INTERNATIONAL STANDARD

ISO/IEC 9594-6

Third edition
1998-12-15

Information technology — Open Systems Interconnection — The Directory: Selected attribute types

Technologies de l'information — Interconnexion de systèmes ouverts (OSI) — L'Annuaire: Types d'attributs sélectionnés

Reference number
ISO/IEC 9594-6:1998(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Published by ISO in 2000

Printed in Switzerland

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 9594 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 9594-6 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 6, *Telecommunications and information exchange between systems*, in collaboration with ITU-T. The identical text is published as ITU-T Recommendation X.520.

This third edition cancels and replaces the second edition (ISO/IEC 9594-6:1995), of which it constitutes a minor revision.

ISO/IEC 9594 consists of the following parts, under the general title *Information technology — Open Systems Interconnection — The Directory*:

— *Part 1: Overview of concepts, models and services*

— *Part 2: Models*

— *Part 3: Abstract service definition*

— *Part 4: Procedures for distributed operation*

— *Part 5: Protocol specifications*

— *Part 6: Selected attribute types*

— *Part 7: Selected object classes*

— *Part 8: Authentication framework*

— *Part 9: Replication*

— *Part 10: Use of systems management for administration of the Directory*

Annex A forms a normative part of this part of ISO/IEC 9594. Annexes B to E are for information only.

Introduction

This Recommendation | International Standard, together with other Recommendations | International Standards, has been produced to facilitate the interconnection of information processing systems to provide directory services. A set of such systems, together with the directory information which they hold, can be viewed as an integrated whole, called the *Directory*. The information held by the Directory, collectively known as the Directory Information Base (DIB), is typically used to facilitate communication between, with or about objects such as application entities, people, terminals, and distribution lists.

The Directory plays a significant role in Open Systems Interconnection, whose aim is to allow, with a minimum of technical agreement outside of the interconnection standards themselves, the interconnection of information processing systems:

- from different manufacturers;
- under different managements;
- of different levels of complexity; and
- of different ages.

This Recommendation | International Standard defines a number of attribute types which may be found useful across a range of applications of the Directory, as well as a number of standard attribute syntaxes and matching rules. One particular use for many of the attributes defined herein is in the formation of names, particularly for the classes of object defined in ITU-T Rec. X.521 | ISO/IEC 9594-7.

This third edition technically revises and enhances, but does not replace, the second edition of this Recommendation | International Standard. Implementations may still claim conformance to the second edition. However, at some point, the second edition will not be supported (i.e. reported defects will no longer be resolved). It is recommended that implementations conform to this third edition as soon as possible.

This third edition specifies version 1 and version 2 of the Directory protocols.

The first and second editions also specified version 1. Most of the services and protocols specified in this edition are designed to function under version 1. When version 1 has been negotiated differences between the services and between the protocols defined in the three editions are accommodated using the rules of extensibility defined in this edition of ITU-T Rec. X.519 | ISO/IEC 9594-5. However some enhanced services and protocols, e.g. signed errors, will not function unless all Directory entities involved in the operation have negotiated version 2.

Implementors should note that a defect resolution process exists and that corrections may be applied to this part of this International Standard in the form of technical corrigenda. The identical corrections will be applied to this Recommendation in the form of corrigenda and/or an Implementor's Guide. A list of approved technical corrigenda for this part of this International Standard can be obtained from the subcommittee secretariat. Published technical corrigenda are available from your national standards organization. The ITU-T corrigenda and Implementor's Guides may be obtained from the ITU Web site.

Annex A, which is an integral part of this Recommendation | International Standard, provides the ASN.1 notation for the complete module which defines the attributes, attribute syntaxes, and matching rules.

Annex B, which is not an integral part of this Recommendation | International Standard, provides a table of attribute types, for easy reference.

Annex C, which is not an integral part of this Recommendation | International Standard, provides suggested upper bounds value constraints used in these Directory Specifications.

Annex D, which is not an integral part of this Recommendation | International Standard, lists alphabetically the attributes and matching rules defined in this Directory Specification.

Annex E, which is not an integral part of this Recommendation | International Standard, lists the amendments and defect reports that have been incorporated to form this edition of this Recommendation | International Standard.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – THE DIRECTORY: SELECTED ATTRIBUTE TYPES

SECTION 1 – GENERAL

1 Scope

This Recommendation | International Standard defines a number of attribute types and matching rules which may be found useful across a range of applications of the Directory.

Attribute types and matching rules fall into three categories, as described below.

Some attribute types and matching rules are used by a wide variety of applications or are understood and/or used by the Directory itself.

NOTE – It is recommended that an attribute type or matching rule defined in this document be used, in preference to the generation of a new one, whenever it is appropriate for the application.

Some attribute types and matching rules are internationally standardized, but are application-specific. These are defined in the standards associated with the application concerned.

Any administrative authority can define its own attribute types and matching rules for any purpose. These are not internationally standardized, and are available to others beyond the administrative authority which created them only by bilateral agreement.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.1 Identical Recommendations | International Standards

- ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model.*
- ITU-T Recommendation X.500 (1997) | ISO/IEC 9594-1:1998, *Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services.*
- ITU-T Recommendation X.501 (1997) | ISO/IEC 9594-2:1998, *Information technology – Open Systems Interconnection – The Directory: Models.*
- ITU-T Recommendation X.509 (1997) | ISO/IEC 9594-8:1998, *Information technology – Open Systems Interconnection – The Directory: authentication framework.*
- ITU-T Recommendation X.511 (1997) | ISO/IEC 9594-3:1998, *Information technology – Open Systems Interconnection – The Directory: Abstract service definition.*
- ITU-T Recommendation X.518 (1997) | ISO/IEC 9594-4:1998, *Information technology – Open Systems Interconnection – The Directory: Procedures for distributed operation.*

- ITU-T Recommendation X.519 (1997) | ISO/IEC 9594-5:1998, *Information technology – Open Systems Interconnection – The Directory: Protocol specifications.*
- ITU-T Recommendation X.521 (1997) | ISO/IEC 9594-7:1998, *Information technology – Open Systems Interconnection – The Directory: Selected object classes.*
- ITU-T Recommendation X.525 (1997) | ISO/IEC 9594-9:1998, *Information technology – Open Systems Interconnection – The Directory: Replication.*
- ITU-T Recommendation X.530 (1997) | ISO/IEC 9594-10:1998, *Information technology – Open Systems Interconnection – The Directory: Use of system management for Administration of the Directory.*
- ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ITU-T Recommendation X.681 (1997) | ISO/IEC 8824-2:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*
- ITU-T Recommendation X.682 (1997) | ISO/IEC 8824-3:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*
- ITU-T Recommendation X.683 (1997) | ISO/IEC 8824-4:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Parametrization of ASN.1 specifications.*

2.2 Other references

- CCITT Recommendation E.123 (1988), *Notation for National and International Telephone numbers.*
- ITU-T Recommendation E.164 (1997), *The international public telecommunication numbering plan.*
- ITU-T Recommendation F.1 (1998), *Operational provisions for the international public telegram service.*
- CCITT Recommendation F.31 (1988), *Telegram retransmission system.*
- CCITT Recommendation F.401 (1992), *Naming and addressing for public message handling services.*
- ITU-T Recommendation T.30 (1996), *Procedures for document facsimile transmission in the general switched telephone network.*
- ITU-T Recommendation T.62 (1993), *Control procedures for teletex and Group 4 facsimile services.*
- ITU-T Recommendation X.121 (1996), *International numbering plan for public data networks.*
- ISO 3166 (all parts), *Codes for the representation of names of countries and their subdivisions.*
- ISO 639-2:1998, *Codes for the representation of names of languages – Part 2: Alpha 3 code*.
- ISO/IEC 9945-2:1993 *Information technology – Portable Operating System Interface (POSIX) – Part 2: Shell and Utilities.*

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

The following terms are defined in ITU-T Rec. X.501 | ISO/IEC 9594-2:

a) *attribute type;*

b) *object class;*

c) *matching rule.*

4 Conventions

With minor exceptions, this Directory Specification has been prepared according to the "Presentation of ITU-T | ISO/IEC common text" guidelines in the Guide for ITU-T and ISO/IEC JTC 1 Cooperation.

The term "Directory Specification" (as in "this Directory Specification") shall be taken to mean ITU-T Rec. X.520 | ISO/IEC 9594-6. The term "Directory Specifications" shall be taken to mean the X.500-series Recommendations | parts of ISO/IEC 9594.

This Directory Specification uses the term "1988 edition systems" to refer to systems conforming to the first (1988) edition of the Directory Specifications, i.e. the 1988 edition of the series of CCITT X.500 Recommendations and the ISO/IEC 9594:1990 edition. This Directory Specification uses the term "1993 edition systems" to refer to systems conforming to the second (1993) edition of the Directory Specifications, i.e. the 1993 edition of the series of ITU-T X.500 Recommendations and the ISO/IEC 9594:1995 edition. Systems conforming to this third edition of the Directory Specifications are referred to as "1997 edition systems".

This Directory Specification presents ASN.1 notation in the bold Helvetica typeface. When ASN.1 types and values are referenced in normal text, they are differentiated from normal text by presenting them in the bold Helvetica typeface. The names of procedures, typically referenced when specifying the semantics of processing, are differentiated from normal text by displaying them in bold Times. Access control permissions are presented in italicized Times.

Attribute types and matching rules are defined in this Recommendation | International Standard by use of the **ATTRIBUTE** and **MATCHING-RULE** information object classes defined in ITU-T Rec. X.501 | ISO/IEC 9594-2.

Examples of the use of the attribute types are described using an informal notation, where attribute type and value pairs are represented by an acronym for the attribute type, followed by an equals sign ("="), followed by the example value for the attribute.

SECTION 2 – SELECTED ATTRIBUTE TYPES

5 Definition of selected attribute types

This Directory Specification defines a number of attribute types which may be found useful across a range of applications of the Directory.

Many of the attributes defined in this Specification are based on a common ASN.1 syntax:

```
DirectoryString { INTEGER : maxSize } ::= CHOICE {
  teletexString          TeletexString (SIZE (1..maxSize)),
  printableString        PrintableString (SIZE (1..maxSize)),
  bmpString              BMPString (SIZE (1..maxSize)),
  universalString        UniversalString (SIZE (1..maxSize)) }
```

Some implementations of the Directory do not support **BMPString** or **UniversalString**, and will not be able to generate, match, or display attributes having such a syntax.

5.1 System attribute types

5.1.1 Knowledge Information

The *Knowledge Information* attribute type specifies a human readable accumulated description of knowledge mastered by a specific DSA.

NOTE – This attribute is now obsolete.

```
knowledgeInformation ATTRIBUTE          ::=          {
  WITH SYNTAX                           DirectoryString {ub-knowledge-information}
  EQUALITY MATCHING RULE                caseIgnoreMatch
  ID                                    id-at-knowledgeInformation }
```

5.2 Labelling attribute types

These attributes type are concerned with information about objects which has been explicitly associated with the objects by a labelling process.

5.2.1 Name

The *Name* attribute type is the attribute supertype from which string attribute types typically used for naming may be formed.

```
name ATTRIBUTE                          ::=     {
  WITH SYNTAX                                   DirectoryString {ub-name}
  EQUALITY MATCHING RULE                        caseIgnoreMatch
  SUBSTRINGS MATCHING RULE                      caseIgnoreSubstringsMatch
  ID                                            id-at-name }
```

5.2.2 Common Name

The *Common Name* attribute type specifies an identifier of an object. A Common Name is not a directory name; it is a (possibly ambiguous) name by which the object is commonly known in some limited scope (such as an organization) and conforms to the naming conventions of the country or culture with which it is associated.

An attribute value for common name is a string chosen either by the person or organization it describes or the organization responsible for the object it describes for devices and application entities. For example, a typical name of a person in an English-speaking country comprises a personal title (e.g. Mr., Ms., Rd, Professor, Sir, Lord), a first name, middle name(s), last name, generation qualifier (if any, e.g. Jr.) and decorations and awards (if any, e.g. QC).

Examples:

CN = "Mr. Robin Lachlan McLeod BSc(Hons) CEng MIEE";

CN = "Divisional Coordination Committee";

CN = "High Speed Modem".

