INTERNATIONAL STANDARD

ISO/IEC 9594-6

Seventh edition
2014-03-01

Information technology — Open Systems Interconnection — The Directory —

Part 6: Selected attribute types

Technologies de l'information — Interconnexion de systèmes ouverts (OSI) — L'annuaire

Partie 6: Types d'attributs sélectionnés

Reference number
ISO/IEC 9594-6:2014(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.org
Web www.iso.org

Published in Switzerland

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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. 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 document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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 Rec. ITU-T X.520 (10/2012).

This seventh edition cancels and replaces the sixth edition (ISO/IEC 9594-6:2008), which has been technically revised. It also incorporates the Technical Corrigenda ISO/IEC 9594-6:2008/Cor.1:2011, ISO/IEC 9594-6:2008/Cor.2:2012 and ISO/IEC 9594-6:2008/Cor.3:2013.

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: Public-key and attribute certificate frameworks*

— *Part 9: Replication*

CONTENTS

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 that 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 objects defined in Rec. ITU-T X.521 | ISO/IEC 9594-7.

This Recommendation | International Standard provides the foundation frameworks upon which industry profiles can be defined by other standards groups and industry forums. Many of the features defined as optional in these frameworks may be mandated for use in certain environments through profiles. This seventh edition technically revises and enhances the sixth edition of this Recommendation | International Standard.

This seventh edition specifies versions 1 and 2 of the Directory protocols.

The first and second editions specified only version 1. Most of the services and protocols specified in this edition are designed to function under version 1. 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. Whichever version has been negotiated, differences between the services and between the protocols defined in the six editions, except for those specifically assigned to version 2, are accommodated using the rules of extensibility defined in Rec. ITU-T X.519 | ISO/IEC 9594-5.

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 an example of upper bounds value constraints. These constraints are not reflected in these Directory Specifications, but are provided as a reference for those implementations applying these constraints.

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, gives examples relevant to the definition of zonal matching.

Annex F, which is not an integral part of this Recommendation | International Standard, describes how a directory distinguished name may be based on object identifiers and on Uniform Resource Names (URNs).

Annex G, which is not an integral part of this Recommendation | International Standard, describes an alternative way of generating directory distinguished based on object identifiers. It contains information retrieved from Rec. ITU-T X.660 | ISO/IEC 9834-1.

Annex H, 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
RECOMMENDATION ITU-T

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

NOTE 2 – The attribute and context types definitions by this Recommendation | International Standard have some associated semantics. Such specifications should not be used in situations where these semantics do not apply.

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 through 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

- Recommendation ITU-T X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The basic model.*
- Recommendation ITU-T X.500 (2012) | ISO/IEC 9594-1:2014, *Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services.*
- Recommendation ITU-T X.501 (2012) | ISO/IEC 9594-2:2014, *Information technology – Open Systems Interconnection – The Directory: Models.*
- Recommendation ITU-T X.509 (2012) | ISO/IEC 9594-8:2014, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks.*
- Recommendation ITU-T X.511 (2012) | ISO/IEC 9594-3:2014, *Information technology – Open Systems Interconnection – The Directory: Abstract service definition.*
- Recommendation ITU-T X.518 (2012) | ISO/IEC 9594-4:2014, *Information technology – Open Systems Interconnection – The Directory: Procedures for distributed operation.*
- Recommendation ITU-T X.519 (2012) | ISO/IEC 9594-5:2014, *Information technology – Open Systems Interconnection – The Directory: Protocol specifications.*
- Recommendation ITU-T X.521 (2012) | ISO/IEC 9594-7:2014, *Information technology – Open Systems Interconnection – The Directory: Selected object classes.*

- Recommendation ITU-T X.525 (2012) | ISO/IEC 9594-9:2014, *Information technology – Open Systems Interconnection – The Directory: Replication.*
- Recommendation ITU-T X.660 (2008) | ISO/IEC 9834-1:2008, *Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities: General procedures and top arcs of the International Object Identifier tree.*
- Recommendation ITU-T X.667 (2008) | ISO/IEC 9834-8:2008, *Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 object identifier components.*
- Recommendation ITU-T X.668 (2008) | ISO/IEC 9834-9:2008, *Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities: Registration of object identifier arcs for applications and services using tag-based identification.*
- Recommendation ITU-T X.680 (2008) | ISO/IEC 8824-1:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- Recommendation ITU-T X.681 (2008) | ISO/IEC 8824-2:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*
- Recommendation ITU-T X.682 (2008) | ISO/IEC 8824-3:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*
- Recommendation ITU-T X.683 (2008) | ISO/IEC 8824-4:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.*

