRevocationList |
                          authorityRevocationList |
                          crossCertificatePair }
  ID                      id-oc-pkiCA }
```

```
cRLDistributionPoint                    OBJECT-CLASS                    ::= {
  SUBCLASS OF                           { top }
  KIND                          structural
  MUST CONTAIN                  { commonName }
  MAY CONTAIN                           { certificateRevocationList |
                                        authorityRevocationList |
                                        deltaRevocationList }
  ID                                    id-oc-cRLDistributionPoint }


cRLDistPtNameForm  NAME-FORM            ::= {
  NAMES                         cRLDistributionPoint
  WITH ATTRIBUTES                       { commonName}
  ID                                    id-nf-cRLDistPtNameForm }


deltaCRL      OBJECT-CLASS              ::= {
  SUBCLASS OF                   {top}
  KIND                    auxiliary
  MAY CONTAIN                   {deltaRevocationList}
  ID                            id-oc-deltaCRL }


cpCps    OBJECT-CLASS           ::= {
  SUBCLASS OF                   {top}
  KIND                    auxiliary
  MAY CONTAIN                   {certificatePolicy |
                                certificationPracticeStmt}
  ID                            id-oc-cpCps }


pkiCertPath            OBJECT-CLASS             ::= {
  SUBCLASS OF                   {top}
  KIND                    auxiliary
  MAY CONTAIN                   { pkiPath }
  ID                            id-oc-pkiCertPath }


-- PKI directory attributes --

userCertificate                 ATTRIBUTE       ::=       {
  WITH SYNTAX                                           Certificate
  EQUALITY MATCHING RULE                certificateExactMatch
  ID                                                    id-at-userCertificate}


cACertificate                   ATTRIBUTE       ::=       {
  WITH SYNTAX                                           Certificate
  EQUALITY MATCHING RULE                certificateExactMatch
  ID                                                    id-at-cAcertificate }


crossCertificatePair          ATTRIBUTE   ::=           {
  WITH SYNTAX                                           CertificatePair
  EQUALITY MATCHING RULE                certificatePairExactMatch
  ID                                                    id-at-crossCertificatePair }


certificateRevocationList     ATTRIBUTE   ::=           {
  WITH SYNTAX                                           CertificateList
  EQUALITY MATCHING RULE                certificateListExactMatch
  ID                                                    id-at-
certificateRevocationList }


authorityRevocationList     ATTRIBUTE    ::=            {
  WITH SYNTAX                                           CertificateList
  EQUALITY MATCHING RULE                 certificateListExactMatch
  ID                                                    id-at-authorityRevocationList
}


deltaRevocationList ATTRIBUTE ::= {
  WITH SYNTAX                                           CertificateList
  EQUALITY MATCHING RULE                 certificateListExactMatch
  ID                                                    id-at-deltaRevocationList }


supportedAlgorithms  ATTRIBUTE ::= {
  WITH SYNTAX                                           SupportedAlgorithm
  EQUALITY MATCHING RULE                 algorithmIdentifierMatch
  ID                                                    id-at-supportedAlgorithms }

SupportedAlgorithm ::= SEQUENCE {
  algorithmIdentifier                                        AlgorithmIdentifier,
  intendedUsage                                         [0]            KeyUsage
OPTIONAL,
  intendedCertificatePolicies                       [1]
  CertificatePoliciesSyntax OPTIONAL }


certificationPracticeStmt   ATTRIBUTE    ::=        {
  WITH SYNTAX               InfoSyntax
  ID                            id-at-certificationPracticeStmt }


InfoSyntax     ::=     CHOICE {
  content      DirectoryString {ub-content},
  pointer      SEQUENCE {
       name                     GeneralNames,
       hash                     HASH { HashedPolicyInfo } OPTIONAL } }


POLICY   ::=   TYPE-IDENTIFIER

HashedPolicyInfo      ::=    POLICY.&Type( {Policies} )

Policies POLICY ::= {...} -- Defined by implementors --


certificatePolicy      ATTRIBUTE    ::=      {
  WITH SYNTAX               PolicySyntax
  ID                        id-at-certificatePolicy }


PolicySyntax  ::=     SEQUENCE {
  policyIdentifier          PolicyID,
  policySyntax              InfoSyntax
  }


PolicyID       ::=     CertPolicyId
```

```
pkiPath   ATTRIBUTE          ::= {
  WITH SYNTAX                PkiPath
  ID                         id-at-pkiPath }


PkiPath   ::=   SEQUENCE OF Certificate


userPassword        ATTRIBUTE    ::=       {
  WITH SYNTAX                                       OCTET STRING (SIZE (0..ub-user-
password))
  EQUALITY MATCHING RULE                   octetStringMatch
  ID                                                id-at-userPassword }


-- object identifier assignments --

-- object classes --

id-oc-cRLDistributionPoint              OBJECT IDENTIFIER ::=      {id-oc 19}
id-oc-pkiUser                           OBJECT IDENTIFIER ::=      {id-oc 21}
id-oc-pkiCA                             OBJECT IDENTIFIER ::=      {id-oc 22}
id-oc-deltaCRL                          OBJECT IDENTIFIER ::=      {id-oc 23}
id-oc-cpCps                             OBJECT IDENTIFIER ::=      {id-oc 30}
id-oc-pkiCertPath                       OBJECT IDENTIFIER ::=      {id-oc 31}


--name forms--

id-nf-cRLDistPtNameForm                 OBJECT IDENTIFIER ::=      {id-nf 14}


--directory attributes--

id-at-userPassword                      OBJECT IDENTIFIER ::=      {id-at 35}
id-at-userCertificate                   OBJECT IDENTIFIER ::=      {id-at 36}
id-at-cAcertificate                     OBJECT IDENTIFIER ::=      {id-at 37}
id-at-authorityRevocationList           OBJECT IDENTIFIER ::=      {id-at 38}
id-at-certificateRevocationList         OBJECT IDENTIFIER ::=      {id-at 39}
id-at-crossCertificatePair              OBJECT IDENTIFIER ::=      {id-at 40}
id-at-supportedAlgorithms               OBJECT IDENTIFIER ::=      {id-at 52}
id-at-deltaRevocationList               OBJECT IDENTIFIER ::=      {id-at 53}
id-at-certificationPracticeStmt         OBJECT IDENTIFIER ::=      {id-at 68}
id-at-certificatePolicy                 OBJECT IDENTIFIER ::=      {id-at 69}
id-at-pkiPath                           OBJECT IDENTIFIER ::=      {id-at 70}


END

-- A.2   Certificate extensions module

CertificateExtensions {joint-iso-itu-t ds(5) module(1) certificateExtensions(26) 5}
DEFINITIONS IMPLICIT TAGS ::=
BEGIN

-- EXPORTS ALL --

IMPORTS
  id-at, id-ce, id-mr, informationFramework, authenticationFramework,
        selectedAttributeTypes, upperBounds
        FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1)
        usefulDefinitions(0) 5}

  Name, RelativeDistinguishedName, ATTRIBUTE, Attribute, MATCHING-RULE
        FROM InformationFramework informationFramework
  CertificateSerialNumber, CertificateList, AlgorithmIdentifier,
        EXTENSION, Time, PolicyID
        FROM AuthenticationFramework authenticationFramework

  DirectoryString
        FROM SelectedAttributeTypes selectedAttributeTypes

  ub-name
        FROM UpperBounds upperBounds

  ORAddress
        FROM MTSAbstractService {joint-iso-itu-t mhs(6) mts(3)
        modules(0) mts-abstract-service(1) version-1999 (1) } ;

-- Unless explicitly noted otherwise, there is no significance to the ordering
-- of components of a SEQUENCE OF construct in this Specification.