Any variants should be associated with the named object as separate and alternative attribute values.

Other common variants should also be admitted, e.g. use of a middle name as a preferred first name; use of "Bill" in place of "William", etc.

```
commonName ATTRIBUTE        ::=  {
  SUBTYPE OF                     name
  WITH SYNTAX                    DirectoryString {ub-common-name}
  ID                             id-at-commonName }
```

5.2.3 Surname

The *Surname* attribute type specifies the linguistic construct which normally is inherited by an individual from the individual's parent or assumed by marriage, and by which the individual is commonly known.

An attribute value for Surname is a string, e.g. "McLeod".

```
surname ATTRIBUTE           ::=  {
    SUBTYPE OF                     name
    WITH SYNTAX                    DirectoryString {ub-surname}
    ID                             id-at-surname }
```

5.2.4 Given Name

The *Given Name* attribute type specifies the linguistic construct which is normally given to an individual by the individual's parent, or is chosen by the individual, or by which the individual is commonly known.

An attribute value for Given Name is a string, e.g. "David", or "Jean Paul".

```
givenName ATTRIBUTE         ::=  {
    SUBTYPE OF                     name
    WITH SYNTAX                    DirectoryString {ub-name}
    ID                             id-at-givenName }
```

5.2.5 Initials

The *Initials* attribute type contains the initials of some or all of an individual's names, but not the surname(s).

An attribute value for Initials is a string, e.g. "D" or "D." or "J.P.".

```
initials ATTRIBUTE          ::=  {
    SUBTYPE OF                     name
    WITH SYNTAX                    DirectoryString {ub-name}
    ID                             id-at-initials }
```

5.2.6 Generation Qualifier

The *Generation Qualifier* attribute type contains a string which is used to provide generation information to qualify an individual's name.

An attribute value for Generation Qualifier is a string, e.g. "Jr." or "II".

```
generationQualifier ATTRIBUTE    ::=    {
    SUBTYPE OF                          name
    WITH SYNTAX                         DirectoryString {ub-name}
    ID                                  id-at-generationQualifier }
```

5.2.7 Unique Identifier

The *Unique Identifier* attribute type specifies an identifier which may be used to distinguish between object references when a distinguished name has been reused. It may be, for example, an encoded object identifier, certificate, date, timestamp, or some other form of certification on the validity of the distinguished name.

An attribute value for Unique Identifier is a bit string.

```
uniqueIdentifier ATTRIBUTE       ::=    {
    WITH SYNTAX                         UniqueIdentifier
    EQUALITY MATCHING RULE              bitStringMatch
    ID                                  id-at-uniqueIdentifier }
UniqueIdentifier                 ::=    BIT STRING
```

5.2.8 DN Qualifier

The *DN Qualifier* attribute type specifies disambiguating information to add to the relative distinguished name of an entry. It is intended to be used for entries held in multiple DSAs which would otherwise have the same name, and that its value be the same in a given DSA for all entries to which this information has been added.

```
dnQualifier ATTRIBUTE            ::=    {
    WITH SYNTAX                         PrintableString
    EQUALITY MATCHING RULE              caseIgnoreMatch
    ORDERING MATCHING RULE              caseIgnoreOrderingMatch
    SUBSTRINGS MATCHING RULE            caseIgnoreSubstringsMatch
    ID                                  id-at-dnQualifier }
```

5.2.9 Serial Number

The *Serial Number* attribute type specifies an identifier, the serial number of a device.

An attribute value for Serial Number is a printable string.

```
serialNumber ATTRIBUTE           ::=    {
    WITH SYNTAX                         PrintableString (SIZE (1..ub-serialNumber))
    EQUALITY MATCHING RULE              caseIgnoreMatch
    SUBSTRINGS MATCHING RULE            caseIgnoreSubstringsMatch
    ID                                  id-at-serial-number }
```

5.3 Geographical Attribute Types

These attribute types are concerned with geographical positions or regions with which objects are associated.

5.3.1 Country Name

The *Country Name* attribute type specifies a country. When used as a component of a directory name, it identifies the country in which the named object is physically located or with which it is associated in some other important way.

An attribute value for country name is a string chosen from ISO 3166.

```
countryName ATTRIBUTE ::= {
    SUBTYPE OF          name
    WITH SYNTAX         CountryName
    SINGLE VALUE        TRUE
    ID                  id-at-countryName }

CountryName ::= PrintableString (SIZE(2))  -- ISO 3166 codes only
```

5.3.2 Locality Name

The *Locality Name* attribute type specifies a locality. When used as a component of a directory name, it identifies a geographical area or locality in which the named object is physically located or with which it is associated in some other important way.

An attribute value for Locality Name is a string, e.g. L = "Edinburgh".

```
localityName ATTRIBUTE          ::=     {
    SUBTYPE OF                          name
    WITH SYNTAX                         DirectoryString {ub-locality-name}
    ID                                  id-at-localityName }
```

The *Collective Locality Name* attribute type specifies a locality name for a collection of entries.

```
collectiveLocalityName ATTRIBUTE  ::=     {
    SUBTYPE OF                          localityName
    COLLECTIVE                          TRUE
    ID                                  id-at-collectiveLocalityName }
```

5.3.3 State or Province Name

The *State or Province Name* attribute type specifies a state or province. When used as a component of a directory name, it identifies a geographical subdivision in which the named object is physically located or with which it is associated in some other important way.

An attribute value for State or Province Name is a string, e.g. S = "Ohio".

```
stateOrProvinceName ATTRIBUTE   ::=     {
    SUBTYPE OF                          name
    WITH SYNTAX                         DirectoryString {ub-state-name}
    ID                                  id-at-stateOrProvinceName }
```

The *Collective State or Province Name* attribute type specifies a state or province name for a collection of entries.

```
collectiveStateOrProvinceName ATTRIBUTE   ::=     {
    SUBTYPE OF                          stateOrProvinceName
    COLLECTIVE                          TRUE
    ID                                  id-at-collectiveStateOrProvinceName }
```

5.3.4 Street Address

The *Street Address* attribute type specifies a site for the local distribution and physical delivery in a postal address, i.e. the street name, place, avenue, and the house number. When used as a component of a directory name, it identifies the street address at which the named object is located or with which it is associated in some other important way.

An attribute value for Street Address is a string, e.g. "Arnulfstraße 60".

```
streetAddress ATTRIBUTE         ::=     {
    WITH SYNTAX                         DirectoryString {ub-street-address}
    EQUALITY MATCHING RULE              caseIgnoreMatch
    SUBSTRINGS MATCHING RULE                    caseIgnoreSubstringsMatch
    ID                                  id-at-streetAddress }
```

The *Collective Street Address* attribute type specifies a street address for a collection of entries.

```
collectiveStreetAddress ATTRIBUTE  ::=     {
    SUBTYPE OF                          streetAddress
    COLLECTIVE                          TRUE
    ID                                  id-at-collectiveStreetAddress }
```

5.3.5 House Identifier

The *House Identifier* attribute type specifies a linguistic construct used to identify a particular building, for example a house number or house name relative to a street, avenue, town or city, etc.

An attribute value for House Identifier is a string, e.g. "14".

```
houseIdentifier ATTRIBUTE       ::=     {
    WITH SYNTAX                         DirectoryString {ub-name}
    EQUALITY MATCHING RULE              caseIgnoreMatch
    SUBSTRINGS MATCHING RULE                    caseIgnoreSubstringsMatch
    ID                                  id-at-houseIdentifier }
```

5.4 Organizational attribute types

These attribute types are concerned with organizations and can be used to describe objects in terms of organizations with which they are associated.

5.4.1 OrganizationName

The *OrganizationName* attribute type specifies an organization. When used as a component of a directory name it identifies an organization with which the named object is affiliated.

An attribute value for OrganizationName is a string chosen by the organization (e.g. O = "Scottish Telecommunications plc"). Any variants should be associated with the named Organization as separate and alternative attribute values.

```
organizationName ATTRIBUTE      ::=    {
  SUBTYPE OF                           name
  WITH SYNTAX                          DirectoryString {ub-organization-name}
  ID                                   id-at-organizationName }
```

The *Collective Organization Name* attribute type specifies an organization name for a collection of entries.

```
collectiveOrganizationName ATTRIBUTE    ::=    {
  SUBTYPE OF                                   organizationName
  COLLECTIVE                                   TRUE
  ID                                           id-at-collectiveOrganizationName }
```

5.4.2 Organizational Unit Name

The *Organizational Unit Name* attribute type specifies an organizational unit. When used as a component of a directory name it identifies an organizational unit with which the named object is affiliated.

The designated organizational unit is understood to be part of an organization designated by an OrganizationName attribute. It follows that if an Organizational Unit Name attribute is used in a directory name, it must be associated with an OrganizationName attribute.

An attribute value for Organizational Unit Name is a string chosen by the organization of which it is part (e.g. OU = "Technology Division"). Note that the commonly used abbreviation "TD" would be a separate and alternative attribute value.

Example:

O = "Scottel", OU = "TD"

```
organizationalUnitName ATTRIBUTE    ::=    {
  SUBTYPE OF                               name
  WITH SYNTAX                              DirectoryString {ub-organizational-unit-name}
  ID                                       id-at-organizationalUnitName }
```

The *Collective Organizational Unit Name* attribute type specifies an organizational unit name for a collection of entries.

```
collectiveOrganizationalUnitName ATTRIBUTE    ::=    {
  SUBTYPE OF                                          organizationalUnitName
  COLLECTIVE                                          TRUE
  ID                                                  id-at-collectiveOrganizationalUnitName }
```

5.4.3 Title

The *Title* attribute type specifies the designated position or function of the object within an organization.

An attribute value for Title is a string.

Example:

T = "Manager, Distributed Applications"

```
title ATTRIBUTE      ::=    {
    SUBTYPE OF              name
    WITH SYNTAX             DirectoryString {ub-title}
    ID                      id-at-title }
```

5.5 Explanatory attribute types

These attribute types are concerned with explanations (e.g. in a natural language) of something about an object.

5.5.1 Description

The *Description* attribute type specifies text which describes the associated object.

For example, the object "Standards Interest" might have the associated description "distribution list for exchange of information about intra-company standards development".

An attribute value for Description is a string.

```
description ATTRIBUTE                ::=    {
  WITH SYNTAX                               DirectoryString {ub-description}
  EQUALITY MATCHING RULE                    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE                  caseIgnoreSubstringsMatch
  ID                                        id-at-description }
```

5.5.2 Search Guide

The *Search Guide* attribute type specifies information of suggested search criteria which may be included in some entries expected to be a convenient base-object for the search operation, e.g. country or organization.

Search criteria consist of an optional identifier for the type of object sought and combinations of attribute types and logical operators to be used in the construction of a filter. It is possible to specify for each search criteria item the matching level, e.g. approximate match.

The Search Guide attribute may recur to reflect the various types of requests, e.g. search for a Residential Person or an Organizational Person, which may be fulfilled from the given base-object where the Search Guide is read.

```
searchGuide ATTRIBUTE                     ::=        {
  WITH SYNTAX                                        Guide
  ID                                                 id-at-searchGuide  }

Guide                                               ::=        SET  {
  objectClass                       [0]              OBJECT-CLASS.&id OPTIONAL,
  criteria                          [1]              Criteria    }

Criteria                                            ::=        CHOICE  {
  type                              [0]              CriteriaItem,
  and                               [1]              SET OF Criteria,
  or                                [2]              SET OF Criteria,
  not                               [3]              Criteria  }

CriteriaItem                        ::=              CHOICE  {
  equality                          [0]              AttributeType,
  substrings                        [1]              AttributeType,
  greaterOrEqual                    [2]              AttributeType,
  lessOrEqual                       [3]              AttributeType,
  approximateMatch                  [4]              AttributeType  }
```

Example:

The following is a potential value of the Search Guide attribute that could be stored in entries of object class Locality to indicate how entries of object class Residential Person might be found:

```
residential-person-guide Guide ::= {
  objectClass residentialPerson.&id,
  criteria and : {
     type : substrings : commonName.&id,
     type : substrings : streetAddress.&id }}
```

The construction of a filter from this value of Guide is straightforward.