2.2 Other references

- Recommendation ITU-T E.123 (2001), *Notation for national and international telephone numbers, e-mail addresses and web addresses.*
- Recommendation ITU-T E.164 (2005), *The international public telecommunication numbering plan.*
- Recommendation ITU-T F.1 (1998), *Operational provisions for the international public telegram service.*
- Recommendation CCITT F.31 (1988), *Telegram retransmission system.*
- Recommendation CCITT F.401 (1992), *Message handling services: Naming and addressing for public message handling services.*
- Recommendation ITU-T T.30 (2005), *Procedures for document facsimile transmission in the general switched telephone network.*
- Recommendation ITU-T T.51 (1992), *Latin based coded character sets for telematic services.*
- Recommendation ITU-T T.62 (1993), *Control procedures for teletex and Group 4 facsimile services.*
- Recommendation ITU-T X.121 (2000), *International numbering plan for public data networks.*
- Recommendation ITU-T Y.2213 (2008), *NGN service requirements and capabilities for network aspects of applications and services using tag-based identification.*
- ISO 3166-1:2006, *Codes for the representation of names of countries and their subdivisions – Part 1: Country codes.*
- ISO 3166-3:1999, *Codes for the representation of names of countries and their subdivisions – Part 3: Code for formerly used names of countries.*
- ISO 639-2:1998, *Codes for the representation of names of languages – Part 2: Alpha-3 code.*
- ISO/IEC/IEEE 9945:2009, *Information technology – Portable Operating System Interface (POSIX) – Base Specifications, Issue 7.*
- ISO/IEC 15897:2001, *Information technology – User interfaces – Procedures for the registration of cultural elements.*
- IETF RFC 3406 (2002), *Uniform Resource Names (URN) Namespace Definition Mechanisms.*
- IETF RFC 3454 (2003), *Preparation of Internationalized Strings ("stringprep").*
- IETF RFC 3641 (2003), *Generic String Encoding Rules (GSER) for ASN.1 Types.*
- IETF RFC 3642 (2003), *Common Elements of Generic String Encoding Rules (GSER) Encodings.*
- IETF RFC 3672 (2003), *Subentries in the Lightweight Directory Access Protocol (LDAP).*
- IETF RFC 3986 (2005), *Uniform Resource Identifier (URI): Generic Syntax.*

- IETF RFC 4510 (2006), *Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map*.
- IETF RFC 4512 (2006), *Lightweight Directory Access Protocol (LDAP): Directory Information Models*.
- IETF RFC 4514 (2006); *Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names*.
- IETF RFC 4517 (2006), *Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules*.
- IETF RFC 4519 (2006), *Lightweight Directory Access Protocol (LDAP): Schema for User Applications*.
- IETF RFC 4520 (2006), *Internet Assigned Numbers Authority (IANA) Considerations for the Lightweight Directory Access Protocol (LDAP)*.
- IETF RFC 4792 (2007), *Encoding Instructions for the Generic String Encoding Rules (GSER)*.
- The Unicode Consortium. *The Unicode Standard, Version 4.0.0*, defined by: *The Unicode Standard, Version 4.0* (Reading, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1).
- *Unicode Standard Annex #15: Unicode Normalization Forms*, by Mark Davis and Martin Dürst. An integral part of *The Unicode Standard, Version 4.0*.
- National Imagery and Mapping Agency (NIMA): TR 8350.2, DoD Word Geodetic System 1984.

2.3 ISO/IEC Standards

- ISO/IEC 10646:2012, *Information technology – Universal Coded Character Set (UCS)*.

3 Definitions

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

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

a) *attribute type*;

b) *context*;

c) *matching rule*;

d) *object class* .

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply:

AFI	Application Family Identifier
EPC	Electronic Product Code
GSER	Generic String Encoding Rules
LDAP	Lightweight Directory Access Protocol
RFID	Radio Frequency Identification
RDN	Relative Distinguished Name
UII	Unique Item Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UTM	Universal Transverse Mercator
UUID	Universally Unique Identifier

5 Conventions

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

This Directory Specification uses the term *first edition systems* to refer to systems conforming to the first 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 *second edition systems* to refer to systems conforming to the second 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.

This Directory Specification uses the term *third edition systems* to refer to systems conforming to the third edition of the Directory Specifications, i.e., the 1997 edition of the series of ITU-T X.500 Recommendations and the ISO/IEC 9594:1998 edition.

This Directory Specification uses the term *fourth edition systems* to refer to systems conforming to the fourth edition of the Directory Specifications, i.e., the 2001 editions of Recs ITU-T X.500, X.501, X.511, X.518, X.519, X.520, X.521, X.525, and X.530, the 2000 edition of Rec. ITU-T X.509, and parts 1-10 of the ISO/IEC 9594:2001 edition.