-- public-key certificate and CRL extensions --


authorityKeyIdentifier EXTENSION ::= {
  SYNTAX                   AuthorityKeyIdentifier
  IDENTIFIED BY     id-ce-authorityKeyIdentifier }

AuthorityKeyIdentifier ::= SEQUENCE {
  keyIdentifier                      [0] KeyIdentifier                            OPTIONAL,
  authorityCertIssuer                        [1] GeneralNames
  OPTIONAL,
  authorityCertSerialNumber[2] CertificateSerialNumber                 OPTIONAL }
  ( WITH COMPONENTS                {..., authorityCertIssuer PRESENT,
                    authorityCertSerialNumber PRESENT} |
  WITH COMPONENTS                  {..., authorityCertIssuer ABSENT,
                    authorityCertSerialNumber ABSENT} )

KeyIdentifier ::= OCTET STRING


subjectKeyIdentifier EXTENSION ::= {
  SYNTAX                   SubjectKeyIdentifier
  IDENTIFIED BY     id-ce-subjectKeyIdentifier }

SubjectKeyIdentifier ::= KeyIdentifier


keyUsage EXTENSION ::= {
  SYNTAX                   KeyUsage
  IDENTIFIED BY     id-ce-keyUsage }

KeyUsage ::= BIT STRING {
  digitalSignature          (0),
  nonRepudiation            (1),
  keyEncipherment           (2),
  dataEncipherment          (3),
  keyAgreement                    (4),
  keyCertSign                     (5),
  cRLSign                         (6),
  encipherOnly                    (7),
  decipherOnly                    (8) }


extKeyUsage EXTENSION ::= {
  SYNTAX                   SEQUENCE SIZE (1..MAX) OF KeyPurposeId
  IDENTIFIED BY     id-ce-extKeyUsage }

KeyPurposeId ::= OBJECT IDENTIFIER
```

```
privateKeyUsagePeriod EXTENSION ::= {
  SYNTAX                     PrivateKeyUsagePeriod
  IDENTIFIED BY              id-ce-privateKeyUsagePeriod }

PrivateKeyUsagePeriod ::= SEQUENCE {
  notBefore          [0]     GeneralizedTime OPTIONAL,
  notAfter           [1]     GeneralizedTime OPTIONAL }
  ( WITH COMPONENTS {..., notBefore PRESENT} |
  WITH COMPONENTS            {..., notAfter PRESENT} )


certificatePolicies EXTENSION ::= {
  SYNTAX                     CertificatePoliciesSyntax
  IDENTIFIED BY              id-ce-certificatePolicies }

CertificatePoliciesSyntax ::= SEQUENCE SIZE (1..MAX) OF PolicyInformation

PolicyInformation ::= SEQUENCE {
  policyIdentifier           CertPolicyId,
  policyQualifiers   SEQUENCE SIZE (1..MAX) OF
                                   PolicyQualifierInfo OPTIONAL }

CertPolicyId ::= OBJECT IDENTIFIER

PolicyQualifierInfo ::= SEQUENCE {
  policyQualifierId          CERT-POLICY-QUALIFIER.&id
                                   ({SupportedPolicyQualifiers}),
  qualifier                        CERT-POLICY-QUALIFIER.&Qualifier
                                   ({SupportedPolicyQualifiers}{@policyQualifierId})
                                   OPTIONAL }

SupportedPolicyQualifiers CERT-POLICY-QUALIFIER ::= { ... }

anyPolicy     OBJECT IDENTIFIER ::=       { 2 5 29 32 0 }

CERT-POLICY-QUALIFIER ::= CLASS {
  &id         OBJECT IDENTIFIER UNIQUE,
  &Qualifier  OPTIONAL }
WITH SYNTAX {
  POLICY-QUALIFIER-ID        &id
  [QUALIFIER-TYPE  &Qualifier] }


policyMappings EXTENSION ::= {
  SYNTAX                     PolicyMappingsSyntax
  IDENTIFIED BY      id-ce-policyMappings }

PolicyMappingsSyntax ::= SEQUENCE SIZE (1..MAX) OF SEQUENCE {
  issuerDomainPolicy               CertPolicyId,
  subjectDomainPolicy              CertPolicyId }


subjectAltName EXTENSION ::= {
  SYNTAX                     GeneralNames
  IDENTIFIED BY      id-ce-subjectAltName }

GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName

GeneralName ::= CHOICE {
  otherName                        [0]     INSTANCE OF OTHER-NAME,
  rfc822Name                       [1]     IA5String,
  dNSName                                  [2]          IA5String,
  x400Address                              [3]          ORAddress,
  directoryName                            [4]          Name,
  ediPartyName                             [5]          EDIPartyName,
  uniformResourceIdentifier [6]    IA5String,
  iPAddress                          [7]          OCTET STRING,
  registeredID                  [8]     OBJECT IDENTIFIER }


OTHER-NAME ::= TYPE-IDENTIFIER

EDIPartyName ::= SEQUENCE {
  nameAssigner              [0]     DirectoryString {ub-name} OPTIONAL,
  partyName          [1]      DirectoryString {ub-name} }


issuerAltName EXTENSION ::= {
  SYNTAX                    GeneralNames
  IDENTIFIED BY      id-ce-issuerAltName }


subjectDirectoryAttributes EXTENSION ::= {
  SYNTAX                    AttributesSyntax
  IDENTIFIED BY      id-ce-subjectDirectoryAttributes }

AttributesSyntax ::= SEQUENCE SIZE (1..MAX) OF Attribute


basicConstraints EXTENSION ::= {
  SYNTAX                    BasicConstraintsSyntax
  IDENTIFIED BY      id-ce-basicConstraints }

BasicConstraintsSyntax ::= SEQUENCE {
  cA                                BOOLEAN DEFAULT FALSE,
  pathLenConstraint                 INTEGER (0..MAX) OPTIONAL }


nameConstraints EXTENSION ::= {
  SYNTAX             NameConstraintsSyntax
  IDENTIFIED BY      id-ce-nameConstraint }

NameConstraintsSyntax ::= SEQUENCE {
  permittedSubtrees         [0]     GeneralSubtrees OPTIONAL,
  excludedSubtrees          [1]     GeneralSubtrees OPTIONAL,

  requiredNameForms         [2]     NameForms OPTIONAL  }

GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree

GeneralSubtree ::= SEQUENCE {
  base               GeneralName,
  minimum    [0]     BaseDistance DEFAULT 0,
  maximum    [1]     BaseDistance OPTIONAL }

BaseDistance ::= INTEGER (0..MAX)


NameForms  ::= SEQUENCE {

   basicNameForms [0]       BasicNameForms OPTIONAL,

   otherNameForms [1]       SEQUENCE SIZE (1..MAX) OF OBJECT IDENTIFIER OPTIONAL }

(ALL EXCEPT ({ --none; i.e.:at least one component shall be present-- }))