Step (1) produces the intermediate Filter value

```
intermediate-filter Filter                ::=
  and : {
    item : substrings  {
      type  commonName.&id,
      strings { any : teletexString : "Dubois" }},
    item : substrings  {
      type streetAddress.&id,
      strings { any : teletexString  "Hugo" }}}
```

Step (2) produces a filter for matching Residential Person entries in the subtree:

```
residential-person-filter Filter ::=
  and : {
    item : equality : {
      type  objectClass.&id,
      assertion  residentialPerson.&id },
    intermediateFilter }
```

5.5.3 Enhanced Search Guide

The *Enhanced Search Guide* attribute provides an enhancement of the **searchGuide** attribute adding information about the recommended search depth for searches among subordinate objects of a given object class.

```
enhancedSearchGuide ATTRIBUTE          ::=        {
  WITH SYNTAX                                     EnhancedGuide
  ID                                              id-at-enhancedSearchGuide }

EnhancedGuide                          ::=        SEQUENCE {
  objectClass                          [0]        OBJECT-CLASS.&id,
  criteria                             [1]        Criteria,
  subset                               [2]        INTEGER
    { baseObject (0), oneLevel (1), wholeSubtree (2) }  DEFAULT oneLevel }
```

5.5.4 Business Category

The *Business Category* attribute type specifies information concerning the occupation of some common objects, e.g. people. For example, this attribute provides the facility to interrogate the Directory about people sharing the same occupation.

```
businessCategory ATTRIBUTE             ::=        {
  WITH SYNTAX                                     DirectoryString {ub-business-category}
  EQUALITY MATCHING RULE                          caseIgnoreMatch
  SUBSTRINGS MATCHING RULE                        caseIgnoreSubstringsMatch
  ID                                              id-at-businessCategory }
```

5.6 Postal Addressing attribute types

These attribute types are concerned with information required for physical postal delivery to an object.

5.6.1 Postal Address

The *Postal Address* attribute type specifies the address information required for the physical delivery of postal messages by the postal authority to the named object.

An attribute value for Postal Address will be typically composed of selected attributes from the MHS Unformatted Postal O/R Address version 1 according to CCITT Recommendation F.401 and limited to 6 lines of 30 characters each, including a Postal Country Name. Normally the information contained in such an address could include an addressee's name, street address, city, state or province, postal code and possibly a Post Office Box number depending on the specific requirements of the named object.

```
postalAddress ATTRIBUTE                ::=        {
  WITH SYNTAX                                     PostalAddress
  EQUALITY MATCHING RULE                          caseIgnoreListMatch
  SUBSTRINGS MATCHING RULE                        caseIgnoreListSubstringsMatch
  ID                                              id-at-postalAddress }

PostalAddress      ::=      SEQUENCE SIZE(1..ub-postal-line) OF  DirectoryString {ub-postal-string}
```

The *Collective Postal Address* attribute type specifies a postal address for a collection of entries.

```
collectivePostalAddress ATTRIBUTE        ::=        {
  SUBTYPE OF                             postalAddress
  COLLECTIVE                             TRUE
  ID                                     id-at-collectivePostalAddress }
```

5.6.2 Postal Code

The *Postal Code* attribute type specifies the postal code of the named object. If this attribute value is present it will be part of the object's postal address.

An attribute value for Postal Code is a string.

```
postalCode ATTRIBUTE              ::=    {
  WITH SYNTAX                            DirectoryString {ub-postal-code}
  EQUALITY MATCHING RULE                 caseIgnoreMatch
  SUBSTRINGS MATCHING RULE               caseIgnoreSubstringsMatch
  ID                                     id-at-postalCode }
```

The *Collective Postal Code* attribute type specifies a postal code for a collection of entries.

```
collectivePostalCode ATTRIBUTE           ::=        {
  SUBTYPE OF                             postalCode
  COLLECTIVE                             TRUE
  ID                                     id-at-collectivePostalCode }
```

5.6.3 Post Office Box

The *Post Office Box* attribute type specifies the Post Office Box by which the object will receive physical postal delivery. If present, the attribute value is part of the object's postal address.

```
postOfficeBox ATTRIBUTE           ::=    {
  WITH SYNTAX                            DirectoryString {ub-post-office-box}
  EQUALITY MATCHING RULE                 caseIgnoreMatch
  SUBSTRINGS MATCHING RULE               caseIgnoreSubstringsMatch
  ID                                     id-at-postOfficeBox }
```

The *Collective Post Office Box* attribute type specifies a post office box for a collection of entries.

```
collectivePostOfficeBox ATTRIBUTE        ::=        {
  SUBTYPE OF                             postOfficeBox
  COLLECTIVE                             TRUE
  ID                                     id-at-collectivePostOfficeBox }
```

5.6.4 Physical Delivery Office Name

The Physical Delivery Office Name attribute type specifies the name of the city, village, etc. where a physical delivery office is situated.

An attribute value for Physical Delivery Office Name is a string.

```
physicalDeliveryOfficeName ATTRIBUTE     ::=        {
 WITH SYNTAX                             DirectoryString {ub-physical-office-name}
 EQUALITY MATCHING RULE                  caseIgnoreMatch
 SUBSTRINGS MATCHING RULE                caseIgnoreSubstringsMatch
 ID                                      id-at-physicalDeliveryOfficeName }
```

The *Collective Physical Delivery Office Name* attribute type specifies a physical delivery office name for a collection of entries.

```
collectivePhysicalDeliveryOfficeName ATTRIBUTE ::=  {
  SUBTYPE OF                             physicalDeliveryOfficeName
  COLLECTIVE                             TRUE
  ID                                     id-at-collectivePhysicalDeliveryOfficeName }
```

5.7 Telecommunications Addressing attribute types

These attribute types are concerned with addressing information needed to communicate with the object using telecommunication means.

5.7.1 Telephone Number

The *Telephone Number* attribute type specifies a telephone number associated with an object.

An attribute value for Telephone Number is a string that complies with the internationally agreed format for showing international telephone numbers, CCITT Recommendation E.123 (e.g. "+ 44 582 10101").

```
telephoneNumber ATTRIBUTE ::= {
  WITH SYNTAX                       TelephoneNumber
  EQUALITY MATCHING RULE            telephoneNumberMatch
  SUBSTRINGS MATCHING RULE          telephoneNumberSubstringsMatch
  ID                                id-at-telephoneNumber }

TelephoneNumber ::= PrintableString (SIZE(1..ub-telephone-number))
  -- String complying with CCITT Recommendation E.123 only
```

The *Collective Telephone Number* attribute type specifies a telephone number for a collection of entries.

```
collectiveTelephoneNumber   ATTRIBUTE   ::=         {
  SUBTYPE OF                            telephoneNumber
  COLLECTIVE                            TRUE
  ID                                    id-at-collectiveTelephoneNumber }
```

5.7.2 Telex Number

The *Telex Number* attribute type specifies the telex number, country code, and answerback code of a telex terminal associated with an object.

```
telexNumber ATTRIBUTE           ::=     {
  WITH SYNTAX                           TelexNumber
  ID                                    id-at-telexNumber }

TelexNumber ::= SEQUENCE {
  telexNumber                           PrintableString (SIZE (1..ub-telex-number)),
  countryCode                           PrintableString (SIZE (1..ub-country-code)),
  answerback                            PrintableString (SIZE (1..ub-answerback)) }
```

The *Collective Telex Number* attribute type specifies a telex number for a collection of entries.

```
collectiveTelexNumber   ATTRIBUTE       ::=         {
  SUBTYPE OF                            telexNumber
  COLLECTIVE                            TRUE
  ID                                    id-at-collectiveTelexNumber }
```

5.7.3 Teletex Terminal Identifier

Since CCITT Recommendation F.200 has been withdrawn and has not been replaced, the use of the ***teletexTerminalIdentifier*** *and the* ***collectiveTeletexTerminalIdentifier*** *attribute types is deprecated.*

The *Teletex Terminal Identifier* attribute type specifies the Teletex terminal identifier (and, optionally, parameters) for a teletex terminal associated with an object.

An attribute value for Teletex Terminal Identifier is a string which complies with CCITT Recommendation F.200 and an optional set whose components are according to ITU-T Recommendation T.62.

```
-- teletexTerminalIdentifier ATTRIBUTE          ::=         {
     -- WITH SYNTAX                             TeletexTerminalIdentifier
     -- ID                                      id-at-teletexTerminalIdentifier }
-- TeletexTerminalIdentifier ::= SEQUENCE       {
     -- teletexTerminal                         PrintableString (SIZE(1..ub-teletex-terminal-id)),
     -- parameters                              TeletexNonBasicParameters OPTIONAL}
```

The *Collective Teletex Terminal Identifier* attribute type specifies a teletex terminal identifier for a collection of entries.

```
-- collectiveTeletexTerminalIdentifier   ATTRIBUTE   ::=         {
     -- SUBTYPE OF                              teletexTerminalIdentifier
     -- COLLECTIVE                              TRUE
     -- ID                                      id-at-collectiveTeletexTerminalIdentifier }
```

5.7.4 Facsimile Telephone Number

The Facsimile Telephone Number attribute type specifies a telephone number for a facsimile terminal (and optionally its parameters) associated with an object.

An attribute value for the facsimile telephone number is a string that complies with the internationally agreed format for showing international telephone numbers, CCITT Recommendation E.123 (e.g. "+81 3 347 7418") and an optional bit string (formatted according to CCITT Reommendation T.30).

```
facsimileTelephoneNumber ATTRIBUTE ::= {
  WITH SYNTAX          FacsimileTelephoneNumber
  ID                   id-at-facsimileTelephoneNumber }

FacsimileTelephoneNumber ::= SEQUENCE {
  telephoneNumber      TelephoneNumber,
  parameters           G3FacsimileNonBasicParameters OPTIONAL}
```

The *Collective Facsimile Telephone Number* attribute type specifies a facsimile telephone number for a collection of entries.

```
collectiveFacsimileTelephoneNumber  ATTRIBUTE     ::=          {
  SUBTYPE OF                        facsimileTelephoneNumber
  COLLECTIVE                        TRUE
  ID                                id-at-collectiveFacsimileTelephoneNumber }
```

5.7.5 X.121 Address

The *X.121 Address* attribute type specifies an address as defined by ITU-T Recommendation X.121 associated with an object.

```
x121Address ATTRIBUTE ::= {
  WITH SYNTAX                       X121Address
  EQUALITY MATCHING RULE            numericStringMatch
  SUBSTRINGS MATCHING RULE          numericStringSubstringsMatch
  ID                                id-at-x121Address }

X121Address ::= NumericString (SIZE(1..ub-x121-address))
        -- String as defined by ITU-T Recommendation X.121
```

5.7.6 International ISDN Number

The *International ISDN Number* attribute type specifies an International ISDN Number associated with an object.

An attribute value for International ISDN Number is a string which complies with the internationally agreed format for ISDN addresses given in ITU-T Recommendation E.164.

```
internationalISDNNumber ATTRIBUTE ::=    {
  WITH SYNTAX                       InternationalISDNNumber
  EQUALITY MATCHING RULE            numericStringMatch
  SUBSTRINGS MATCHING RULE          numericStringSubstringsMatch
  ID                                id-at-internationalISDNNumber }

InternationalISDNNumber ::= NumericString (SIZE(1..ub-international-isdn-number))
        -- String complying with ITU-T Recommendation E.164 only
```

The *Collective International ISDN Number* attribute type specifies an international ISDN number for a collection of entries.

```
collectiveInternationalISDNNumber ATTRIBUTE       ::=          {
  SUBTYPE OF                        internationalISDNNumber
  COLLECTIVE                        TRUE
  ID                                id-at-collectiveInternationalISDNNumber }
```

5.7.7 Registered Address

The *Registered Address* attribute type specifies a mnemonic for an address associated with an object at a particular city location. The mnemonic is registered in the country in which the city is located and is used in the provision of the Public Telegram Service (according to ITU-T Recommendation F.1).

```
registeredAddress ATTRIBUTE       ::=     {
SUBTYPE OF                                postalAddress
WITH SYNTAX                               PostalAddress
ID                                        id-at-registeredAddress }
```

5.7.8 Destination Indicator

The *Destination Indicator* attribute type specifies (according to ITU-T Rec. F.1 and CCITT Rec. F.31) the country and city associated with the object (the addressee) needed to provide the Public Telegram Service.