This Directory Specification uses the term *fifth edition systems* to refer to systems conforming to the fifth edition of the Directory Specifications, i.e., the 2005 edition of the series of ITU-T X.500 Recommendations and the ISO/IEC 9594:2005 edition.

This Directory Specification uses the term *sixth edition systems* to refer to systems conforming to the sixth edition of the Directory Specifications, i.e., the 2008 edition of the series of ITU-T X.500 Recommendations and the ISO/IEC 9594:2008 edition.

This Directory Specification uses the term *seventh edition systems* to refer to systems conforming to the seventh edition of these Directory Specifications, i.e., the 2012 edition of the series of ITU-T X.500 Recommendations and the ISO/IEC 9594:2014 edition.

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

If the items in a list are numbered (as opposed to using "–" or letters), then the items shall be considered steps in a procedure.

Attribute types, matching rules and context types are defined in this Recommendation | International Standard by use of the **ATTRIBUTE**, **MATCHING-RULE** and **CONTEXT** information object classes defined in Rec. ITU-T 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

6 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 attribute types defined in this Directory Specification are based on a common ASN.1 syntax:

```
UnboundedDirectoryString ::= CHOICE {
  teletexString     TeletexString(SIZE (1..MAX)),
  printableString   PrintableString(SIZE (1..MAX)),
  bmpString         BMPString(SIZE (1..MAX)),
  universalString   UniversalString(SIZE (1..MAX)),
  uTF8String        UTF8String(SIZE (1..MAX)) }
```

A few attribute types are based on the following data 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,...)),
  uTF8String        UTF8String(SIZE (1..maxSize,...)) }
```

NOTE 1 – The above syntaxes are also used in other parts of these Directory Specifications.

NOTE 2 – The use of **TeletexString** is deprecated.

6.1 System attribute types

6.1.1 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               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  OBSOLETE                  TRUE
  ID                        id-at-knowledgeInformation }
```

6.2 Labelling attribute types

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

6.2.1 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               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE  caseIgnoreSubstringsMatch
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"name"}
  ID                        id-at-name }
```

6.2.2 Common Name

An attribute of the type **commonName** specifies an identification of an object. A Common Name is not a directory name in itself; 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 by either 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               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"cn","commonName"}
  ID                        id-at-commonName }
```

6.2.3 Surname

An attribute of the type **surname** 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               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"sn"}
  ID                        id-at-surname }
```

6.2.4 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               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"givenName"}
  ID                        id-at-givenName }
```

6.2.5 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               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"initials"}
  ID                        id-at-initials }
```

6.2.6 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               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"generationQualifier"}
  ID                        id-at-generationQualifier }
```

6.2.7 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
  LDAP-SYNTAX               bitString.&id
  LDAP-NAME                 {"x500UniqueIdentifier"}
  ID                        id-at-uniqueIdentifier }

UniqueIdentifier ::= BIT STRING
```

6.2.8 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
  LDAP-SYNTAX               printableString.&id
  LDAP-NAME                 {"dnQualifier"}
  ID                        id-at-dnQualifier }
```

6.2.9 Serial Number

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

An attribute value for Serial Number is a printable string.

```
serialNumber ATTRIBUTE ::= {
  WITH SYNTAX               PrintableString(SIZE (1..MAX))
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               printableString.&id
  LDAP-NAME                 {"serialNumber"}
  ID                        id-at-serialNumber }
```

6.2.10 Pseudonym

The *Pseudonym* attribute type specifies a pseudonym for an object. It is used for naming an object when it is to be made clear that its name is a pseudonym.

```
pseudonym ATTRIBUTE ::= {
  SUBTYPE OF                name
  WITH SYNTAX               UnboundedDirectoryString
  ID                        id-at-pseudonym }
```

6.2.11 Universal Unique Identifier Pair

The *Universal Unique Identifier Pair* attribute type specifies a pair of Universal Unique Identifiers (UUID), as specified in Rec. ITU-T X.667 | ISO/IEC 9834-8. The pair collectively represents an issuer/subject relationship, the nature of which is outside the scope of this Directory Specification. The initial UUID in the pair represents the issuer, and the

trailing UUID in the pair represents the subject of the issuer/subject relationship. An example of such a relationship is a user account.

```
uUIDPair ATTRIBUTE ::= {
  WITH SYNTAX              UUIDPair
  EQUALITY MATCHING RULE   uUIDPairMatch
  ID                       id-at-uuidpair }

UUIDPair ::= SEQUENCE {
  issuerUUID   UUID,
  subjectUUID  UUID,
  ... }

UUID ::= OCTET STRING(SIZE (16)) -- UUID format only
```