BasicNameForms  ::= BIT STRING {

         rfc822Name                 (0),

  dNSName                   (1),
  x400Address                (2),

  directoryName              (3),

  ediPartyName               (4),

  uniformResourceIdentifier (5),

  iPAddress                  (6),

  registeredID               (7) }  (SIZE (1..MAX))


policyConstraints EXTENSION ::= {
  SYNTAX                     PolicyConstraintsSyntax
  IDENTIFIED BY      id-ce-policyConstraints }

PolicyConstraintsSyntax ::= SEQUENCE {
  requireExplicitPolicy      [0] SkipCerts OPTIONAL,
  inhibitPolicyMapping       [1] SkipCerts OPTIONAL }

SkipCerts ::= INTEGER (0..MAX)


cRLNumber EXTENSION ::= {

  SYNTAX                              CRLNumber

  IDENTIFIED BY      id-ce-cRLNumber }

CRLNumber ::= INTEGER (0..MAX)


reasonCode EXTENSION ::= {
  SYNTAX                     CRLReason
  IDENTIFIED BY      id-ce-reasonCode }

CRLReason ::= ENUMERATED {
  unspecified                         (0),
  keyCompromise                       (1),
  cACompromise                        (2),
  affiliationChanged                  (3),
  superseded                          (4),
  cessationOfOperation                (5),
  certificateHold                               (6),
  removeFromCRL                       (8),
  privilegeWithdrawn                            (9),
  aaCompromise                        (10) }


holdInstructionCode EXTENSION ::= {
  SYNTAX                     HoldInstruction
  IDENTIFIED BY      id-ce-instructionCode }

HoldInstruction ::= OBJECT IDENTIFIER


invalidityDate EXTENSION ::= {
  SYNTAX                     GeneralizedTime
  IDENTIFIED BY      id-ce-invalidityDate }


crlScope EXTENSION  ::= {
  SYNTAX                     CRLScopeSyntax
  IDENTIFIED BY      id-ce-cRLScope }

CRLScopeSyntax  ::=          SEQUENCE SIZE (1..MAX) OF PerAuthorityScope
```

```
PerAuthorityScope ::= SEQUENCE {
  authorityName                  [0]                 GeneralName OPTIONAL,
  distributionPoint         [1]            DistributionPointName OPTIONAL,
  onlyContains                   [2]                 OnlyCertificateTypes OPTIONAL,
  onlySomeReasons           [4]            ReasonFlags OPTIONAL,
  serialNumberRange              [5]                 NumberRange OPTIONAL,
  subjectKeyIdRange              [6]                 NumberRange OPTIONAL,
  nameSubtrees                   [7]                 GeneralNames OPTIONAL,
  baseRevocationInfo        [9]            BaseRevocationInfo OPTIONAL
  }
```

-- (Note that this replacement text for the IDP extension may get changed again. There are serious migration and backward compatibility issues with the approved solution for DR 280 (reflected here) and new initiatives are planned to reverse some of these changes it is NOT recommended that anyone actually follow this change (at least not until the new initiatives are resolved).

```
OnlyCertificateTypes        ::= BIT STRING {

  userPublicKey     (0),

  cA                (1),

  userAttribute     (2),

  aA                (3),

  sOAPublicKey      (4) }

NumberRange ::= SEQUENCE {
  startingNumber    [0]            INTEGER OPTIONAL,
  endingNumber              [1]            INTEGER OPTIONAL,
  modulus                                  INTEGER OPTIONAL }


BaseRevocationInfo ::= SEQUENCE {
  cRLStreamIdentifier            [0]     CRLStreamIdentifier          OPTIONAL,
  cRLNumber                 [1]     CRLNumber,
  baseThisUpdate            [2]     GeneralizedTime }

statusReferrals EXTENSION  ::= {
  SYNTAX                    StatusReferrals
  IDENTIFIED BY     id-ce-statusReferrals }

StatusReferrals ::= SEQUENCE SIZE (1..MAX) OF StatusReferral

StatusReferral  ::=  CHOICE {
  cRLReferral       [0]            CRLReferral,
  otherReferral             [1]            INSTANCE OF OTHER-REFERRAL}

CRLReferral ::= SEQUENCE  {
  issuer            [0]            GeneralName OPTIONAL,
  location                  [1]            GeneralName OPTIONAL,
  deltaRefInfo      [2]            DeltaRefInfo OPTIONAL,
  cRLScope                                 CRLScopeSyntax,
  lastUpdate        [3]            GeneralizedTime OPTIONAL,
  lastChangedCRL    [4]            GeneralizedTime OPTIONAL}

DeltaRefInfo  ::=  SEQUENCE {
  deltaLocation             GeneralName,
  lastDelta                 GeneralizedTime OPTIONAL }

OTHER-REFERRAL ::= TYPE-IDENTIFIER


cRLStreamIdentifier EXTENSION    ::= {
  SYNTAX                    CRLStreamIdentifier
  IDENTIFIED BY             id-ce-cRLStreamIdentifier }
```

```
CRLStreamIdentifier  ::=  INTEGER (0..MAX)


orderedList EXTENSION        ::= {
  SYNTAX                     OrderedListSyntax
  IDENTIFIED BY              id-ce-orderedList }

OrderedListSyntax  ::= ENUMERATED {
ascSerialNum         (0),
ascRevDate           (1) }


deltaInfo EXTENSION          ::=     {
  SYNTAX                     DeltaInformation
  IDENTIFIED BY              id-ce-deltaInfo }

DeltaInformation     ::=     SEQUENCE {
  deltaLocation              GeneralName,
  nextDelta                  GeneralizedTime OPTIONAL }


cRLDistributionPoints EXTENSION ::= {
  SYNTAX                     CRLDistPointsSyntax
  IDENTIFIED BY      id-ce-cRLDistributionPoints }

CRLDistPointsSyntax ::= SEQUENCE SIZE (1..MAX) OF DistributionPoint

DistributionPoint ::= SEQUENCE {
  distributionPoint   [0]            DistributionPointName OPTIONAL,
  reasons                    [1]             ReasonFlags OPTIONAL,
  cRLIssuer                  [2]             GeneralNames OPTIONAL }

DistributionPointName ::= CHOICE {
  fullName                                   [0]               GeneralNames,
  nameRelativeToCRLIssuer [1]                RelativeDistinguishedName }

ReasonFlags ::= BIT STRING {
  unused                             (0),
  keyCompromise              (1),
  cACompromise               (2),
  affiliationChanged         (3),
  superseded                 (4),
  cessationOfOperation       (5),
  certificateHold                    (6),
  privilegeWithdrawn                 (7),
  aACompromise               (8) }
```

-- (Note that this replacement text for the IDP extension may get changed again. There are serious migration andbackward compatibility issues with the approved solution for DR 280 (reflected here) and new initiatives are planned to reverse some of these changes – it is NOT recommended that anyone actually follow this change (at least not until the new initiatives are resolved).