An attribute value for Destination Indicator is a string.

```
destinationIndicator ATTRIBUTE ::= {
    WITH SYNTAX                     DestinationIndicator
    EQUALITY MATCHING RULE          caseIgnoreMatch
    SUBSTRINGS MATCHING RULE        caseIgnoreSubstringsMatch
    ID                              id-at-destinationIndicator }

DestinationIndicator ::= PrintableString (SIZE(1..ub-destination-indicator))
        -- alphabetical characters only
```

5.8 Preferences attribute types

These attribute types are concerned with the preferences of an object.

5.8.1 Preferred Delivery Method

The *Preferred Delivery Method* attribute type specifies the object's priority order regarding the method to be used for communicating with it.

```
preferredDeliveryMethod ATTRIBUTE     ::={
  WITH SYNTAX                         SEQUENCE OF INTEGER {
      ny-delivery-method                      (0),
      mhs-delivery                            (1),
      physical-delivery                       (2),
      telex-delivery                          (3),
      teletex-delivery                        (4),
      g3-facsimile-delivery                   (5),
      g4-facsimile-delivery                   (6),
      ia5-terminal-delivery                   (7),
      videotex-delivery                       (8),
      telephone-delivery                      (9) }
  SINGLE VALUE                        TRUE
  ID                                  id-at-preferredDeliveryMethod }
```

5.9 OSI Application attribute types

These attribute types are concerned with information regarding objects in the OSI Application Layer.

5.9.1 Presentation Address

The *Presentation Address* attribute type specifies a presentation address associated with an object representing an OSI application entity.

An attribute value for Presentation Address is a presentation address as defined in ITU-T Rec. X.200 | ISO/IEC 7498-1.

```
presentationAddress ATTRIBUTE         ::=         {
  WITH SYNTAX                         PresentationAddress
  EQUALITY MATCHING RULE              presentationAddressMatch
  SINGLE VALUE                        TRUE
  ID                                  id-at-presentationAddress }

PresentationAddress ::= SEQUENCE {
  pSelector                 [0]     OCTET STRING OPTIONAL,
  sSelector                 [1]     OCTET STRING OPTIONAL,
  tSelector                 [2]     OCTET STRING OPTIONAL,
  nAddresses                [3]     SET SIZE (1..MAX)  OF  OCTET STRING }
```

5.9.2 Supported Application Context

The *Supported Application Context* attribute type specifies the object identifier(s) of application context(s) that the object (an OSI application entity) supports.

```
supportedApplicationContext ATTRIBUTE ::=          {
  WITH SYNTAX                       OBJECT IDENTIFIER
  EQUALITY MATCHING RULE            objectIdentifierMatch
  ID                                id-at-supportedApplicationContext }
```

5.9.3 Protocol Information

The *Protocol Information* attribute type associates protocol information with each network address in the Presentation Address attribute.

For each **nAddress**, the protocol component identifies the protocol or profile for the network and transport layers.

```
protocolInformation ATTRIBUTE ::=       {
  WITH SYNTAX                           ProtocolInformation
  EQUALITY MATCHING RULE                 protocolInformationMatch
  ID                                    id-at-protocolInformation }

ProtocolInformation                ::=  SEQUENCE {
  nAddress                              OCTET STRING,
  profiles                              SET OF OBJECT IDENTIFIER }
```

5.10 Relational attribute types

These attribute types are concerned with information regarding the objects which are related to a particular object in certain ways.

NOTE – The DistinguishedName syntax used in these attribute types allows use of the primary distinguished name or an alternative distinguished name. Use of the primary distinguished name, if it is known, ensures consistency and interworking with pre-1997 DSAs. Specific usage may require that a particular alternative name be used. Context information and alternative distinguished values may also be kept as part of the **valuesWithContext** component of any RDN, as described in 9.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

5.10.1 Distinguished Name

The *Distinguished Name* attribute type is an attribute for specifying the name of an object.

```
distinguishedName ATTRIBUTE ::=         {
  WITH SYNTAX                           DistinguishedName
  EQUALITY MATCHING RULE                distinguishedNameMatch
  ID                                    id-at-distinguishedName }
```

5.10.2 Member

The *Member* attribute type specifies a group of names associated with the object.

An attribute value for Member is a distinguished name.

```
member ATTRIBUTE                   ::=  {
  SUBTYPE OF                            distinguishedName
  ID                                    id-at-member }
```

5.10.3 Unique Member

The *Unique Member* attribute type specifies a group of unique names associated with an object. A unique name is a name that is optionally disambiguated by the inclusion of its unique identifier.

An attribute value for Unique Member is a distinguished name accompanied by an optional unique identifier.

```
uniqueMember ATTRIBUTE             ::=  {
  WITH SYNTAX                           NameAndOptionalUID
  EQUALITY MATCHING RULE                uniqueMemberMatch
  ID                                    id-at-uniqueMember }

NameAndOptionalUID                 ::=  SEQUENCE {
  dn                                    DistinguishedName,
  uid                                   UniqueIdentifier OPTIONAL }
```

5.10.4 Owner

The *Owner* attribute type specifies the name of some object which has some responsibility for the associated object.

An attribute value for Owner is a distinguished name (which could represent a group of names) and can recur.

```
owner ATTRIBUTE                ::=    {
  SUBTYPE OF                          distinguishedName
  ID                                  id-at-owner }
```

5.10.5 Role Occupant

The *Role Occupant* attribute type specifies the name of an object which fulfills an organizational role.

An attribute value for Role Occupant is a distinguished name.

```
roleOccupant ATTRIBUTE         ::=    {
  SUBTYPE OF                          distinguishedName
  ID                                  id-at-roleOccupant }
```

5.10.6 See Also

The *See Also* attribute type specifies names of other Directory objects which may be other aspects (in some sense) of the same real world object.

An attribute value for See Also is a distinguished name.

```
seeAlso ATTRIBUTE              ::=    {
  SUBTYPE OF                          distinguishedName
  ID                                  id-at-seeAlso }
```

5.11 Domain attribute types

5.11.1 DMD name

The DMD Name attribute type specifies a DMD. When used as a component of a directory name it identifies a DMD which manages the named object.

An attribute value for DMD Name is a string chosen by the DMD.

```
dmdName   ATTRIBUTE ::= {
  SUBTYPE OF          name
  WITH SYNTAX         DirectoryString{ub-common-name}
  ID                  id-at-dmdName }
```


SECTION 3 – MATCHING RULES

6 Definition of matching rules

NOTE – For definitions of **objectIdentifierMatch** and **distinguishedNameMatch**, see ITU-T Rec. X.501 | ISO/IEC 9594-2.

6.1 String matching rules

In the matching rules specified in 6.1.1 through 6.1.11, the following spaces are regarded as not significant:

- leading spaces (i.e. those preceding the first character that is not a space);
- trailing spaces (i.e. those following the last character that is not a space);
- multiple consecutive spaces (these are taken as equivalent to a single space character).

A string consisting entirely of spaces is equivalent to a string containing exactly one space.

In the matching rules to which these apply, the strings to be matched shall be matched as if the insignificant spaces were not present in either string.

6.1.1 Case Ignore Match

The *Case Ignore Match* rule compares for equality a presented string with an attribute value of **type PrintableString, NumericString, TelexString, BMPString, UniversalString,** or **DirectoryString**, without regard to the case (upper or lower) of the strings (e.g. "Dundee" and "DUNDEE" match).

```
caseIgnoreMatch MATCHING-RULE        ::=        {
  SYNTAX                             DirectoryString {ub-match}
  ID                                 id-mr-caseIgnoreMatch }
```

The rule returns TRUE if the strings are the same length and corresponding characters are identical except possibly with regard to case.

Where the strings being matched are of different ASN.1 syntax, the comparison proceeds as normal so long as the corresponding characters are in both character sets. Otherwise matching fails.

6.1.2 Case Ignore Ordering Match

The *Case Ignore Ordering Match* rule compares the collation order of a presented string an attribute value whose type is one of the ones listed in 6.1.1, without regard to the case (upper or lower) of the strings.

```
caseIgnoreOrderingMatch MATCHING-RULE        ::=        {
SYNTAX                                       DirectoryString {ub-match}
ID                                           id-mr-caseIgnoreOrderingMatch }
```

The rule returns TRUE if the attribute value is "less" or appears earlier than the presented value, when the strings are compared using the normal collation order for their syntax after lower-case letters in both strings have been replaced by their upper-case equivalents.

Where the strings being matched are of different ASN.1 syntax, the comparison proceeds as normal so long as the corresponding characters are in both character sets. Otherwise matching fails.

6.1.3 Case Ignore Substrings Match

The *Case Ignore Substrings Match* rule determines whether a presented value is a substring of an attribute value whose type is one of the ones listed in 6.1.1 , without regard to the case (upper or lower) of the strings.

```
caseIgnoreSubstringsMatch MATCHING-RULE        ::=        {
  SYNTAX                                       SubstringAssertion
  ID                                           id-mr-caseIgnoreSubstringsMatch }

SubstringAssertion ::= SEQUENCE OF CHOICE {
  initial                          [0]     DirectoryString {ub-match},
  any                              [1]     DirectoryString {ub-match},
  final                            [2]     DirectoryString {ub-match} }
  -- at most one initial and one final component
```

The rule returns TRUE if there is a partitioning of the attribute value (into portions) such that:

- the specified substrings (**initial**, **any**, **final**) match different portions of the value in the order of the **strings** sequence;
- **initial**, if present, matches the first portion of the value;
- **final**, if present, matches the last portion of the value;
- **any**, if present, matches some arbitrary portion of the value.

There shall be at most one **initial**, and at most one **final** in **strings**. If **initial** is present, it shall be the first element of **strings**. If **final** is present, it shall be the last element of **strings**. There shall be zero or more **any** in **strings**.

For a component of substrings to match a portion of the attribute value, corresponding characters must be identical, except in regard to case. Where the strings being matched are of different ASN.1 syntax, the comparison proceeds as normal so long as the corresponding characters are in both character sets. Otherwise matching fails.

6.1.4 Case Exact Match

The *Case Exact Match* rule compares for equality a presented string with an attribute value whose type is one of the ones listed in 6.1.1.

```
caseExactMatch MATCHING-RULE        ::=        {
  SYNTAX                                       DirectoryString {ub-match}
  ID                                           id-mr-caseExactMatch }
```

The rule is identical to the **caseIgnoreMatch** rule except that case is not ignored.

6.1.5 Case Exact Ordering Match

The *Case Exact Ordering Match* rule compares the collation order of a presented string with an attribute value whose type is one of the ones listed in 6.1.1.

```
caseExactOrderingMatch MATCHING-RULE        ::=        {
  SYNTAX                                    DirectoryString {ub-match}
  ID                                        id-mr-caseExactOrderingMatch }
```

The rule is identical to the **caseIgnoreOrderingMatch** rule except that lower-case letters are not replaced by upper-case letters.

6.1.6 Case Exact Substrings Match

The *Case Exact Substrings Match* rule determines whether a presented value is a substring of an attribute value whose type is one of the ones listed in 6.1.1.

```
caseExactSubstringsMatch MATCHING-RULE        ::=        {
  SYNTAX                                      SubstringAssertion -- only the PrintableString choice
  ID                                          id-mr-caseExactSubstringsMatch }
```

The rule is identical to the **caseIgnoreSubstringsMatch** rule except that case is not ignored.

6.1.7 Numeric String Match

The *Numeric String Match* rule compares for equality a presented numeric string with an attribute value of type **NumericString**.

```
numericStringMatch MATCHING-RULE        ::=        {
  SYNTAX                                NumericString
  ID                                    id-mr-numericStringMatch }
```

The rule is identical to the **caseIgnoreMatch** rule except that all space characters are skipped during comparison (case is irrelevant as characters are numeric).

6.1.8 Numeric String Ordering Match

The *Numeric String Ordering Match* rule compares the collation order of a presented string with an attribute value of type **NumericString**.

```
numericStringOrderingMatch MATCHING-RULE        ::=        {
  SYNTAX                                        NumericString
  ID                                            id-mr-numericStringOrderingMatch }
```

The rule is identical to the **caseIgnoreOrderingMatch** rule except that all space characters are skipped during comparison (case is irrelevant as characters are numeric).