6.2.12 uri

The *uri* attribute type is used for holding a Uniform Resource Identifier (URI) as defined in IETF RFC 3986.

```
uri ATTRIBUTE ::= {
  WITH SYNTAX                URI
  EQUALITY MATCHING RULE     uriMatch
  LDAP-SYNTAX                directoryString.&id
  LDAP-NAME                  {"uri"}
  ID                         id-at-uri }

URI ::= UTF8String
```

6.2.13 URN

The *URN* attribute type is used for holding a Uniform Resource Name (URN) as defined in IETF RFC 3406.

```
urn ATTRIBUTE ::= {
  SUBTYPE OF                 uri
  LDAP-SYNTAX                directoryString.&id
  LDAP-NAME                  {"urn"}
  ID                         id-at-urn }
```

6.2.14 URL

The *URL* attribute type is used for holding a Uniform Resource Locator (URL).

```
url ATTRIBUTE ::= {
  SUBTYPE OF                 uri
  LDAP-SYNTAX                directoryString.&id
  LDAP-NAME                  {"url"}
  ID                         id-at-url }
```

6.3 Geographical attribute types

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

6.3.1 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 3166-1 alpha-2 or ISO 3166-3 alpha-2.

```
countryName ATTRIBUTE ::= {
  SUBTYPE OF                 name
  WITH SYNTAX                CountryName
  SINGLE VALUE               TRUE
  LDAP-SYNTAX                countryString.&id
  LDAP-NAME                  {"c"}
  ID                         id-at-countryName }

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

6.3.2 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               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"l"}
  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 }
```

6.3.3 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               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"st"}
  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 }
```

6.3.4 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 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               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"street"}
  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 }
```

6.3.5 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               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"houseIdentifier"}
  ID                        id-at-houseIdentifier }
```

6.3.6 UTM Coordinates attribute type

An attribute of type **utmCoordinates** gives the coordinates in the Universal Transverse Mercator (UTM) coordinate system.

```
utmCoordinates  ATTRIBUTE  ::=  {
  WITH SYNTAX               UtmCoordinates
  SINGLE VALUE              TRUE
  ID                        id-at-utmCoordinates }

UtmCoordinates ::= SEQUENCE {
zone       PrintableString,
easting    NumericString,
northing   NumericString }
```

The **zone** component gives the value of the UTM zone. It consists of a single letter followed by up to two numeric characters.

The **easting** component gives the easting values in metres.

The **northing** component gives the northing value in metres.

6.4 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.

6.4.1 Organization Name

The *Organization Name* 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               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"o"}
  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 }
```

6.4.2 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 shall 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               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"ou"}
  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 }
```

6.4.3 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               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"title"}
  ID                        id-at-title }
```

6.4.4 Organization Identifier

An attribute of type **organizationIdentifier** holds an identification of an organization different from the organization name.

```
organizationIdentifier ATTRIBUTE ::= {
  WITH SYNTAX               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"organizationIdentifier"}
  ID                        id-at-organizationIdentifier }
```

6.5 Explanatory attribute types

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

6.5.1 Description

The *Description* attribute type specifies text that 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               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"description"}
  ID                        id-at-description }
```

6.5.2 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
  LDAP-SYNTAX               guide.&id
  LDAP-NAME                 {"searchGuide"}
  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 }
```

6.5.3 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
  LDAP-SYNTAX               enhancedGuide.&id
  LDAP-NAME                 {"enhancedSearchGuide"}
  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,
  ... }
```

6.5.4 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               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"businessCategory"}
  ID                        id-at-businessCategory }
```

6.6 Postal addressing attribute types

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

6.6.1 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 Rec. 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
  LDAP-SYNTAX               postalAddr.&id
  LDAP-NAME                 {"postalAddress"}
  ID                        id-at-postalAddress }
PostalAddress ::= SEQUENCE SIZE (1..MAX) OF UnboundedDirectoryString
```

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 }
```

6.6.2 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               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"postalCode"}
  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 }
```

6.6.3 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               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"postOfficeBox"}
  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 }
```

6.6.4 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               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"physicalDeliveryOfficeName"}
  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 }
```

6.7 Telecommunications addressing attribute types

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

6.7.1 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 shall comply with the internationally agreed format for showing international telephone numbers, as indicated in clause 2.5 of Rec. ITU-T E.123 (e.g., "+ 44 582 10101"). However, it is

allowed to insert hyphens (-) in addition to the + sign, spaces and figures. Other characters from the **PrintableString** repertoire shall not be used.