```
issuingDistributionPoint       EXTENSION     ::=              {
  SYNTAX              IssuingDistPointSyntax
  IDENTIFIED BY       id-ce-issuingDistributionPoint }

IssuingDistPointSyntax ::= SEQUENCE {

--If containsUserPublicKeyCerts, containsCACerts, containsUserAttributeCerts,
containsAACerts,  and

-- containsSOAPublicKeyCerts s are all absent, or not set to TRUE, (), the CRL covers
allthese  certificate types--
```

```
   distributionPoint              [0] DistributionPointName OPTIONAL,
   containsUserPublicKeyCerts     [1] BOOLEAN DEFAULT FALSE,
   containsCACerts                [2] BOOLEAN DEFAULT FALSE,
   onlySomeReasons                [3] ReasonFlags OPTIONAL,
   indirectCRL                    [4] BOOLEAN DEFAULT FALSE,

   containsUserAttributeCerts     [5] BOOLEAN DEFAULT FALSE,

   containsAACerts                [6] BOOLEAN DEFAULT FALSE,

   containsSOAPublicKeyCerts      [7] BOOLEAN DEFAULT FALSE }


certificateIssuer EXTENSION ::= {
   SYNTAX                    GeneralNames
   IDENTIFIED BY       id-ce-certificateIssuer }


deltaCRLIndicator EXTENSION ::= {
   SYNTAX                    BaseCRLNumber
   IDENTIFIED BY       id-ce-deltaCRLIndicator }


BaseCRLNumber ::= CRLNumber


baseUpdateTime EXTENSION          ::= {
   SYNTAX                    GeneralizedTime
   IDENTIFIED BY             id-ce-baseUpdateTime }


freshestCRL EXTENSION        ::= {
   SYNTAX                    CRLDistPointsSyntax
   IDENTIFIED BY             id-ce-freshestCRL }


inhibitAnyPolicy       EXTENSION     ::= {
   SYNTAX              SkipCerts
IDENTIFIED BY          id-ce-inhibitAnyPolicy }


-- toBeRevoked Extension added 9/14/2003


toBeRevoked  EXTENSION  ::= {

   SYNTAX              ToBeRevokedSyntax

   IDENTIFIED BY       id-ce-toBeRevoked }


ToBeRevokedSyntax::=         SEQUENCE SIZE(1..MAX) OF ToBeRevokedGroup


ToBeRevokedGroup ::=         SEQUENCE {

   certificateIssuer   [0]   GeneralName OPTIONAL,

   reasonInfo          [1]   ReasonInfo     OPTIONAL,

   revocationTime            GeneralizedTime,

   certificateGroup          CertificateGroup }
```

```
ReasonInfo                  ::=     SEQUENCE {
  reasonCode                CRLReason,
  holdInstructionCode       HoldInstruction OPTIONAL }

CertificateGroup      ::=   CHOICE {
  serialNumbers             [0]     CertificateSerialNumbers,
  serialNumberRange         [1]     CertificateGroupNumberRange,
  nameSubtree               [2]     GeneralName }

CertificateGroupNumberRange ::= SEQUENCE {
  startingNumber      [0]   INTEGER,
  endingNumber        [1]   INTEGER }

CertificateSerialNumbers ::= SEQUENCE SIZE(1..MAX) OF CertificateSerialNumber

-- revoked groups extension added 9/14/2003

revokedGroups  EXTENSION ::= {
  SYNTAX              RevokedGroupsSyntax
  IDENTIFIED BY       id-ce-RevokedGroups }

RevokedGroupsSynta ::= SEQUENCE SIZE (1..MAX) OF RevokedGroup

RevokedGroup ::= SEQUENCE {
  certificateIssuer   [0]   GeneralName OPTIONAL,
  reasonInfo          [1]   ReasonInfo OPTIONAL,
  invalidityDate      [2]   GeneralizedTime OPTIONAL,
  revokedcertificateGroup   RevokedCertificateGroup }

RevokedCertificateGroup ::= CHOICE {
  serialNumberRange         NumberRange,
  nameSubtree               GeneralName }

-- expired certificates extension added 9/14/2003

expiredCertsOnCRL EXTENSION ::= {
  SYNTAX              ExpiredCertsOnCRL
  IDENTIFIED BY       id-ce-expiredCertsOnCRL }

ExpiredCertsOnCRL ::= GeneralizedTime
```

```
-- PKI matching rules --


certificateExactMatch MATCHING-RULE ::= {
  SYNTAX              CertificateExactAssertion
  ID                  id-mr-certificateExactMatch }

CertificateExactAssertion ::= SEQUENCE {
  serialNumber                CertificateSerialNumber,
  issuer              Name }


certificateMatch MATCHING-RULE ::= {
  SYNTAX              CertificateAssertion
  ID                  id-mr-certificateMatch }

CertificateAssertion ::= SEQUENCE {
  serialNumber                      [0] CertificateSerialNumber          OPTIONAL,
  issuer                    [1] Name                                          OPTIONAL,
  subjectKeyIdentifier      [2] SubjectKeyIdentifier             OPTIONAL,
  authorityKeyIdentifier    [3] AuthorityKeyIdentifier           OPTIONAL,
  certificateValid                  [4] Time
  OPTIONAL,
  privateKeyValid           [5] GeneralizedTime                           OPTIONAL,
  subjectPublicKeyAlgID     [6] OBJECT IDENTIFIER                OPTIONAL,
  keyUsage                          [7] KeyUsage
  OPTIONAL,
  subjectAltName            [8] AltNameType                                   OPTIONAL,
  policy                    [9] CertPolicySet                                 OPTIONAL,
  pathToName                        [10] Name
  OPTIONAL,
  subject                   [11]    Name                                      OPTIONAL,
  nameConstraints           [12]    NameConstraintsSyntax        OPTIONAL }

AltNameType ::= CHOICE {
  builtinNameForm   ENUMERATED {
            rfc822Name                                           (1),
            dNSName                                     (2),
            x400Address                                          (3),
            directoryName                               (4),
            ediPartyName                                (5),
            uniformResourceIdentifier                   (6),
            iPAddress                                   (7),
            registeredId                                         (8) },
  otherNameForm     OBJECT IDENTIFIER }

CertPolicySet ::= SEQUENCE SIZE (1..MAX) OF CertPolicyId


certificatePairExactMatch MATCHING-RULE                 ::=     {
  SYNTAX              CertificatePairExactAssertion
  ID                  id-mr-certificatePairExactMatch }

CertificatePairExactAssertion ::= SEQUENCE {
  issuedToThisCAAssertion           [0] CertificateExactAssertion OPTIONAL,
  issuedByThisCAAssertion           [1] CertificateExactAssertion OPTIONAL }
  ( WITH COMPONENTS                         {..., issuedToThisCAAssertion PRESENT} |
    WITH COMPONENTS                         {..., issuedByThisCAAssertion PRESENT} )


certificatePairMatch MATCHING-RULE ::=    {
  SYNTAX              CertificatePairAssertion
  ID                  id-mr-certificatePairMatch }
```

```
CertificatePairAssertion ::= SEQUENCE {
  issuedToThisCAAssertion  [0] CertificateAssertion OPTIONAL,
  issuedByThisCAAssertion  [1] CertificateAssertion OPTIONAL }
  ( WITH COMPONENTS        {..., issuedToThisCAAssertion PRESENT} |
    WITH COMPONENTS        {..., issuedByThisCAAssertion PRESENT} )


certificateListExactMatch MATCHING-RULE::=          {
  SYNTAX              CertificateListExactAssertion
  ID                  id-mr-certificateListExactMatch }

CertificateListExactAssertion ::= SEQUENCE {
  issuer              Name,
  thisUpdate          Time,
  distributionPoint   DistributionPointName OPTIONAL }


certificateListMatch MATCHING-RULE ::= {
  SYNTAX              CertificateListAssertion
  ID                  id-mr-certificateListMatch }