6.1.9 Numeric String Substrings Match

The *Numeric String Substrings Match* rule determines whether a presented value is a substring of an attribute value of type **NumericString**.

```
numericStringSubstringsMatch MATCHING-RULE        ::=        {
  SYNTAX                                          SubstringAssertion
  ID                                              id-mr-numericStringSubstringsMatch }
```

The rule is identical to the **caseIgnoreSubstringsMatch** rule except that all space characters are skipped during comparison (case is irrelevant as characters are numeric).

6.1.10 Case Ignore List Match

The *Case Ignore List Match* rule compares for equality a presented sequence of strings with an attribute value which is a sequence of **DirectoryString**s, without regard to the case (upper or lower) of the strings.

```
caseIgnoreListMatch MATCHING-RULE      ::=          {
  SYNTAX                               SEQUENCE OF DirectoryString {ub-match}
  ID                                   id-mr-caseIgnoreListMatch }
```

The rule returns TRUE if and only if the number of strings in each is the same, and corresponding strings match. The latter matching is as for the **caseIgnoreMatch** matching rule.

6.1.11 Case Ignore List Substrings Match

The *Case Ignore List Substring Match* rule compares a presented substring with an attribute value which is a sequence of **DirectoryString**s, but where the case (upper or lower) is not significant for comparison purposes.

```
caseIgnoreListSubstringsMatch MATCHING-RULE  ::=      {
  SYNTAX                                     SubstringAssertion
  ID                                         id-mr-caseIgnoreListSubstringsMatch }
```

A presented value matches a stored value if and only if the presented value matches the string formed by concatenating the strings of the stored value. This matching is done according to the **caseIgnoreSubstringsMatch** rule; however, none of the **initial**, **any**, or **final** values of the presented value are considered to match a substring of the concatenated string which spans more than one of the strings of the stored value.

6.1.12 Stored Prefix Match

The *Stored Prefix Match* rule determines whether an attribute value, whose syntax is DirectoryString, is a prefix (i.e. initial substring) of the presented value, without regard to the case (upper or lower) of the strings.

NOTE – It can be used, for example, to compare values in the Directory which are telephone area codes with a purported value which is a telephone number.

```
storedPrefixMatch MATCHING-RULE    ::=      {
  SYNTAX                           DirectoryString {ub-match}
  ID                               id-mr-storedPrefixMatch }
```

The rule returns TRUE if the attribute value is an initial substring of the presented value with corresponding characters identical except possibly with regard to case.

6.2 Syntax-based matching rules

6.2.1 Boolean Match

The *Boolean Match* rule compares for equality a presented Boolean value with an attribute value of type **BOOLEAN**.

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

The rule returns TRUE if the values are the same, i.e. both are TRUE or both are FALSE.

6.2.2 Integer Match

The *Integer Match* rule compares for equality a presented integer value with an attribute value of type **INTEGER**.

```
integerMatch MATCHING-RULE           ::=          {
  SYNTAX                             INTEGER
  ID                                 id-mr-integerMatch }
```

The rule returns TRUE if the integers are equal.

6.2.3 Integer Ordering Match

The *Integer Ordering Match* rule compares a presented integer value with an attribute value of type **INTEGER**.

```
integerOrderingMatch MATCHING-RULE   ::=          {
  SYNTAX                             INTEGER
  ID                                 id-mr-integerOrderingMatch }
```

The rule returns TRUE if the attribute value is less than the presented value.

6.2.4 Bit String Match

The *Bit String Match* rule compares a presented bit string with an attribute value of type **BIT STRING**.

```
bitStringMatch MATCHING-RULE        ::=        {
  SYNTAX                            BIT STRING
  ID                                id-mr-bitStringMatch }
```

The rule returns TRUE if the attribute value has the same number of bits as the presented value and the bits match on a bitwise basis.

6.2.5 Octet String Match

The *Octet String Match* rule compares for equality a presented octet string with an attribute value of type **OCTET STRING**.

```
octetStringMatch MATCHING-RULE      ::=        {
  SYNTAX                            OCTET STRING
  ID                                id-mr-octetStringMatch }
```

The rule returns TRUE if and only if the strings are the same length and corresponding octets are identical.

6.2.6 Octet String Ordering Match

The *Octet String Ordering Match* rule compares the collation order of a presented octet string with an attribute value of type **OCTET STRING**.

```
octetStringOrderingMatch MATCHING-RULE   ::=      {
  SYNTAX                            OCTET STRING
  ID                                id-mr-octetStringOrderingMatch }
```

The rule compares octet strings from first octet to last octet, and from the most significant bit to the least significant bit within the octet. The first occurrence of a different bit determines the ordering of the strings. A zero bit precedes a one bit. If the strings are identical but contain different numbers of octets, the shorter string precedes the longer string.

6.2.7 Octet String Substrings Match

The *Octet String Substrings Match* rule determines whether a presented octet string is a substring of an attribute value of type **OCTET STRING**.

```
octetStringSubstringsMatch MATCHING-RULE   ::=        {
  SYNTAX                            OctetSubstringAssertion
  ID                                id-mr-octetStringSubstringsMatch }

OctetSubstringAssertion ::= SEQUENCE OF CHOICE {
  initial                           [0]     OCTET STRING,
  any                               [1]     OCTET STRING,
  final                             [2]     OCTET STRING }
  -- at most one initial and one final component
```

The rule returns TRUE if the attribute value contains the sequence of octets in the presented string, as described for **caseIgnoreSubstringsMatch**.

6.2.8 Telephone Number Match

The *Telephone Number Match* rule compares for equality a presented value with an attribute value of type **PrintableString** which is a telephone number.

```
telephoneNumberMatch MATCHING-RULE ::=        {
  SYNTAX                            PrintableString
  ID                                id-mr-telephoneNumberMatch }
```

The rules for matching are identical to those for **caseIgnoreMatch**, except that all space and "-" characters are skipped during the comparison.

6.2.9 Telephone Number Substrings Match

The *Telephone Number Substrings Match* rule determines if a presented substring is a substring of an attribute value of type **PrintableString** which is a telephone number.

telephoneNumberSubstringsMatch MATCHING-RULE ::= {
SYNTAX SubstringAssertion
ID id-mr-telephoneNumberSubstringsMatch }

The rules for matching are identical to those for **caseExactSubstringsMatch**, except that all space and "-" characters are skipped during the comparison.

6.2.10 Presentation Address Match

The *Presentation Address Match* rule compares for equality a presented Presentation Address with an attribute value of type **PresentationAddress**.

presentationAddressMatch MATCHING-RULE ::= {
SYNTAX PresentationAddress
ID id-mr-presentationAddressMatch }

The rule returns TRUE if and only if the selectors of the presented and stored presentation address are equal and the presented **nAddresses** are a subset of the stored ones.

6.2.11 Unique Member Match

The *Unique Member Match* rule compares for equality a presented Unique Member value with an attribute value of type **NameAndOptionalUID**.

uniqueMemberMatch MATCHING-RULE ::= {
SYNTAX NameAndOptionalUID
ID id-mr-uniqueMemberMatch }

The rule returns TRUE if and only if the **dn** components of the attribute value and the presented value match according to the **distinguishedNameMatch** rule, and the **uid** component is absent from the attribute value or matches the corresponding component from the presented value according to the **bitStringMatch** rule.

6.2.12 Protocol Information Match

The *Protocol Information Match* rule compares for equality presented values of **ProtocolInformation** with values of the same type.

protocolInformationMatch MATCHING-RULE ::= {
SYNTAX OCTET STRING
ID id-mr-protocolInformationMatch }

A value of the assertion syntax is derived from a value of the attribute syntax by using the **nAddress** component.

The value returns True if the presented value and the **nAddress** component of the stored value match according to the **octetStringMatch** rule.

6.3 Time matching rules

6.3.1 UTC Time Match

The *UTC Time Match* rule compares for equality a presented value with an attribute value of type **UTCTime**.

uTCTimeMatch MATCHING-RULE ::= {
SYNTAX UTCTime
ID id-mr-uTCTimeMatch }

The rule returns TRUE if the attribute value represents the same time as the presented value. If a UTC time is specified with the seconds absent, the number of seconds is assumed to be zero.

6.3.2 UTC Time Ordering Match

The *UTC Time Ordering* rule compares the time ordering of a presented value with an attribute value of type **UTCTime**.

```
uTCTimeOrderingMatch MATCHING-RULE ::= {
  SYNTAX                    UTCTime
  ID                        id-mr-uTCTimeOrderingMatch }
```

The rule returns TRUE if the attribute value represents a time which is earlier than the presented time. UTC times with year values 50 to 99 shall be taken to represent times that are earlier than UTC times with year values 00 to 49. If a UTC time is specified with the seconds absent, the number of seconds is assumed to be zero.

The value of the two-digit year field shall be rationalized into a four-digit year value as follows:

– if the 2-digit value is 00 through 49 inclusive, the value shall have 2000 added to it; and

– if the 2-digit value is 50 through 99 inclusive, the value shall have 1900 added to it.

6.3.3 Generalized Time Match

The *Generalized Time Match* rule compares for equality a presented value with an attribute value of type **GeneralizedTime** [as per 41.3 b) or c) of ITU-T Rec. X.680 | ISO/IEC 8824-1].

```
generalizedTimeMatch MATCHING-RULE  ::=          {
  SYNTAX                            GeneralizedTime
                                    -- as per 41.3 b) or c) of ITU-T Rec. X.680 | ISO/IEC 8824-1
  ID                                id-mr-generalizedTimeMatch }
```

The rule returns TRUE if the attribute value represents the same time as the presented value. If a time is specified with the minutes or seconds absent, the number of minutes or seconds is assumed to be zero.

6.3.4 Generalized Time Ordering Match

The *Generalized Time Ordering Match* rule compares the time ordering of a presented value with an attribute value of type **GeneralizedTime** [as per 41.3 b) and c) of ITU-T Rec. X.680 | ISO/IEC 8824-1].

```
generalizedTimeOrderingMatch MATCHING-RULE  ::= {
  SYNTAX                            GeneralizedTime
                                    -- as per 41.3 b) or c) of ITU-T Rec. X.680 | ISO/IEC 8824-1
  ID                                id-mr-generalizedTimeOrderingMatch }
```

The rule returns TRUE if the attribute value represents a time which is earlier than the presented time. If a time is specified with the minutes or seconds absent, the number of minutes or seconds is assumed to be zero.

6.4 First component matching rules

6.4.1 Integer First Component Match

The *Integer First Component Match* rule compares for equality a presented integer value with an attribute value of type **SEQUENCE** whose first component is mandatory and of type **INTEGER**.

```
integerFirstComponentMatch MATCHING-RULE  ::=  {
SYNTAX                                  INTEGER
ID                                      id-mr-integerFirstComponentMatch }
```

The rule returns TRUE if the attribute value has a first component whose value equals the presented integer.

A value of the assertion syntax is derived from a value of the attribute syntax by using the value of the first component of the SEQUENCE.

6.4.2 Object Identifier First Component Match

The *Object Identifier First Component Match* rule compares for equality a presented object identifier value with attribute values of type **SEQUENCE** whose first component is mandatory and of type **OBJECT IDENTIFIER**.

```
objectIdentifierFirstComponentMatch MATCHING-RULE ::=        {
  SYNTAX                                  OBJECT IDENTIFIER
  ID                                      id-mr-objectIdentifierFirstComponentMatch }
```

The rule returns TRUE if the attribute value has a first component whose value matches the presented object identifier using the rules of **objectIdentifierMatch**.

A value of the assertion syntax is derived from a value of the attribute syntax by using the value of the first component of the SEQUENCE.

6.4.3 Directory String First Component Match

The *Directory String First Component Match* rule compares for equality a presented **DirectoryString** value with an attribute value of type **SEQUENCE** whose first component is mandatory and of type **DirectoryString**.

```
directoryStringFirstComponentMatch MATCHING-RULE ::=        {
  SYNTAX                                DirectoryString {ub-directory-string-first-component-match}
  ID                                    id-mr-directoryStringFirstComponentMatch }
```

The rule returns TRUE if the attribute value has a first component whose value matches the presented **DirectoryString** using the rules of **caseIgnoreMatch**.