```
telephoneNumber ATTRIBUTE ::= {
  WITH SYNTAX               TelephoneNumber
  EQUALITY MATCHING RULE    telephoneNumberMatch
  SUBSTRINGS MATCHING RULE telephoneNumberSubstringsMatch
  LDAP-SYNTAX               printableString.&id
  LDAP-NAME                 {"telephoneNumber"}
  ID                        id-at-telephoneNumber }

TelephoneNumber ::= PrintableString(SIZE (1..ub-telephone-number))
-- String complying with Rec. ITU-T E.123 only

ub-telephone-number INTEGER ::= 32
```

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 }
```

6.7.2 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
  LDAP-SYNTAX               telexNr.&id
  LDAP-NAME                 {"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)),
  ... }

ub-telex-number INTEGER ::= 14
ub-country-code INTEGER ::= 4
ub-answerback   INTEGER ::= 8
```

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 }
```

6.7.3 Teletex Terminal Identifier

*Since CCITT Rec. 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 Rec. F.200 and an optional set whose components are according to Rec. ITU-T 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 }
```

6.7.4 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, Rec. ITU-T E.123 (e.g., "+81 3 347 7418") and an optional bit string (formatted according to Rec. ITU-T T.30).

```
facsimileTelephoneNumber ATTRIBUTE ::= {
  WITH SYNTAX               FacsimileTelephoneNumber
  EQUALITY MATCHING RULE    facsimileNumberMatch
  SUBSTRINGS MATCHING RULE facsimileNumberSubstringsMatch
  LDAP-SYNTAX               facsimileTelephoneNr.&id
  LDAP-NAME                 {"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 }
```

6.7.5 X.121 Address

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

```
x121Address ATTRIBUTE ::= {
  WITH SYNTAX               X121Address
  EQUALITY MATCHING RULE    numericStringMatch
  SUBSTRINGS MATCHING RULE numericStringSubstringsMatch
  LDAP-SYNTAX               numericString.&id
  LDAP-NAME                 {"x121Address"}
  ID                        id-at-x121Address }

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

ub-x121-address INTEGER ::= 15
```

6.7.6 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 Rec. ITU-T E.164.

```
internationalISDNNumber ATTRIBUTE ::= {
  WITH SYNTAX               InternationalISDNNumber
  EQUALITY MATCHING RULE    numericStringMatch
  SUBSTRINGS MATCHING RULE numericStringSubstringsMatch
  LDAP-SYNTAX               numericString.&id
  LDAP-NAME                 {"internationalISDNNumber"}
  ID                        id-at-internationalISDNNumber }

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

```
ub-international-isdn-number INTEGER ::= 16
```

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 }
```

6.7.7 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 Rec. ITU-T F.1).

```
registeredAddress ATTRIBUTE ::= {
  SUBTYPE OF                postalAddress
  WITH SYNTAX               PostalAddress
  LDAP-SYNTAX               postalAddr.&id
  LDAP-NAME                 {"registeredAddress"}
  ID                        id-at-registeredAddress }
```

6.7.8 Destination Indicator

The *Destination Indicator* attribute type specifies (according to Rec. ITU-T 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
  LDAP-SYNTAX               printableString.&id
  LDAP-NAME                 {"destinationIndicator"}
  ID                        id-at-destinationIndicator }

DestinationIndicator ::= PrintableString(SIZE (1..MAX))
-- alphabetical characters only
```

6.7.9 Communications Service

The *Communications Service* attribute type specifies the type of service(s) associated with a communications address.

```
communicationsService ATTRIBUTE ::= {
  WITH SYNTAX               CommunicationsService
  EQUALITY MATCHING RULE    objectIdentifierMatch
  ID                        id-at-communicationsService }

CommunicationsService ::= OBJECT IDENTIFIER
```

This attribute describes the class of service that the Communications Address provides access to, for example, telephone (voice), facsimile, electronic mail, SMS (short messaging service), EDI, file transfer, etc.

Allocation of object identifiers for the identification of services is done outside this Directory Specification.

6.7.10 Communications Network

The *Communications Network* attribute type specifies the type of network for which a communications address is used.

```
communicationsNetwork ATTRIBUTE ::= {
  WITH SYNTAX               CommunicationsNetwork
  EQUALITY MATCHING RULE    objectIdentifierMatch
  SINGLE VALUE              TRUE
  ID                        id-at-communicationsNetwork }

CommunicationsNetwork ::= OBJECT IDENTIFIER
```

This attribute describes the type of network where the Communications Address is allocated. For example, a Public Switched Telephone Network (PSTN), an ISDN network, or a GSM mobile phone network. It could also be an application oriented network, e.g., a banking network.

Allocation of object identifiers for the identification of networks is done outside this Directory Specification.

6.8 Preferences attribute types

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

6.8.1 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               PreferredDeliveryMethod
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               deliveryMethod.&id
  LDAP-NAME                 {"preferredDeliveryMethod"}
  ID                        id-at-preferredDeliveryMethod }

PreferredDeliveryMethod ::= 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) }
```

6.9 OSI application attribute types

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

6.9.1 Presentation Address

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

An attribute value for Presentation Address is a presentation-address as defined in Rec. ITU-T X.519 | ISO/IEC 9594-5.