CertificateListAssertion ::= SEQUENCE {
  issuer                                      Name                    OPTIONAL,
  minCRLNumber              [0]               CRLNumber        OPTIONAL,
  maxCRLNumber              [1]               CRLNumber        OPTIONAL,
  reasonFlags                                 ReasonFlags             OPTIONAL,
  dateAndTime                                             Time                OPTIONAL,
  distributionPoint         [2]               DistributionPointName OPTIONAL,
  authorityKeyIdentifier    [3]               AuthorityKeyIdentifier OPTIONAL}


algorithmIdentifierMatch MATCHING-RULE ::=   {
  SYNTAX              AlgorithmIdentifier
  ID                  id-mr-algorithmIdentifierMatch }


policyMatch MATCHING-RULE       ::= {
  SYNTAX              PolicyID
  ID                  id-mr-policyMatch }


pkiPathMatch MATCHING-RULE  ::= {
SYNTAX        PkiPathMatchSyntax
ID                    id-mr-pkiPathMatch }


PkiPathMatchSyntax ::= SEQUENCE {
firstIssuer           Name,
lastSubject           Name }


-- enhanced certificate match added 9/14/2003


enhancedCertificateMatch MATCHING-RULE  ::= {
  SYNTAX      EnhancedCertificateAssertion
  ID          id-mr-enhancedCertificateMatch }

EnhancedCertificateAssertion ::= SEQUENCE {
  serialNumber                  [0]      CertificateSerialNumber     OPTIONAL,
  issuer                        [1]      Name
  OPTIONAL,
  subjectKeyIdentifier          [2]      SubjectKeyIdentifier                OPTIONAL,
  authorityKeyIdentifier      [3]     AuthorityKeyIdentifier             OPTIONAL,
  certificateValid                    [4]      Time
  OPTIONAL,
  privateKeyValid                     [5]      GeneralizedTime                    OPTIONAL,
  subjectPublicKeyAlgID       [6]     OBJECT IDENTIFIER          OPTIONAL,
  keyUsage                            [7]      KeyUsage                           OPTIONAL,
  subjectAltName                      [8]      AltName
  OPTIONAL,
  policy                              [9]      CertPolicySet
  OPTIONAL,
  pathToName                          [10]     GeneralNames                       OPTIONAL,

  subject                     [11] Name                       OPTIONAL,

  nameConstraints             [12] NameConstraintsSyntax      OPTIONAL

}


(ALL EXCEPT ({ --none; at least one component shall be present -- }))

AltName  ::= SEQUENCE {

  altnameType         AltNameType,

  altNameValue        GeneralName OPTIONAL }



-- Object identifier assignments --

id-ce-subjectDirectoryAttributes           OBJECT IDENTIFIER        ::=       {id-ce 9}
id-ce-subjectKeyIdentifier                 OBJECT IDENTIFIER        ::=       {id-ce 14}
id-ce-keyUsage                             OBJECT IDENTIFIER        ::=       {id-ce 15}
id-ce-privateKeyUsagePeriod                OBJECT IDENTIFIER        ::=       {id-ce 16}
id-ce-subjectAltName                       OBJECT IDENTIFIER        ::=       {id-ce 17}
id-ce-issuerAltName                        OBJECT IDENTIFIER        ::=       {id-ce 18}
id-ce-basicConstraints                     OBJECT IDENTIFIER        ::=       {id-ce 19}
id-ce-cRLNumber                            OBJECT IDENTIFIER        ::=       {id-ce 20}
id-ce-reasonCode                           OBJECT IDENTIFIER        ::=       {id-ce 21}
id-ce-instructionCode                      OBJECT IDENTIFIER        ::=       {id-ce 23}
id-ce-invalidityDate                       OBJECT IDENTIFIER        ::=       {id-ce 24}
id-ce-deltaCRLIndicator                    OBJECT IDENTIFIER        ::=       {id-ce 27}
id-ce-issuingDistributionPoint             OBJECT IDENTIFIER        ::=       {id-ce 28}
id-ce-certificateIssuer                    OBJECT IDENTIFIER        ::=       {id-ce 29}
id-ce-nameConstraint                       OBJECT IDENTIFIER        ::=       {id-ce 30 1}

id-ce-cRLDistributionPoints                OBJECT IDENTIFIER        ::=       {id-ce 31}
id-ce-certificatePolicies                  OBJECT IDENTIFIER        ::=       {id-ce 32}
id-ce-policyMappings                       OBJECT IDENTIFIER        ::=       {id-ce 33}
id-ce-authorityKeyIdentifier               OBJECT IDENTIFIER        ::=       {id-ce 35}
id-ce-policyConstraints                    OBJECT IDENTIFIER        ::=       {id-ce 36}
id-ce-extKeyUsage                          OBJECT IDENTIFIER        ::=       {id-ce 37}
id-ce-cRLStreamIdentifier                  OBJECT IDENTIFIER         ::=      {id-ce 40}
id-ce-cRLScope                             OBJECT IDENTIFIER        ::=       {id-ce 44}
id-ce-statusReferrals                      OBJECT IDENTIFIER        ::=       {id-ce 45}
id-ce-freshestCRL                          OBJECT IDENTIFIER        ::=       {id-ce 46}
id-ce-orderedList                          OBJECT IDENTIFIER        ::=       {id-ce 47}
id-ce-baseUpdateTime                       OBJECT IDENTIFIER        ::=       {id-ce 51}
id-ce-deltaInfo                            OBJECT IDENTIFIER        ::=       {id-ce 53}
id-ce-inhibitAnyPolicy                     OBJECT IDENTIFIER        ::=       {id-ce 54}

id-ce-toBeRevoked                         OBJECT IDENTIFIER         ::=       {id-ce 58}

id-ce-revokedGroups                       OBJECT IDENTIFIER         ::=       {id-ce 59}

id-ce-expiredCertsOnCRL                    OBJECT IDENTIFIER        ::=       {id-ce 60}
```

```
-- matching rule OIDs --

id-mr-certificateExactMatch              OBJECT IDENTIFIER     ::=     {id-mr 34}
id-mr-certificateMatch                   OBJECT IDENTIFIER     ::=     {id-mr 35}
id-mr-certificatePairExactMatch          OBJECT IDENTIFIER     ::=     {id-mr 36}
id-mr-certificatePairMatch               OBJECT IDENTIFIER     ::=     {id-mr 37}
id-mr-certificateListExactMatch          OBJECT IDENTIFIER     ::=     {id-mr 38}
id-mr-certificateListMatch               OBJECT IDENTIFIER     ::=     {id-mr 39}
id-mr-algorithmIdentifierMatch           OBJECT IDENTIFIER     ::=     {id-mr 40}
id-mr-policyMatch                        OBJECT IDENTIFIER     ::=     {id-mr 60}
id-mr-pkiPathMatch                       OBJECT IDENTIFIER     ::=     {id-mr 62}

id-mr-enhancedCertificateMatch          OBJECT IDENTIFIER     ::=      {id-mr 65}

-- The following OBJECT IDENTIFIERS are not used by this Specification:
-- {id-ce 2}, {id-ce 3}, {id-ce 4}, {id-ce 5}, {id-ce 6}, {id-ce 7},
-- {id-ce 8}, {id-ce 10}, {id-ce 11}, {id-ce 12}, {id-ce 13},
-- {id-ce 22}, {id-ce 25}, {id-ce 26}, {id-ce 30}

