Recommendation X.509 (2000 edition) | ISO/IEC 9594-8:2000

Draft Technical Corrigendum 10

(covering resolution to defect report 314)

This corrects the defects reported in defect report 314

<u>Replace subclause 8.4.2.2 with the following:</u>

8.4.2.2 Name constraints extension

This field, which shall be used only in a CA-certificate, indicates a name space within which all subject names in subsequent certificates in a certification path must be located. This field is defined as follows:

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

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

If present, the **permittedSubtrees** and **excludedSubtrees** components each specify one or more naming subtrees, each defined by the name of the root of the subtree and optionally, within that subtree, an area that is bounded by upper and/or lower levels. If **permittedSubtrees** is present, subject names within these subtrees are acceptable. If **excludedSubtrees** is present, any certificate issued by the subject CA or subsequent CAs in the certification path that has a subject name within these subtrees is unacceptable. If both **permittedSubtrees** and **excludedSubtrees** are present and the name spaces overlap, the exclusion statement takes precedence for names within that overlap. If neither permitted nor excluded subtrees are specified for a name form, then any name within that name form is acceptable.

If **permittedSubtrees** is present, the following applies to all subsequent certificates in the path. If any certificate contains a subject name (in the **subject** field or **subjectAltNames** extension) of a name form for which permitted subtrees are specified, the name must fall within at least one of the specified subtrees. If any certificate contains only subject names of name forms other than those for which permittee subtrees are specified, the subject names are not required to fall within any of the specified subtrees. For example, assume that two permitted subtrees are specified, one for the DN name form and one for the rfc822 name form and no excluded subtrees are specified. A certificate that only contained a DN and where the DN is within the specified permitted subtree, would be acceptable. A certificate that contained both a DN and an rfc822 name and where only one of them is within its specified permitted subtree, would be unacceptable. A certificate that contained only names other than a DN or rfc822 name would also be acceptable.

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.

If **excludedSubtrees** is present, any certificate issued by the subject CA or subsequent CAs in the certification path that has a subject name (in the **subject** field or **subjectAltNames** extension) within these subtrees is unacceptable. For example, assume that two excluded subtrees are specified, one for the DN name form and one for the rfc822 name form. A certificate that only contained a DN and where the DN is within the specified excluded subtree, would be unacceptable. A certificate that contained both a DN and an rfc822 name and where at least one of them is within its specified excluded subtree, would be unacceptable.

When a certificate subject has multiple names of the same name form (including, in the case of the **directoryName** name form, the name in the subject field of the certificate if non-null) then all such names shall be tested for consistency with a name constraint of that name form.

Of the name forms available through the **GeneralName** type, only those name forms that have a well-defined hierarchical structure may be used in the **permittedSubtrees** and **excludedSubtrees** fields. The **directoryName** name form satisfies this requirement; when using this name form a naming subtree corresponds to a DIT subtree.

The **minimum** field specifies the upper bound of the area within the subtree. All names whose final name component is above the level specified are not contained within the area. A value of **minimum** equal to zero (the default) corresponds to the base, i.e. the top node of the subtree. For example, if **minimum** is set to one, then the naming subtree excludes the base node but includes subordinate nodes.

The **maximum** field specifies the lower bound of the area within the subtree. All names whose last component is below the level specified are not contained within the area. A value of **maximum** of zero corresponds to the base, i.e. the top of the subtree. An absent **maximum** component indicates that no lower limit should be imposed on the area within the subtree. For example, if **maximum** is set to one, then the naming subtree excludes all nodes except the subtree base and its immediate subordinates.

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.

This extension may, at the option of the certificate issuer, be either critical or non-critical. It is recommended that it be flagged critical, otherwise a certificate user may not check that subsequent certificates in a certification path are located in the name space intended by the issuing CA.

Conformant implementations are not required to recognize all possible name forms.