A value of the assertion syntax is derived from a value of the attribute syntax by using the value of the first component of the SEQUENCE.

6.5 Word matching rules

6.5.1 Word Match

The *Word Match* rule compares a presented string with words in an attribute value of type **DirectoryString**.

```
wordMatch MATCHING-RULE    ::=      {
  SYNTAX                              DirectoryString {ub-match}
  ID                                  id-mr-wordMatch }
```

The rule returns TRUE if a presented word matches any word in the attribute value. Individual word matching is as for the **caseIgnoreMatch** matching rule. The precise definition of a "word" is a local matter.

6.5.2 Keyword Match

The *Keyword Match* rule compares a presented string with keywords in an attribute value of type **DirectoryString**.

```
keywordMatch MATCHING-RULE        ::= {
  SYNTAX                              DirectoryString {ub-match}
  ID                                  id-mr-keywordMatch }
```

The rule returns TRUE if a presented value matches any *keyword* in the attribute value. The identification of keywords in an attribute value and of the exactness of match are both local matters.

SECTION 4 – CONTEXTS

7 Definition of Context Types

This Directory Specification defines a number of context types which may be found useful across a range of applications of the Directory.

7.1 Language Context

The *Language Context* associates an attribute value with a specific language(s):

```
languageContext CONTEXT ::= {
  WITH SYNTAX           LanguageContextSyntax
  ID                    id-avc-language }

LanguageContextSyntax ::= PrintableString (SIZE(2..3))  -- ISO 639-2 codes only
```

A presented value is considered to match a stored value if the sequence of characters in the presented value is identical to that in the stored value.

7.2 Temporal Context

The Temporal Context associates an attribute value with a set of times. Various expressions of time are possible, including:

a) absolute start or end times (e.g. 24:00 December 14, 1994);

b) specific time bands within the day (e.g. 09:00 to 17:00)

c) days within the week (e.g. Monday);

d) days within the month (e.g. the 10th; the 2nd last day, etc.);

e) months within the year (e.g. March);

f) a particular year (e.g. 1995);

g) weeks within the month (e.g. the second week);

h) periodic day or week (e.g. every 2nd week);

i) logical negatives (e.g. not Monday).

```
temporalContext CONTEXT ::= {
  WITH SYNTAX          TimeSpecification
  ASSERTED AS          TimeAssertion
  ID                   id-avc-temporal }

TimeSpecification ::= SEQUENCE {
  time    CHOICE {
          absolute          SEQUENCE {
                  startTime         [0] GeneralizedTime OPTIONAL,
                  endTime           [1] GeneralizedTime OPTIONAL },
          periodic          SET OF Period },
  notThisTime  BOOLEAN DEFAULT FALSE,
  timeZone     TimeZone OPTIONAL }

Period ::= SEQUENCE {
  timesOfDay       [0]    SET OF DayTimeBand OPTIONAL,
  days             [1]    CHOICE  {
                            intDay      SET OF INTEGER,
                            bitDay      BIT STRING  { sunday (0), monday (1) , tuesday (2),
                                        wednesday (3), thursday (4), friday (5), saturday (6) },
                            dayOf             XDayOf } OPTIONAL,
  weeks            [2]    CHOICE  {
                            allWeeks    NULL,
                            intWeek     SET OF INTEGER,
                            bitWeek     BIT STRING  { week1 (0), week2 (1), week3 (2), week4 (3),
                                        week5 (4) } } OPTIONAL,
  months           [3]    CHOICE  {
                            allMonths   NULL,
                            intMonth    SET OF INTEGER,
                            bitMonth    BIT STRING  { january (0), february (1), march (2), april (3),
                                        may (4), june (5), july (6), august (7), september (8),
                                        october (9), november (10), december (11) }
                                  } OPTIONAL,
  years            [4]    SET OF INTEGER (1000 .. MAX) OPTIONAL }

XDayOf ::= CHOICE {
  first            [1]    NamedDay,
  second           [2]    NamedDay,
  third            [3]    NamedDay,
  fourth           [4]    NamedDay,
  fifth            [5]    NamedDay}
```

```
NamedDay ::= CHOICE {
        intNamed Days       ENUMERATED {
                                sunday      (1),
                                monday      (2),
                                tuesday     (3),
                                wednesday   (4),
                                thursday    (5),
                                friday      (6),
                                saturday    (7) },
        bitNamedDays        BIT STRING  { sunday (0), monday (1) , tuesday (2),
                                wednesday (3), thursday (4), friday (5), saturday (6) } }

DayTimeBand ::= SEQUENCE {
  startDayTime      [0]     DayTime DEFAULT { hour 0 },
  endDayTime        [1]     DayTime DEFAULT { hour 23, minute 59, second 59 }}

DayTime ::= SEQUENCE {
  hour              [0]     INTEGER (0..23),
  minute            [1]     INTEGER (0..59) DEFAULT 0,
  second            [2]     INTEGER (0..59) DEFAULT 0}

TimeZone ::= INTEGER (-12..12)

TimeAssertion ::= CHOICE {
  now               NULL,
  at                GeneralizedTime,
  between           SEQUENCE {
        startTime       [0] GeneralizedTime,
        endTime         [1] GeneralizedTime OPTIONAL,
        entirely            BOOLEAN DEFAULT FALSE  }}
```

The **absolute** choice of **time** expresses a specific time or time band using absolute time notations (GeneralizedTime). A specific time is expressed by setting the **startTime** equal to the **endTime**. Otherwise, **startTime** is earlier in time than **endTime** and a span of time is expressed. If **endTime** is missing the time span includes all times after **startTime**.

periodic allows the specification of **time** as a set of periods. The combined effect is a logical OR of the set.

NOTE 1 – Alternatively, an attribute value could be associated with the temporal context with multiple context values, one for each of the periods, since this also acts a logical OR. However, the SET OF is included here to allow **notThisTime** to cover the set and thus effect a logical 'neither'. When **notThisTime** is FALSE, the choice of which approach to use to specify a set of periods is up to the specifier.

Within each Period each element in the SEQUENCE OF is considered as "within" the following element in the SEQUENCE OF. The SEQUENCE OF is in rising order of granularity of time period, although not all levels must be present.

The final element in a **Period** is assumed to be valid for all time periods of higher granularity.

NOTE 2 – For example, if a **Period** SEQUENCE OF ends with **timesOfDay**, it is considered valid for all days.

A **timesOfDay** indicates the valid time bands during the days specified in the next element of **Period**. If **days** is not the next element, then the time bands are valid for all possible days within the next element. If **timesOfDay** is not included, all times of the day are valid within the next element. Different time bands may be specified for different days, by having multiple occurrences of **Period**.

The **days** element expresses specific days of a week, month or year depending on the next element of **Period**. If **days** precedes **weeks** in a **Period**, then it expresses days of the week and the INTEGERs are constrained to the values 1 to 7, where 1 is Sunday. If **days** precedes **months** in a **Period**, then it expresses days in the month and the INTEGERs are constrained to the values 1 to 31, where 1 is the first day of the month. If **days** precedes **years** in a **Period**, then it expresses days of the year and the INTEGERs are constrained to the values 1 to 366, where 1 is the first day of the year.

dayOf is used to indicate the 1st, 2nd, 3rd, 4th, and 5th occurrence of the **NamedDay** in a month (e.g. the first Monday of the month, or the second Tuesday and Friday of August). The use of **fifth** shall always indicate the last **NamedDay** of that month (e.g. the last Tuesday of July). If the **dayOf** choice for **days** is specified, then the **weeks** element of **Period** is not meaningful if present and is ignored.

If **days** is not specified, then all days are valid within the next element of the **Period**.

The **weeks** element expresses specific weeks of a month or year, depending on the next element of **Period**. If **weeks** precedes months in a **Period**, then it expresses weeks of the month and the INTEGERs are constrained to the values 1 to 5, where 1 is the first week of the month. The first week of the month shall be assumed to be the first week containing at least four days of that month. The fifth week always means the last week of the month.

If **weeks** precedes years in a **Period**, then it expresses weeks of the year and the INTEGERs are constrained to the values 1 to 53, where 1 is the first week of the year. The first week of the year shall be assumed to be the first week containing at least four days of that year. The 53rd week is always the last week of the year.

If **allWeeks** is specified, then all weeks are valid within the next element of the **Period** (this allows **days** to express days of the week for all weeks).

If **weeks** is not specified, then all weeks are valid within the next element of the **Period**.

The **months** element expresses specific months of the year. When **months** is expressed with INTEGERs, the INTEGERs are constrained to the values 1 to 12, where 1 is the first month of the year (i.e. January).

If **allMonths** is specified, then all months of the year are valid (this allows **weeks** to express weeks of the month for all months, or if **weeks** is not specified it allows **days** to express days of the month for all months).

If **months** is not specified, then all months of the year are valid.

The **years** component expresses one or more years. If **years** is not specified, then all years are valid.

timeZone expresses the time zone, in hours delta from GMT, in which **time** is expressed. If **timeZone** is not present, a DSA processing the temporal context shall interpret the **time** relevant in the time zone of the DSA.

If **notThisTime** is **FALSE**, then the temporal context value is the time expressed in **time** in the **TimeSpecification.** If **notThisTime** is **TRUE**, then the temporal context value is considered to be all time except that expressed in **time** in the **TimeSpecification** (that is, a logical NOT is performed).

A time assertion is considered to match a time specification if there is an overlap in the times specified. If the time assertion contains **now**, then the current time is used in the evaluation. If **now** or **at** is specified, then the assertion is considered true if the specific time falls within the times covered by the stored **TimeSpecification**. If the time assertion uses **between** and **entirely** is **FALSE**, then the assertion is considered true if any portion of the **between** time band falls within the times covered by the stored **TimeSpecification** (the overlap need not be complete: as long as there is a period of overlap within the two time specifications, they are considered to match). If the time assertion uses **between** and **entirely** is **TRUE**, then the assertion is considered true only if the entire **between** time band falls within the times covered by the stored **TimeSpecification.**

Examples:

NOTE 3 – The following examples use the INTEGER formats for elements where a choice is available of INTEGER or BIT STRING.

a) 09:00 to 17:00 every day, would be expressed as:

```
periodic {
        timesOfDay { {
            startDayTime hour 9,
            endDayTime hour 17 } } }.
```

b) Every Monday would be expressed as:

```
periodic {
        days intDay : {2} }
```

c) 09:00 to 12:00 noon Monday to Friday and all day Saturday during January, and all day for Tuesdays in February and March would be expressed as:

```
periodic {
        timesOfDay { {
            startDayTime hour 9,
            endDayTime hour 12} }
        days      intDay : {2,3,4,5,6},
        weeks     allWeeks : NULL,
        months    intMonth : {1} },
```

```
{       days {7},
        weeks {1,2,3,4,5}
        months {1} }

{       days {3}
        weeks {1,2,3,4,5},
        months {2,3} } }
```

d) All of August 1996 would be expressed as:

```
periodic {
        {       months {8}
                years {1996} } }
```

e) The first day of every month would be expressed as:

```
periodic {
        {       days {1}
                months NULL } }
```

7.3 Locale Context

The Locale Context associates an attribute value with a specific locale(s) as defined in POSIX:

```
localeContext CONTEXT ::= {
  WITH SYNTAX          localeContextSyntax
  ID                   id-avc-locale }

localeContextSyntax :: = CHOICE {
  localeID1                 OBJECT IDENTIFIER,
  localeID2                 DirectoryString }
```

A presented value is considered to match a stored value if they are both object identifiers and the two object identifiers are equal, or they are both strings and are the same.

Only registered object identifiers or strings for locales may be used as context values. The concept of locales is described in ISO/IEC 9945-2:1993 Information Technology – Portable Operating System Interface (POSIX) – Part 2: Shell and Utilities.

NOTE – Registration authorities will be created to assign OIDs and/or string identifiers to locale specifications. For example, the European Committee for Standardization, CEN, has published a European standard for registration of locale information, ENV12005 :1996 Procedures for European Registration of Cultural Elements.