END




-- A.3    Attribute Certificate Framework module

AttributeCertificateDefinitions {joint-iso-itu-t ds(5) module(1)
attributeCertificateDefinitions(32) 5}
DEFINITIONS IMPLICIT TAGS ::=
BEGIN

-- EXPORTS ALL --

IMPORTS
  id-at, id-ce, id-mr, informationFramework, authenticationFramework,
         selectedAttributeTypes, upperBounds, id-oc, certificateExtensions
         FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1)
         usefulDefinitions(0) 5}


  Name, RelativeDistinguishedName, ATTRIBUTE, Attribute,
         MATCHING-RULE, AttributeType, OBJECT-CLASS, top
         FROM InformationFramework informationFramework

  CertificateSerialNumber, CertificateList, AlgorithmIdentifier,
         EXTENSION, SIGNED, InfoSyntax, PolicySyntax, Extensions, Certificate
         FROM AuthenticationFramework authenticationFramework

  DirectoryString, TimeSpecification, UniqueIdentifier
         FROM SelectedAttributeTypes selectedAttributeTypes

  GeneralName, GeneralNames, NameConstraintsSyntax, certificateListExactMatch

         FROM CertificateExtensions certificateExtensions

  ub-name
         FROM UpperBounds upperBounds

  UserNotice
         FROM PKIX1Implicit93 {iso(1) identified-organization(3) dod(6) internet(1) security(5)
mechanisms(5) pkix(7) id-mod(0) id-pkix1-implicit-93(4)}
```

```
  ORAddress
         FROM MTSAbstractService {joint-iso-itu-t mhs(6) mts(3)
         modules(0) mts-abstract-service(1) version-1999 (1) } ;

-- Unless explicitly noted otherwise, there is no significance to the ordering
-- of components of a SEQUENCE OF construct in this Specification.


-- attribute certificate constructs --


AttributeCertificate ::= SIGNED {AttributeCertificateInfo}

AttributeCertificateInfo ::= SEQUENCE
  {
  version                                   AttCertVersion, -- version is v2
  holder                            Holder,
  issuer                            AttCertIssuer,
  signature                                 AlgorithmIdentifier,
  serialNumber                              CertificateSerialNumber,
  attrCertValidityPeriod            AttCertValidityPeriod,
  attributes                                SEQUENCE OF Attribute,
  issuerUniqueID                    UniqueIdentifier OPTIONAL,
  extensions                        Extensions                OPTIONAL
  }


AttCertVersion       ::= INTEGER { v2(1) }


Holder ::=      SEQUENCE
         {
         baseCertificateID          [0] IssuerSerial                   OPTIONAL,
              -- the issuer and serial number of  the holder's Public Key Certificate
         entityName                         [1] GeneralNames                    OPTIONAL,
              -- the name of the entity or role
         objectDigestInfo           [2] ObjectDigestInfo        OPTIONAL
              -- used to directly authenticate the holder, e.g. an executable
-- at least one of baseCertificateID, entityName or objectDigestInfo shall be present --}


ObjectDigestInfo    ::= SEQUENCE {
  digestedObjectType  ENUMERATED {
         publicKey                  (0),
         publicKeyCert              (1),
         otherObjectTypes     (2) },
  otherObjectTypeID          OBJECT IDENTIFIER  OPTIONAL,
  digestAlgorithm            AlgorithmIdentifier,
  objectDigest                      BIT STRING }


AttCertIssuer ::=     [0]      SEQUENCE {

    issuerName                      GeneralNames  OPTIONAL,

    baseCertificateID [0]      IssuerSerial  OPTIONAL,

    objectDigestInfo  [1]      ObjectDigestInfo  OPTIONAL }

-- At least one component shall be present
  ( WITH COMPONENTS { ..., issuerName  PRESENT } |
    WITH COMPONENTS { ..., baseCertificateID  PRESENT } |
    WITH COMPONENTS { ..., objectDigestInfo PRESENT } )
```

```
IssuerSerial  ::=  SEQUENCE {
  issuer          GeneralNames,
  serial          CertificateSerialNumber,
  issuerUID               UniqueIdentifier OPTIONAL }

AttCertValidityPeriod  ::= SEQUENCE {
  notBeforeTime                 GeneralizedTime,
  notAfterTime                  GeneralizedTime }

AttributeCertificationPath  ::=  SEQUENCE {
  attributeCertificate                  AttributeCertificate,
  acPath                                SEQUENCE OF ACPathData OPTIONAL }

ACPathData  ::=  SEQUENCE {
  certificate                   [0]  Certificate  OPTIONAL,
  attributeCertificate [1]  AttributeCertificate  OPTIONAL }


PrivilegePolicy         ::=         OBJECT IDENTIFIER


-- privilege attributes --


role      ATTRIBUTE  ::= {
          WITH SYNTAX                   RoleSyntax
          ID                                          id-at-role }


RoleSyntax     ::=       SEQUENCE {
  roleAuthority                [0]                GeneralNames OPTIONAL,
  roleName                     [1]                GeneralName }


-- xml Privilege Information attribute added 9/14/2003


xmlPrivilegeInfo        ATTRIBUTE  ::= {

  WITH SYNTAX UTF8String –contains XML-encoded privilege information

  ID                      id-at-xMLPrivilegeInfo }




-- PMI object classes --


pmiUser OBJECT-CLASS ::= {
  SUBCLASS OF                   {top}
  KIND                    auxiliary
  MAY CONTAIN                   {attributeCertificateAttribute}
  ID                            id-oc-pmiUser
  }


pmiAA OBJECT-CLASS ::= {
 -- a PMI AA
  SUBCLASS OF                   {top}
  KIND                    auxiliary
  MAY CONTAIN                   {aACertificate |
                                attributeCertificateRevocationList |
                                attributeAuthorityRevocationList}
  ID                          id-oc-pmiAA
  }


pmiSOA OBJECT-CLASS ::= {  -- a PMI Source of Authority
  SUBCLASS OF                 {top}
  KIND                   auxiliary
  MAY CONTAIN                 {attributeCertificateRevocationList |
                                attributeAuthorityRevocationList |
                                attributeDescriptorCertificate}
  ID                          id-oc-pmiSOA
  }


attCertCRLDistributionPt       OBJECT-CLASS ::= {
  SUBCLASS OF                        {top}
  KIND                        auxiliary
  MAY CONTAIN                        { attributeCertificateRevocationList |
                                     attributeAuthorityRevocationList }
  ID                                 id-oc-attCertCRLDistributionPts
  }


pmiDelegationPath              OBJECT-CLASS            ::= {
  SUBCLASS OF                        {top}
  KIND                        auxiliary
  MAY CONTAIN                        { delegationPath }
  ID                                 id-oc-pmiDelegationPath }


privilegePolicy                       OBJECT-CLASS      ::= {
  SUBCLASS OF                        {top}
  KIND                        auxiliary
  MAY CONTAIN                        {privPolicy }
  ID                                 id-oc-privilegePolicy }


-- Protected Privilege Object Class added 9/14/2003


protectedPrivilegePolicy       OBJECT-CLASS ::= {
          SUBCLASS OF                   {top}
          KIND                                  auxiliary
          MAY CONTAIN                   {ProtPrivPolicy }