```
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,
  ... }
```

6.9.2 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 }
```

6.9.3 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 }
```

6.10 Relational attribute types

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

6.10.1 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
  LDAP-SYNTAX               dn.&id
  LDAP-NAME                 {"distinguishedName"}
  ID                        id-at-distinguishedName }
```

6.10.2 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
  LDAP-SYNTAX               dn.&id
  LDAP-NAME                 {"member"}
  ID                        id-at-member }
```

6.10.3 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
  LDAP-SYNTAX               nameAndOptionalUID.&id
  LDAP-NAME                 {"uniqueMember"}
  ID                        id-at-uniqueMember }

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

6.10.4 Owner

The *Owner* attribute type specifies the name of an 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
  LDAP-SYNTAX               dn.&id
```

```
  LDAP-NAME                 {"owner"}
 ID                        id-at-owner }
```

6.10.5 Role Occupant

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

An attribute value for Role Occupant is a distinguished name.

```
roleOccupant ATTRIBUTE ::= {
  SUBTYPE OF                distinguishedName
  LDAP-SYNTAX               dn.&id
  LDAP-NAME                 {"roleOccupant"}
  ID                        id-at-roleOccupant }
```

6.10.6 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
  LDAP-SYNTAX               dn.&id
  LDAP-NAME                 {"seeAlso"}
  ID                        id-at-seeAlso }
```

6.11 Domain attribute types

6.11.1 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               UnboundedDirectoryString
  ID                        id-at-dmdName }
```

6.12 Hierarchical attribute types

Hierarchical attribute types are used for mapping the hierarchical structure of object identifiers and Uniform Resource Names (URNs) into a Directory Distinguished Name.

6.12.1 Top level object identifier arc

An attribute of the **oidC1** attribute type specifies the value for the top level arc of an object identifier. An attribute of this type shall take the value 0, 1 or 2. It is intended to be used as a naming attribute in an entry of **oidC1obj** object class and of **oidRoot** object class.

```
oidC1 ATTRIBUTE ::= {
  WITH SYNTAX               INTEGER
  EQUALITY MATCHING RULE    integerMatch
  SINGLE VALUE              TRUE
  ID                        id-oidC1 }
```

This attribute type has been moved from Rec. ITU-T X.660 | ISO/IEC 9834-1. The object identifier **id-oidC1** is allocated from the object identifier arc of Rec. ITU-T X.660 | ISO/IEC 9834-1.

6.12.2 Second level object identifier arc

An attribute of the **oidC2** attribute type specifies the value for the second level arc of an object identifier. An attribute of this type is intended to be used as a naming attribute in an entry of **oidC2obj** object class and of **oidRoot** object class.

```
oidC2 ATTRIBUTE ::= {
  WITH SYNTAX               INTEGER
  EQUALITY MATCHING RULE    integerMatch
  SINGLE VALUE              TRUE
```

```
  ID                        id-oidC2 }
```

This attribute type has been moved from Rec. ITU-T X.660 | ISO/IEC 9834-1. The object identifier **id-oidC2** is allocated from the object identifier arc of Rec. ITU-T X.660 | ISO/IEC 9834-1.

6.12.3 Lower level object identifier arcs attribute type

An attribute of the **oidC** attribute type specifies the value for a third level or lower level arcs of an object identifier. An attribute of this type is intended to be used as a naming attribute in an entry of **oidCobj** object class and of **oidRoot** object class.

```
oidC ATTRIBUTE ::= {
  WITH SYNTAX               INTEGER
  EQUALITY MATCHING RULE    integerMatch
  SINGLE VALUE              TRUE
  ID                        id-oidC }
```

This attribute type has been moved from Rec. ITU-T X.660 | ISO/IEC 9834-1. The object identifier **id-oidC** is allocated from the object identifier arc of Rec. ITU-T X.660 | ISO/IEC 9834-1.

6.12.4 URN component attribute type

An attribute of the **urnC** attribute type is used for holding a URN component when creating a DIT subtree representation of a URN. An attribute of this type is the naming attribute of an entry of the **urnCobj** structural object class.

```
urnC ATTRIBUTE ::= {
  WITH SYNTAX               PrintableString
  EQUALITY MATCHING RULE    caseExactMatch
  SINGLE VALUE              TRUE
  LDAP-SYNTAX                printableSting.&id
  LDAP-NAME                  {"urnC"}
  ID                        id-at-urnC }
```

The subtree root for a class of URNs shall have an attribute of this type which shall hold the URN name space component, as defined by the Internet Assigned Numbers Authority (IANA).