Annex A

Selected attribute types in ASN.1

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

This annex includes all of the ASN.1 type and value definitions contained in this Directory Specification in the form of the ASN.1 module **SelectedAttributeTypes**.

```
SelectedAttributeTypes {joint-iso-itu-t ds(5) module(1) selectedAttributeTypes(5) 3}
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
    informationFramework, upperBounds, id-at, id-mr, id-avc
      FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 3 }

    ATTRIBUTE, MATCHING-RULE, AttributeType, OBJECT-CLASS, DistinguishedName,
    objectIdentifierMatch, distinguishedNameMatch
      FROM InformationFramework informationFramework

    G3FacsimileNonBasicParameters, TeletexNonBasicParameters
      FROM MTSAbstractService{joint-iso-itu-t mhs(6) mts(3) modules(0)
          mts-abstract-service(1)version-1999(1) }

    ub-answerback, ub-name, ub-common-name, ub-surname, ub-serial-number, ub-locality-name,
    ub-state-name, ub-street-address, ub-organization-name, ub-organizational-unit-name, ub-title,
    ub-description, ub-business-category, ub-postal-line, ub-postal-string, ub-postal-code,
    ub-post-office-box, ub-physical-office-name, ub-telex-number, ub-country-code,
    ub-teletex-terminal-id, ub-telephone-number, ub-x121-address, ub-international-isdn-number,
    ub-destination-indicator, ub-user-password, ub-match, ub-knowledge-information, ub-name,
    ub-directory-string-first-component-match
      FROM UpperBounds upperBounds ;

-- Directory string type --

DirectoryString { INTEGER : maxSize } ::= CHOICE {
    teletexString           TeletexString (SIZE (1..maxSize)),
    printableString         PrintableString (SIZE (1..maxSize)),
    bmpString               BMPString (SIZE (1..maxSize)),
    universalString         UniversalString (SIZE (1..maxSize)) }

-- Attribute types --

knowledgeInformation ATTRIBUTE          ::=     {
    WITH SYNTAX                         DirectoryString { ub-knowledge-information }
    EQUALITY MATCHING RULE              caseIgnoreMatch
    ID                                  id-at-knowledgeInformation }

name ATTRIBUTE ::=  {
    WITH SYNTAX                         DirectoryString { ub-name }
    EQUALITY MATCHING RULE              caseIgnoreMatch
    SUBSTRINGS MATCHING RULE            caseIgnoreSubstringsMatch
    ID                                  id-at-name }

commonName ATTRIBUTE        ::=         {
    SUBTYPE OF                          name
    WITH SYNTAX                         DirectoryString {ub-common-name}
    ID                                  id-at-commonName }

surname ATTRIBUTE       ::=             {
    SUBTYPE OF                          name
    WITH SYNTAX                         DirectoryString {ub-surname}
    ID                                  id-at-surname }
```

```
givenName ATTRIBUTE   ::=   {
      SUBTYPE OF              name
      WITH SYNTAX                     DirectoryString {ub-name}
      ID                              id-at-givenName }

initials ATTRIBUTE ::=          {
      SUBTYPE OF              name
      WITH SYNTAX                     DirectoryString {ub-name}
      ID                              id-at-initials }

generationQualifier ATTRIBUTE ::=               {
      SUBTYPE OF              name
      WITH SYNTAX                     DirectoryString {ub-name}
      ID                              id-at-generationQualifier}

uniqueIdentifier ATTRIBUTE    ::=       {
      WITH SYNTAX                     UniqueIdentifier
      EQUALITY MATCHING RULE          bitStringMatch
      ID                              id-at-uniqueIdentifier }

UniqueIdentifier   ::=      BIT STRING

dnQualifier ATTRIBUTE ::=     {
      WITH SYNTAX                     PrintableString
      EQUALITY MATCHING RULE          caseIgnoreMatch
      ORDERING MATCHING RULE          caseIgnoreOrderingMatch
      SUBSTRINGS MATCHING RULE        caseIgnoreSubstringsMatch
      ID                              id-at-dnQualifier }

serialNumber ATTRIBUTE ::=  {
      WITH SYNTAX                     PrintableString (SIZE (1..ub-serial-number))
      EQUALITY MATCHING RULE          caseIgnoreMatch
      SUBSTRINGS MATCHING RULE        caseIgnoreSubstringsMatch
      ID                              id-at-serialNumber }

countryName ATTRIBUTE ::=  {
      SUBTYPE OF              name
      WITH SYNTAX                     CountryName
      SINGLE VALUE                    TRUE
      ID                              id-at-countryName }

CountryName ::=          PrintableString (SIZE (2)) --SO 3166 codes only

localityName ATTRIBUTE ::=  {
      SUBTYPE OF              name
      WITH SYNTAX                     DirectoryString {ub-locality-name}
      ID                              id-at-localityName }

collectiveLocalityName ATTRIBUTE        ::=     {
      SUBTYPE OF              localityName
      COLLECTIVE              TRUE
      ID                              id-at-collectiveLocalityName }

stateOrProvinceName ATTRIBUTE           ::=     {
      SUBTYPE OF              name
      WITH SYNTAX                     DirectoryString {ub-state-name}
      ID                              id-at-stateOrProvinceName }

collectiveStateOrProvinceName ATTRIBUTE ::=  {
      SUBTYPE OF              stateOrProvinceName
      COLLECTIVE              TRUE
      ID                              id-at-collectiveStateOrProvinceName }

streetAddress ATTRIBUTE ::=  {
      WITH SYNTAX                     DirectoryString {ub-street-address}
      EQUALITY MATCHING RULE          caseIgnoreMatch
      SUBSTRINGS MATCHING RULE        caseIgnoreSubstringsMatch
      ID                              id-at-streetAddress }

collectiveStreetAddress ATTRIBUTE       ::=     {
      SUBTYPE OF              streetAddress
      COLLECTIVE              TRUE
      ID                              id-at-collectiveStreetAddress }
```

```
houseIdentifier ATTRIBUTE     ::=          {
        WITH SYNTAX                        DirectoryString {ub-name}
        EQUALITY MATCHING RULE             caseIgnoreMatch
        SUBSTRINGS MATCHING RULE           caseIgnoreSubstringsMatch
        ID                                 id-at-houseIdentifier }

organizationName ATTRIBUTE ::=             {
        SUBTYPE OF                 name
        WITH SYNTAX                        DirectoryString {ub-organization-name}
        ID                                 id-at-organizationName }

collectiveOrganizationName ATTRIBUTE              ::= {
        SUBTYPE OF                 organizationName
        COLLECTIVE                 TRUE
        ID                                 id-at-collectiveOrganizationName }

organizationalUnitName ATTRIBUTE       ::=      {
        SUBTYPE OF                 name
        WITH SYNTAX                        DirectoryString {ub-organizational-unit-name}
        ID                                 id-at-organizationalUnitName }

collectiveOrganizationalUnitName ATTRIBUTE    ::=          {
        SUBTYPE OF                 organizationalUnitName
        COLLECTIVE                 TRUE
        ID                                 id-at-collectiveOrganizationalUnitName }

title ATTRIBUTE ::=                        {
        SUBTYPE OF                 name
        WITH SYNTAX                        DirectoryString {ub-title}
        ID                                 id-at-title }

description ATTRIBUTE ::=                  {
        WITH SYNTAX                        DirectoryString {ub-description}
        EQUALITY MATCHING RULE             caseIgnoreMatch
        SUBSTRINGS MATCHING RULE           caseIgnoreSubstringsMatch
        ID                                 id-at-description }

searchGuide ATTRIBUTE          ::=                 {
        WITH SYNTAX                                Guide
        ID                                         id-at-searchGuide }

Guide    ::=      SET  {
        objectClass                [0]     OBJECT-CLASS.&id OPTIONAL,
        criteria                   [1]     Criteria  }

Criteria  ::=              CHOICE {
        type                       [0]     CriteriaItem,
        and                        [1]     SET OF Criteria,
        or                         [2]     SET OF Criteria,
        not                        [3]     Criteria}

CriteriaItem ::= CHOICE {
        equality                   [0]     AttributeType,
        substrings                 [1]     AttributeType,
        greaterOrEqual             [2]     AttributeType,
        lessOrEqual                [3]     AttributeType,
        approximateMatch           [4]     AttributeType}

enhancedSearchGuide ATTRIBUTE                 ::=          {
        WITH SYNTAX                                EnhancedGuide
        ID                                         id-at-enhancedSearchGuide }

EnhancedGuide    ::=           SEQUENCE {
        objectClass                [0]     OBJECT-CLASS.&id,
        criteria                   [1]     Criteria,
        subset                     [2]     INTEGER
                { baseObject (0), oneLevel (1), wholeSubtree (2) }  DEFAULT oneLevel }

businessCategory ATTRIBUTE  ::=            {
        WITH SYNTAX                        DirectoryString {ub-business-category}
        EQUALITY MATCHING RULE             caseIgnoreMatch
        SUBSTRINGS MATCHING RULE           caseIgnoreSubstringsMatch
        ID                                 id-at-businessCategory }
```

```
postalAddress ATTRIBUTE      ::=          {
      WITH SYNTAX                         PostalAddress
      EQUALITY MATCHING RULE              caseIgnoreListMatch
      SUBSTRINGS MATCHING RULE            caseIgnoreListSubstringsMatch
      ID                                  id-at-postalAddress }

PostalAddress      ::=          SEQUENCE SIZE(1..ub-postal-line) OF  DirectoryString {ub-postal-string}

collectivePostalAddress ATTRIBUTE        ::=      {
      SUBTYPE OF                 postalAddress
      COLLECTIVE                 TRUE
      ID                                  id-at-collectivePostalAddress }

postalCode ATTRIBUTE         ::=          {
      WITH SYNTAX                         DirectoryString {ub-postal-code}
      EQUALITY MATCHING RULE              caseIgnoreMatch
      SUBSTRINGS MATCHING RULE            caseIgnoreSubstringsMatch
      ID                                  id-at-postalCode }

collectivePostalCode ATTRIBUTE           ::=      {
      SUBTYPE OF                 postalCode
      COLLECTIVE                 TRUE
      ID                                  id-at-collectivePostalCode }

postOfficeBox ATTRIBUTE      ::=              {
      WITH SYNTAX                         DirectoryString {ub-post-office-box}
      EQUALITY MATCHING RULE              caseIgnoreMatch
      SUBSTRINGS MATCHING RULE            caseIgnoreSubstringsMatch
      ID                                  id-at-postOfficeBox }

collectivePostOfficeBox ATTRIBUTE        ::= {
      SUBTYPE OF                 postOfficeBox
      COLLECTIVE                 TRUE
      ID                                  id-at-collectivePostOfficeBox }

physicalDeliveryOfficeName ATTRIBUTE       ::=  {
      WITH SYNTAX                         DirectoryString {ub-physical-office-name}
      EQUALITY MATCHING RULE              caseIgnoreMatch
      SUBSTRINGS MATCHING RULE            caseIgnoreSubstringsMatch
      ID                                  id-at-physicalDeliveryOfficeName }

collectivePhysicalDeliveryOfficeName  ATTRIBUTE  ::=    {
      SUBTYPE OF                 physicalDeliveryOfficeName
      COLLECTIVE                 TRUE
      ID                                  id-at-collectivePhysicalDeliveryOfficeName }

telephoneNumber ATTRIBUTE  ::=           {
      WITH SYNTAX                         TelephoneNumber
      EQUALITY MATCHING RULE              telephoneNumberMatch
      SUBSTRINGS MATCHING RULE            telephoneNumberSubstringsMatch
      ID                                  id-at-telephoneNumber }

TelephoneNumber  ::= PrintableString (SIZE(1..ub-telephone-number))
                -- String complying with CCITT Recommendation E.123 only

collectiveTelephoneNumber    ATTRIBUTE      ::=      {
      SUBTYPE OF                 telephoneNumber
      COLLECTIVE                 TRUE
      ID                                  id-at-collectiveTelephoneNumber }

telexNumber ATTRIBUTE  ::=  {
      WITH SYNTAX                         TelexNumber
      ID                                  id-at-telexNumber }

TelexNumber ::= SEQUENCE {
      telexNumber          PrintableString (SIZE(1..ub-telex-number)),
      countryCode          PrintableString (SIZE(1..ub-country-code)),
      answerback                PrintableString (SIZE(1..ub-answerback))}

collectiveTelexNumber   ATTRIBUTE        ::=      {
      SUBTYPE OF                 telexNumber
      COLLECTIVE                 TRUE
      ID                                  id-at-collectiveTelexNumber }
```

```
facsimileTelephoneNumber ATTRIBUTE          ::=  {
    WITH SYNTAX                             FacsimileTelephoneNumber
    ID                                      id-at-facsimileTelephoneNumber }

FacsimileTelephoneNumber    ::=             SEQUENCE {
    telephoneNumber         TelephoneNumber
    parameters              G3FacsimileNonBasicParameters  OPTIONAL}

collectiveFacsimileTelephoneNumber      ATTRIBUTE ::=     {
    SUBTYPE OF              facsimileTelephoneNumber
    COLLECTIVE              TRUE
    ID                                      id-at-collectiveFacsimileTelephoneNumber }

x121Address ATTRIBUTE  ::=  {
    WITH SYNTAX                             X.121Address
    EQUALITY MATCHING RULE                  numericStringMatch
    SUBSTRINGS MATCHING RULE                numericStringSubstringsMatch
    ID                                      id-at-x121Address }

X121Address  ::=  NumericString (SIZE(1  ub-x121-address))
            -- String as defined by ITU-T Recommendation X.121

internationalISDNNumber ATTRIBUTE    ::= {
    WITH SYNTAX                             InternationalISDNNumber
    EQUALITY MATCHING RULE                  numericStringMatch
    SUBSTRINGS MATCHING RULE                numericStringSubstringsMatch
    ID                                      id-at-internationalISDNNumber }

InternationalISDNNumber ::=NumericString (SIZE(1..ub-international-isdn-number))
            -- String complying with ITU-T Recommendation E.164 only

collectiveInternationalISDNNumber ATTRIBUTE    ::=    {
    SUBTYPE OF              internationalISDNNumber
    COLLECTIVE              TRUE
    ID                                      id-at-collectiveInternationalISDNNumber }

registeredAddress ATTRIBUTE  ::=        {
    SUBTYPE OF              postalAddress
    WITH SYNTAX                             PostalAddress
    ID                                      id-at-registeredAddress }

destinationIndicator ATTRIBUTE  ::=     {
    WITH SYNTAX                             DestinationIndicator
    EQUALITY MATCHING RULE                  caseIgnoreMatch
    SUBSTRINGS MATCHING RULE                caseIgnoreSubstringsMatch
    ID                                      id-at-destinationIndicator }

DestinationIndicator  ::=                   PrintableString (SIZE (1..ub-destination-indicator))
                                            -- alphabetical characters only

preferredDeliveryMethod ATTRIBUTE       ::=     {
WITH SYNTAX                             SEQUENCE OF INTEGER {
        any-delivery-method                     (0),
        mhs-delivery                            (1),
        physical-delivery                       (2),
        telex-delivery                          (3),
        teletex-delivery                        (4),
        g3-facsimile-delivery                   (5),
        g4-facsimile-delivery                   (6),
        ia5-terminal-delivery                   (7),
        videotex-delivery                       (8),
        telephone-delivery                      (9) }

    SINGLE VALUE                            TRUE
    ID                                      id-at-preferredDeliveryMethod }
```

```
presentationAddress ATTRIBUTE          ::=     {
        WITH SYNTAX                            PresentationAddress
        EQUALITY MATCHING RULE         presentationAddressMatch
        SINGLE VALUE                           TRUE
        ID                                     id-at-presentationAddress }

PresentationAddress ::= SEQUENCE {
        pSelector       [0]     OCTET STRING OPTIONAL,
        sSelector       [1]     OCTET STRING OPTIONAL,
        tSelector       [2]     OCTET STRING OPTIONAL,
        nAddresses      [3]     SET SIZE (1..MAX)  OF  OCTET STRING}

supportedApplicationContext ATTRIBUTE     ::= {
        WITH SYNTAX                            OBJECT IDENTIFIER
        EQUALITY MATCHING RULE         objectIdentifierMatch
        ID                                     id-at-supportedApplicationContext }

protocolInformation ATTRIBUTE::=       {
        WITH SYNTAX                            ProtocolInformation
        EQUALITY MATCHING RULE         protocolInformationMatch
        ID                                     id-at-protocolInformation }

ProtocolInformation  ::=        SEQUENCE {
        nAddress                       OCTET STRING,
        profiles                       SET OF OBJECT IDENTIFIER }

distinguishedName ATTRIBUTE ::=        {
        WITH SYNTAX                            DistinguishedName
        EQUALITY MATCHING RULE         distinguishedNameMatch
        ID                                     id-at-distinguishedName }

member ATTRIBUTE       ::=     {
        SUBTYPE OF                     distinguishedName
        ID                                     id-at-member }

uniqueMember ATTRIBUTE     ::=                 {
        WITH SYNTAX                            NameAndOptionalUID
        EQUALITY MATCHING RULE         uniqueMemberMatch
        ID                                     id-at-uniqueMember }

NameAndOptionalUID    ::=      SEQUENCE {
        dn                             DistinguishedName,
        uid                            UniqueIdentifier OPTIONAL }

owner ATTRIBUTE        ::=     {
        SUBTYPE OF                     distinguishedName
        ID                                     id-at-owner }

roleOccupant ATTRIBUTE ::=  {
        SUBTYPE OF                     distinguishedName
        ID                                     id-at-roleOccupant }

seeAlso ATTRIBUTE      ::=     {
        SUBTYPE OF                     distinguishedName
        ID                                     id-at-seeAlso }

dmdName ATTRIBUTE  ::=         {
        SUBTYPE OF                     name
        WITH SYNTAX                    DirectoryString{ub-common-name}
        ID                             id-at-dmdName }
```