          ID                                    id-oc-protectedPrivilegePolicy }


-- PMI directory attributes --

attributeCertificateAttribute                ATTRIBUTE  ::= {
  WITH SYNTAX                                          AttributeCertificate
  EQUALITY MATCHING RULE                    attributeCertificateExactMatch
  ID                                                   id-at-attributeCertificate }


aACertificate                                           ATTRIBUTE    ::=       {
  WITH SYNTAX                                          AttributeCertificate
  EQUALITY MATCHING RULE                    attributeCertificateExactMatch
  ID                                                   id-at-aACertificate }
```

```
attributeDescriptorCertificate              ATTRIBUTE  ::= {
  WITH SYNTAX                                           AttributeCertificate
  EQUALITY MATCHING RULE                    attributeCertificateExactMatch
  ID                                                    id-at-attributeDescriptorCertificate
}


attributeCertificateRevocationList   ATTRIBUTE ::= {
  WITH SYNTAX                                           CertificateList
  EQUALITY MATCHING RULE                    certificateListExactMatch
  ID                                                    id-at-
attributeCertificateRevocationList}


attributeAuthorityRevocationList    ATTRIBUTE            ::= {
  WITH SYNTAX                                           CertificateList
  EQUALITY MATCHING RULE                    certificateListExactMatch
  ID                                                    id-at-
attributeAuthorityRevocationList }


delegationPath                  ATTRIBUTE     ::= {
  WITH SYNTAX                   AttCertPath
  ID                            id-at-delegationPath }

AttCertPath    ::=      SEQUENCE OF AttributeCertificate


privPolicy                      ATTRIBUTE     ::= {
  WITH SYNTAX                   PolicySyntax
  ID                            id-at-privPolicy }

-- Protected Privilege Policy & XML Protected Privilege attributes added 9/14/2003


  protPrivPolicy        ATTRIBUTE    ::= {

        WITH SYNTAX                             AttributeCertificate

        EQUALITY MATCHING RULE                  attributeCertificateExactMatch
        ID                                      id-at-protPrivPolicy }


xmlPrivPolicy ATTRIBUTE     ::= {

        WITH SYNTAX             UTF8String –contains XML-encoded privilege policy information

        ID                      id-at-xMLPrivPolicy }


     noAssertion EXTENSION ::= {

     WITH SYNTAX            NULL

     ID                { id-ce-noAssertion }}


--Attribute certificate extensions and matching rules --


attributeCertificateExactMatch MATCHING-RULE  ::= {
  SYNTAX               AttributeCertificateExactAssertion
  ID                   id-mr-attributeCertificateExactMatch }
```

```
AttributeCertificateExactAssertion  ::=        SEQUENCE {
  serialNumber                    CertificateSerialNumber,
  issuer                 AttCertIssuer
  }


attributeCertificateMatch  MATCHING-RULE  ::=  {
  SYNTAX                 AttributeCertificateAssertion
  ID                     id-mr-attributeCertificateMatch  }


AttributeCertificateAssertion  ::=  SEQUENCE  {
  holder                 [0]        CHOICE {
        baseCertificateID      [0]  IssuerSerial,
        holderName                         [1]  GeneralNames} OPTIONAL,
  issuer                 [1]        GeneralNames OPTIONAL,
  attCertValidity                   [2]      GeneralizedTime OPTIONAL,
  attType                [3]        SET OF AttributeType OPTIONAL}
```

-- At least one component of the sequence shall be present

```
holderIssuerMatch MATCHING-RULE  ::= {
  SYNTAX                 HolderIssuerAssertion
  ID                     id-mr-holderIssuerMatch }

HolderIssuerAssertion           ::=      SEQUENCE {
  holder          [0]               Holder            OPTIONAL,
  issuer          [1]               AttCertIssuer   OPTIONAL
  }


delegationPathMatch MATCHING-RULE  ::= {
  SYNTAX                 DelMatchSyntax
  ID                     id-mr-delegationPathMatch }


DelMatchSyntax          ::=   SEQUENCE {
  firstIssuer            AttCertIssuer,
  lastHolder             Holder }


sOAIdentifier EXTENSION  ::= {
  SYNTAX                            NULL
  IDENTIFIED BY          id-ce-sOAIdentifier }


authorityAttributeIdentifier EXTENSION  ::=
  {
  SYNTAX                            AuthorityAttributeIdentifierSyntax
  IDENTIFIED BY          { id-ce-authorityAttributeIdentifier } }


AuthorityAttributeIdentifierSyntax  ::=        SEQUENCE SIZE (1..MAX) OF AuthAttId


AuthAttId ::=    IssuerSerial

authAttIdMatch MATCHING-RULE  ::= {
  SYNTAX                 AuthorityAttributeIdentifierSyntax
  ID                     id-mr-authAttIdMatch }
```

```
roleSpecCertIdentifier EXTENSION ::=
  {
  SYNTAX                    RoleSpecCertIdentifierSyntax
  IDENTIFIED BY      { id-ce-roleSpecCertIdentifier } }

RoleSpecCertIdentifierSyntax  ::= SEQUENCE SIZE (1..MAX) OF RoleSpecCertIdentifier


RoleSpecCertIdentifier  ::= SEQUENCE {
  roleName                                [0]                GeneralName,
  roleCertIssuer                          [1]                GeneralName,
  roleCertSerialNumber             [2]                 CertificateSerialNumber
  OPTIONAL,
  roleCertLocator                  [3]                 GeneralNames
  OPTIONAL }


roleSpecCertIdMatch MATCHING-RULE  ::= {
  SYNTAX             RoleSpecCertIdentifierSyntax
  ID                 id-mr-roleSpecCertIdMatch }


basicAttConstraints EXTENSION ::=
  {
         SYNTAX                    BasicAttConstraintsSyntax
         IDENTIFIED BY             { id-ce-basicAttConstraints }
  }

BasicAttConstraintsSyntax ::= SEQUENCE
  {
         authority                 BOOLEAN DEFAULT FALSE,
         pathLenConstraint    INTEGER (0..MAX) OPTIONAL
  }


basicAttConstraintsMatch MATCHING-RULE  ::= {
  SYNTAX             BasicAttConstraintsSyntax
  ID                 id-mr-basicAttConstraintsMatch }


delegatedNameConstraints EXTENSION ::= {
  SYNTAX                    NameConstraintsSyntax
  IDENTIFIED BY              id-ce-delegatedNameConstraints }


delegatedNameConstraintsMatch MATCHING-RULE  ::= {
  SYNTAX             NameConstraintsSyntax
  ID                 id-mr-delegatedNameConstraintsMatch}


timeSpecification EXTENSION  ::=  {
  SYNTAX                    TimeSpecification
  IDENTIFIED BY      id-ce-timeSpecification }


timeSpecificationMatch MATCHING-RULE  ::= {
  SYNTAX             TimeSpecification
  ID                 id-mr-timeSpecMatch }
```

```
acceptableCertPolicies EXTENSION ::= {
  SYNTAX                          AcceptableCertPoliciesSyntax
  IDENTIFIED BY                   id-ce-acceptableCertPolicies }

AcceptableCertPoliciesSyntax ::= SEQUENCE SIZE (1..MAX) OF CertPolicyId

CertPolicyId ::= OBJECT IDENTIFIER


acceptableCertPoliciesMatch MATCHING-RULE  ::= {
  SYNTAX               AcceptableCertPoliciesSyntax
  ID                   id-mr-acceptableCertPoliciesMatch }

attributeDescriptor EXTENSION  ::= {
  SYNTAX                          AttributeDescriptorSyntax
  IDENTIFIED BY                   {id-ce-attributeDescriptor } }