6.13 Attributes for applications using tag-based identification

Attribute types defined by this clause provide support for applications using tag-based identification.

6.13.1 Tag OID

An attribute of type **tagOid** is used for holding an object identifier. This object identifier indicates the type of UII following the object identifier in the ID tag (e.g., an RFID tag).

```
tagOid ATTRIBUTE ::= {
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               oid.&id
  LDAP-NAME                 {"tagOid"}
  ID                        id-at-tagOid }
```

6.13.2 UII Format

The *UII Format* attribute type specifies how a UII bit string may be partitioned into components.

```
uiiFormat ATTRIBUTE ::= {
  WITH SYNTAX               UiiFormat
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               uiiForm.&id
  LDAP-NAME                 {"uiiFormat"}
  ID                        id-at-uiiFormat }

UiiFormat ::= SEQUENCE {
  baseObject  URI,
  subset      ENUMERATE {
    baseObject    (0),
    oneLevel      (1),
    wholeSubtree (2) } DEFAULT baseObject,
  next        CHOICE {
```

```
    length      INTEGER,
    filter      UiiFilter } }

UiiFilter ::= CHOICE {
  item   [0]  UiiItem,
  and    [1]  SET OF UiiFilter,
  or     [2]  SET OF UiiFilter,
  not    [3]  UiiFilter }

UiiItem ::= SEQUENCE {
  type   ATTRIBUTE.&id OPTIONAL,
  length INTEGER       OPTIONAL }
```

The **baseObject** component shall contain the URN corresponding to the base object of the search. If this component is absent, the search is suggested to start at the root.

The **subset** component recommends how to fill the **subset** component of the **Filter** in a subsequent Search operation. **baseObject** specifies that the search shall only be performed against the base object. **oneLevel** specifies that only the entries immediately subordinate to the base object are to be searched. **wholeSubtree** specifies that all entries in the subtree which have the base object as a root are to be searched.

The **next** component gives some information about the next field of the UII:

– The choice **length** subcomponent shall be taken if the length of the following UII field has a fixed length and the subcomponent signals the length in characters.

– The choice **filter** shall be taken if the following UII field does not have a fixed length. It provides guidance as to how filter items should be constructed to explore the actual length of the next UII field. The **UiiFilter** data type has a recursive structure similar to the structure of the **Filter** data type as defined by Rec. ITU-T X.511 allowing for specification of a filter of arbitrary complexity.

  The recommendation for a particular filter item is given by the **UiiItem** data type:

  i) the **type** subcomponent specifies the attribute type to be used in the attribute value assertion; and

  ii) the **length** subcomponent specifies how many characters to be used as the value of the attribute value assertion.

6.13.3 UII in URN attribute type

An attribute of **uiiInUrn** type holds a Unique Item Identifier (UII) encoded in a unique URN format.

```
uiiInUrn ATTRIBUTE ::= {
  SUBTYPE OF                urn
  SINGLE VALUE              TRUE
  ID                        id-at-uiiInUrn }
```

6.13.4 Content URL

An attribute of **contentURL** is used for holding the URL of the information content associated with an EPC or a UII.

```
contentUrl ATTRIBUTE ::= {
  SUBTYPE OF                url
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"contentUrl"}
  ID                        id-at-contentUrl }
```

6.13.5 UII attribute type

The *UII* attribute type is used for holding a bit-encoded Unique Item Identifier (UII) allocated within the ISO environment.

```
uii ATTRIBUTE ::= {
  WITH SYNTAX               BIT STRING
  EQUALITY MATCHING RULE    bitStringMatch
  LDAP-SYNTAX               bitString.&id
  LDAP-NAME                 {"uii"}
  ID                        id-at-uii }
```

6.13.6 EPC attribute type

An attribute of the **epc** attribute type is used for holding a bit-encoded Electronic Product Code (EPC).

```
epc ATTRIBUTE ::= {
  WITH SYNTAX               BIT STRING
  SINGLE VALUE              TRUE
  EQUALITY MATCHING RULE    bitStringMatch
  LDAP-SYNTAX               bitString.&id
  LDAP-NAME                 {"epc"}
  ID                        id-at-epcUii }
```

6.13.7 Tag AFI attribute type

The *Tag AFI* attribute type is used for holding the Application Family Identifier (AFI) associated with a specific ISO UII type. AFIs are only allocated in the ISO environment. Together, an ISO UII and the associated AFI provide a global unique identification of an item.

```
tagAfi ATTRIBUTE ::= {
  WITH SYNTAX               OCTET STRING
  EQUALITY MATCHING RULE    octetStringMatch
  LDAP-SYNTAX               octetString.&id
  LDAP-NAME                 {"tagAfi"}
  ID                        id-at-isoTagAfi }
```

An AFI is typically one octet long, but provision is made for multi-octet AFIs.