If the extension is present and is flagged critical, a certificate-using implementation must recognize and process all name forms for which there is both a subtree specification (permitted or excluded) in the extension and a corresponding value in the **subject** field or **subjectAltNames** extension of any subsequent certificate in the certification path. If an unrecognized name form appears in both a subtree specification and a subsequent certificate, that certificate shall be handled as if an unrecognized critical extension was encountered. If any subject name in the certificate falls within an excluded subtree, the certificate is unacceptable. If a subtree is specified for a name form that is not contained in any subsequent certificate, that subtree can be ignored.

If the extension is present and is flagged non-critical and a certificate-using implementation does not recognize a name form used in any **base** component, then that subtree specification may be ignored.

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.

Replace subclause 10.5.2 with the following:

10.5.2 Processing intermediate certificates

For an intermediate certificate, the following constraint recording actions are then performed, in order to correctly set up the state variables for the processing of the next certificate. Self-signed certificates, if encountered in the path, are ignored.

a) If the **nameConstraints** extension with a **permittedSubtrees** component is present in the certificate, set the *permitted-subtrees* state variable to the intersection of its previous value and the value indicated in the certificate extension.

b) If the **nameConstraints** extension with an **excludedSubtrees** component is present in the certificate, set the *excluded-subtrees* state variable to the union of its previous value and the value indicated in the certificate extension.

c) If *policy-mapping-inhibit-indicator* is set:

– process any policy mappings extension by, for each mapping identified in the extension, locate all rows in the *authorities-constrained-policy-set* table whose [*path-depth*] column entry is equal to the issuer domain policy value in the extension and delete the row.

d) If *policy-mapping-inhibit-indicator* is not set:

– process any policy mappings extension by, for each mapping identified in the extension, locate all rows in the *authorities-constrained-policy-set* table whose [*path-depth*] column entry is equal to the issuer domain policy value in the extension, and write the subject domain policy value from the extension in the [*path-depth*+1] column entry of the same row. If the extension maps an issuer domain policy to more than one subject domain policy, then the affected row is copied and the new entry added to each row. If the value in *authorities-constrained-policy-set*[0, *path-depth*] is *any-policy*, then write each issuer domain policy identifier from the policy mappings extension in the [*path-depth*] column, making duplicate rows as necessary and retaining qualifiers if they are present, and write the subject domain policy value from the extension in the [*path-depth*+1] column entry of the same row.

– if the *policy-mapping-inhibit-pending* indicator is set and the certificate is not self-issued, decrement the corresponding *skip-certificates* value and, if this value becomes zero, set the *policy-mapping-inhibit-indicator*.

– If the **inhibitPolicyMapping** constraint is present in the certificate, perform the following. For a **SkipCerts** value of 0, set the *policy-mapping-inhibit-indicator*. For any other **SkipCerts** value, set the *policy-mapping-inhibit-pending* indicator, and set the corresponding *skip-certificates* value to the lesser of the **SkipCerts** value and the previous *skip-certificates* value (if the *policy-mapping-inhibit-pending* indicator was already set).

e) For any row not modified in either step c) or d), above (and every row in the case that there is no mapping extension present in the certificate), write the policy identifier from [*path-depth*] column in the [*path-depth*+1] column of the row.

f) If *inhibit-any-policy-indicator* is not set:

– If the *inhibit-any-policy-pending* indicator is set and the certificate is not self-issued, decrement the corresponding *skip-certificates* value and, if this value becomes zero, set the *inhibit-any-policy-indicator*.

- If the **inhibitAnyPolicy** constraint is prensent in the certificate, perform the following. For a **SkipCerts** value of 0, set the *inhibit-any-policy-indicator*. For any other **SkipCerts** value, set the *inhibit-any-policy-pending* indicator, and set the corresponding *skip-certificates* value to the lesser of the **SkipCerts** value and the previous *skip-certificates* value (if the *inhibit-any-policy-pending* indicator was already set).

g) Increment [*path-depth*].

In Annex A, subclause A.1 replace:

```
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))
```

with:

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

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

In Annex A, subclause A.1 replace:

```
id-ce-nameConstraint          OBJECT IDENTIFIER   ::=   {id-ce 30 1}
```

with:

```
id-ce-nameConstraints         OBJECT IDENTIFIER   ::=   {id-ce 30}
```