AttributeDescriptorSyntax  ::= SEQUENCE {
  identifier                                        AttributeIdentifier,
  attributeSyntax                          OCTET STRING (SIZE(1..MAX)),
  name                 [0]                 AttributeName   OPTIONAL,
  description          [1]                 AttributeDescription    OPTIONAL,
  dominationRule                           PrivilegePolicyIdentifier}


AttributeIdentifier  ::= ATTRIBUTE.&id({AttributeIDs})

AttributeIDs ATTRIBUTE  ::= {...}

AttributeName  ::= UTF8String(SIZE(1..MAX))

AttributeDescription  ::= UTF8String(SIZE(1..MAX))


PrivilegePolicyIdentifier        ::=      SEQUENCE {
  privilegePolicy                                   PrivilegePolicy,
  privPolSyntax                                     InfoSyntax }


attDescriptor MATCHING-RULE  ::= {
  SYNTAX               AttributeDescriptorSyntax
  ID                   id-mr-attDescriptorMatch }


userNotice  EXTENSION  ::= {
  SYNTAX                          SEQUENCE SIZE (1..MAX) OF UserNotice
  IDENTIFIED BY                   id-ce-userNotice }


targetingInformation  EXTENSION  ::= {
  SYNTAX                          SEQUENCE SIZE (1..MAX) OF Targets
  IDENTIFIED BY        id-ce-targetInformation }


Targets::=       SEQUENCE SIZE (1..MAX) OF Target


Target  ::=      CHOICE {
  targetName           [0]                 GeneralName,
  targetGroup          [1]                 GeneralName,
  targetCert                      [2]               TargetCert }
```

```
TargetCert      ::=      SEQUENCE {
  targetCertificate      IssuerSerial,
  targetName             GeneralName OPTIONAL,
  certDigestInfo                  ObjectDigestInfo OPTIONAL }


noRevAvail EXTENSION ::=  {
  SYNTAX                          NULL
  IDENTIFIED BY          id-ce-noRevAvail }


acceptablePrivilegePolicies EXTENSION ::= {
  SYNTAX                          AcceptablePrivilegePoliciesSyntax
  IDENTIFIED BY          id-ce-acceptablePrivilegePolicies }


AcceptablePrivilegePoliciesSyntax ::=         SEQUENCE SIZE (1..MAX) OF PrivilegePolicy



-- Indirect issuer extension & matching rule added 9/14/2003


indirectIssuer EXTENSION ::=        {

  SYNTAX                 BOOLEAN

  ID                     id-ce-indirectIssuer  }


indirectIssuerMatch MATCHING-RULE ::= {

SYNTAX  BOOLEAN

   ID     id-mr-indirectIssuerMatch }


  issuedOnBehalfOf EXTENSION ::= {

      SYNTAX GeneralName

      ID id-ce-issuedOnBehalfOf }

-- SOA identifier matching rule added 9/14/2003


sOAIdentifierMatch MATCHING-RULE  ::= {

  SYNTAX      NULL

  ID          id-mr-sOAIdentifierMatch }



-- object identifier assignments --


-- object classes --


id-oc-pmiUser                      OBJECT IDENTIFIER ::=      {id-oc 24}
id-oc-pmiAA                                 OBJECT IDENTIFIER ::=      {id-oc 25}
id-oc-pmiSOA                       OBJECT IDENTIFIER ::=      {id-oc 26}
```

```
id-oc-attCertCRLDistributionPts              OBJECT IDENTIFIER ::=     {id-oc 27}
id-oc-privilegePolicy                        OBJECT IDENTIFIER ::=     {id-oc 32}
id-oc-pmiDelegationPath                      OBJECT IDENTIFIER ::=     {id-oc 33}

id-oc-protectedPrivilegePolicy             OBJECT IDENTIFIER ::=     {id-oc 34}



-- directory attributes --

id-at-attributeDescriptorCertificate         OBJECT IDENTIFIER ::=     {id-at 62}
id-at-privPolicy                             OBJECT IDENTIFIER ::=     {id-at 71}
id-at-role                                   OBJECT IDENTIFIER ::=     {id-at 72}
id-at-delegationPath                         OBJECT IDENTIFIER ::=     {id-at 73}

id-at-xMLProtPrivPolicy                    OBJECT IDENTIFIER   ::=   {id-at 74}

id-at-xMLPrivilegeInfo                     OBJECT IDENTIFIER   ::=     {id-at 75}



--attribute certificate extensions --

id-ce-authorityAttributeIdentifier           OBJECT IDENTIFIER ::=     {id-ce 38}
id-ce-roleSpecCertIdentifier                 OBJECT IDENTIFIER ::=     {id-ce 39}
id-ce-basicAttConstraints                    OBJECT IDENTIFIER ::=     {id-ce 41}
id-ce-delegatedNameConstraints               OBJECT IDENTIFIER ::=     {id-ce 42}
id-ce-timeSpecification                      OBJECT IDENTIFIER ::=     {id-ce 43}
id-ce-attributeDescriptor                    OBJECT IDENTIFIER ::=     {id-ce 48}
id-ce-userNotice                             OBJECT IDENTIFIER ::=     {id-ce 49}
id-ce-sOAIdentifier                          OBJECT IDENTIFIER ::=     {id-ce 50}
id-ce-acceptableCertPolicies          OBJECT IDENTIFIER ::=     {id-ce 52}
id-ce-targetInformation                      OBJECT IDENTIFIER ::=     {id-ce 55}
id-ce-noRevAvail                             OBJECT IDENTIFIER ::=     {id-ce 56}
id-ce-acceptablePrivilegePolicies            OBJECT IDENTIFIER ::=     {id-ce 57}

id-ce-indirectIssuer                       OBJECT IDENTIFIER ::=     {id-ce 61}

id-ce-noAssertion                            OBJECT IDENTIFIER ::=     {id-ce 62}


-- PMI matching rules --

id-mr-attributeCertificateMatch              OBJECT IDENTIFIER ::=     {id-mr 42}
id-mr-attributeCertificateExactMatch         OBJECT IDENTIFIER ::=     {id-mr 45}
id-mr-holderIssuerMatch                      OBJECT IDENTIFIER ::=     {id-mr 46}
id-mr-authAttIdMatch                  OBJECT IDENTIFIER ::=     {id-mr 53}
id-mr-roleSpecCertIdMatch                    OBJECT IDENTIFIER ::=     {id-mr 54}
id-mr-basicAttConstraintsMatch               OBJECT IDENTIFIER ::=     {id-mr 55}
id-mr-delegatedNameConstraintsMatch          OBJECT IDENTIFIER ::=     {id-mr 56}
id-mr-timeSpecMatch                          OBJECT IDENTIFIER ::=     {id-mr 57}
id-mr-attDescriptorMatch                     OBJECT IDENTIFIER ::=     {id-mr 58}
id-mr-acceptableCertPoliciesMatch            OBJECT IDENTIFIER ::=     {id-mr 59}
id-mr-delegationPathMatch                    OBJECT IDENTIFIER ::=     {id-mr 61}

id-mr-sOAIdentifierMatch                    OBJECT IDENTIFIER     ::=  {id-mr 66}

id-mr-indirectIssuerMatch                  OBJECT IDENTIFIER     ::= {id-mr 67}


END
```