-- Matching rules --

```
caseIgnoreMatch MATCHING-RULE          ::=     {
        SYNTAX                         DirectoryString {ub-match}
        ID                             id-mr-caseIgnoreMatch }

caseIgnoreOrderingMatch MATCHING-RULE      ::=     {
        SYNTAX                         DirectoryString {ub-match}
        ID                             id-mr-caseIgnoreOrderingMatch }

caseIgnoreSubstringsMatch MATCHING-RULE    ::=             {
        SYNTAX                         SubstringAssertion
        ID                             id-mr-caseIgnoreSubstringsMatch }
```

```
SubstringAssertion ::= SEQUENCE OF CHOICE {
    initial         [0]     DirectoryString {ub-match},
    any             [1]     DirectoryString {ub-match},
    final           [2]     DirectoryString {ub-match} }
    -- at most one initial and one final component

caseExactMatch MATCHING-RULE        ::=     {
    SYNTAX                      DirectoryString {ub-match}
    ID                          id-mr-caseExactMatch }

caseExactOrderingMatch MATCHING-RULE      ::=     {
    SYNTAX                      DirectoryString {ub-match}
    ID                          id-mr-caseExactOrderingMatch }

caseExactSubstringsMatch MATCHING-RULE    ::=     {
    SYNTAX                      SubstringAssertion -- only the PrintableString choice
    ID                          id-mr-caseExactSubstringsMatch }

numericStringMatch MATCHING-RULE  ::=  {
    SYNTAX                      NumericString
    ID                          id-mr-numericStringMatch }

numericStringOrderingMatch MATCHING-RULE    ::=     {
    SYNTAX                      NumericString
    ID                          id-mr-numericStringOrderingMatch }

numericStringSubstringsMatch MATCHING-RULE          ::=         {
    SYNTAX                      SubstringAssertion
    ID                          id-mr-numericStringSubstringsMatch }

caseIgnoreListMatch MATCHING-RULE   ::= {
    SYNTAX                      SEQUENCE OF DirectoryString {ub-match}
    ID                          id-mr-caseIgnoreListMatch }

caseIgnoreListSubstringsMatch MATCHING-RULE       ::=     {
    SYNTAX                      SubstringAssertion
    ID                          id-mr-caseIgnoreListSubstringsMatch }

storedPrefixMatch MATCHING-RULE  ::=  {
    SYNTAX                      DirectoryString {ub-match}
    ID                          id-mr-storedPrefixMatch }

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

integerMatch MATCHING-RULE   ::=    {
    SYNTAX                          INTEGER
    ID                              id-mr-integerMatch }

integerOrderingMatch MATCHING-RULE ::=     {
    SYNTAX                      INTEGER
    ID                          id-mr-integerOrderingMatch }

bitStringMatch MATCHING-RULE::=     {
    SYNTAX                          BIT STRING
    ID                              id-mr-bitStringMatch }

octetStringMatch MATCHING-RULE   ::=   {
    SYNTAX                      OCTET STRING
    ID                          id-mr-octetStringMatch }

octetStringOrderingMatch MATCHING-RULE ::=  {
    SYNTAX                      OCTET STRING
    ID                          id-mr-octetStringOrderingMatch }

octetStringSubstringsMatch MATCHING-RULE   ::=   {
    SYNTAX                      OctetSubstringAssertion
    ID                          id-mr-octetStringSubstringsMatch }

OctetSubstringAssertion ::= SEQUENCE OF CHOICE {
    initial         [0]     OCTET STRING,
    any             [1]     OCTET STRING,
    final           [2]     OCTET STRING }
    -- at most one initial and one final component
```

```
telephoneNumberMatch MATCHING-RULE  ::=  {
        SYNTAX                  PrintableString
        ID                      id-mr-telephoneNumberMatch }

telephoneNumberSubstringsMatch MATCHING-RULE    ::=     {
        SYNTAX                  SubstringAssertion
        ID                      id-mr-telephoneNumberSubstringsMatch }

presentationAddressMatch MATCHING-RULE     ::=      {
        SYNTAX                  PresentationAddress
        ID                      id-mr-presentationAddressMatch }

-- Contexts --

languageContext CONTEXT ::=  {
        WITH SYNTAX             LanguageContextSyntax
        ID                      id-avc-language

LanguageContextSyntax  ::=  PrintableString (SIZE(2..3))  -- ISO 639-2 codes only

temporalContext CONTEXT  ::=  {
        WITH SYNTAX             TimeSpecification
        ASSERTED AS             TimeAssertion
        ID                      id-avc-temporal }

TimeSpecification ::= SEQUENCE {
        time                    CHOICE {
          absolute                      SEQUENCE {
                startTime                       [0]     GeneralizedTime OPTIONAL,
                endTime                         [1]     GeneralizedTime OPTIONAL },
          periodic                      SET OF Period },
        notThisTime             BOOLEAN DEFAULT FALSE,
        timeZone                TimeZone OPTIONAL }

Period ::= SEQUENCE {
        timesOfDay              [0]     SET OF DayTimeBand OPTIONAL,
        days                    [1]     CHOICE {
                                            intDay      SET OF INTEGER,
                                            bitDay      BIT STRING { sunday (0), monday (1), tuesday (2),
                                                        wednesday (3), thursday (4), friday (5), saturday (6),
                                            dayOf       XDayOf } OPTIONAL,
        weeks                   [2]     CHOICE {
                                            allWeeks            NULL,
                                            intWeeks            SET OF INTEGER,
                                            bitWeeks            BIT STRING {week1 (0), week2 (1), week3 (2), week4 (3),
                                                                week5 (4) } } OPTIONAL,
        months                  [3]     CHOICE {
                                            allMonths           NULL,
                                            intMonth            SET OF INTEGER,
                                            bitMonth            BIT STRING { january (0), february (1), march (2), april (3),
                                                                may (4), june (5), july (6), august (7), september (8),
                                                                october (9), november (10), december (11) }
                                        } OPTIONAL,
        years                   [4]     SET OF INTEGER (1000 .. MAX) OPTIONAL }

XDayOf  ::=  CHOICE {
        first           [1]             NamedDay,
        second          [2]             NamedDay,
        third           [3]             NamedDay,
        fourth          [4]             NamedDay,
        fifth           [5]             NamedDay}

NamedDay ::= CHOICE {
                intNamedDays            ENUMARATED  {
                        sunday                      (1),
                        monday                      (2),
                        tuesday                     (3),
```