6.13.8 EPC Format attribute

An attribute of the **epcFormat** attribute type specifies how an EPC bit string may be partitioned into components.

```
epcFormat  ATTRIBUTE  ::=  {
  WITH SYNTAX               EpcFormat
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               epcForm.&id
  LDAP-NAME                 {"epcFormat"}
  ID                        id-at-epcFormat }

EpcFormat ::= SEQUENCE {
  fields          SEQUENCE SIZE (1..MAX) OF SEQUENCE {
    bits            INTEGER,
    charField       CHOICE {
      characters  [0] INTEGER,
      maxValue    [1] INTEGER },
    result          ENUMERATED {
      numericPad      (0),
      numeric         (1),
      alpha7bits      (2) } DEFAULT numericPad,
  digitShift  [0] INTEGER                       OPTIONAL,
  checkCalc   [1] INTEGER                       OPTIONAL,
  urnPrefix       UTF8String                    OPTIONAL }
```

An attribute of the **uiiFormat** attribute type carries formatting information about the fields of a UII as retrieved from an RFID tag. It is intended to carry sufficient information to allow the RFID bit representation to be converted to a character representation. In addition, it allows conversion to a URN format, possibly to be used for a directory access.

Only the fields after the Header, the Filter field and the Partition field are considered when generating the character encoded UII or when creating a URN. It is assumed that the DUA/LDAP client has been able to identify these fields.

The attribute type syntax has the following components:

The **fields** component holds information about each of the EPC fields for which information is returned. For each field the following information is provided:

a) The **bits** subcomponent indicates how many bits the field occupies in the EPC.

b) The **charField** subcomponent is a choice of:

- The **characters** choice indicates how many characters to which the field shall be converted when decoding the EPC into a character representation. If it is a numeric character field and the result exceeds the indicated number of numeric characters, the EPC is invalid.
- The **maxValue** choice is only valid for a numeric field and indicates the maximum value allowed. If the value exceeds this value, the EPC is invalid.

c) The **result** subcomponent shall indicate how the bit field shall be converted and shall take one of the following values:

- **numericPad** meaning that the bit string of the field shall be considered an unsigned integer that shall be converted to a numeric string. When this value is chosen, the **characters** alternative of the **charField** subcomponent shall be taken. If the number of numeric characters are less than the value of the **characters** choice, then the result shall be prefixed '0' numeric characters to get the length as indicated by the **characters** choice.
- **numeric** meaning that the bit string of the field shall be considered an unsigned integer that shall be converted to a numeric string. There shall be no leading zero numeric characters. However, if the bit string are all zero bits, the result shall be a single zero numeric character.
- **alpha7bits** meaning that the bit string of the field consists of 7 bits subfields each representing an ASCII character to be converted to an 8 bits ASCII character.

Based on the above information, the bit encoded UII can be converted to a character encoded format where the characters can be considered as numbered from one to maximum from the left.

The **digitShift** component is only relevant if a particular digit (numeric character) shall be shifted as part of the procedure when character encoding an EPC. The value shall indicate the position of the numeric character in the converted EPC to be shifted. If this component is present, the digit in question shall be moved to the front of the converted EPC. The moved digit is now character number one. However, if the intention is to produce a URN, the indicator digit shall not be moved.

NOTE – indicator digit is used as a common nomination for an indicator digit and an extension digit.

The **checkDigit** component shall only be present if a check digit shall be generated for the character encoded EPC. The check digit shall not be generated when producing a URN. The value shall indicate how many of the initial characters that are used for generating the check digit. All these characters shall be numeric characters (digits). The check digit is generated by taking the sum of the digits after having multiplied all the uneven numbered digits with 3 and then subtract the sum from the nearest equal or higher multiple of ten. The check digit shall be inserted right after the last digit that was used for generating the check digit.

If the DUA or LDAP client elects to translate the EPC into a URN based on this information, the converted fields shall be concatenated with a full stop ('.') inserted between the fields.

To make the URN globally unique, the string in the **urnPrefix** component may be used to prefix the result.

6.13.9 EPC in URN attribute type

An attribute of type **epcInUrn** specifies a Unique Item Identifier (UII) encoded in a unique URN format.

```
epcInUrn ATTRIBUTE ::= {
  SUBTYPE OF                urn
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"epcInUrn"}
  ID                        id-at-epcInUrn }
```

6.13.10 LDAP URL attribute type

An attribute of type **ldapUrl** is used for holding the URL of an LDAP system.

```
ldapUrl ATTRIBUTE ::= {
  SUBTYPE OF                url
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"ldapUrl"}
  ID                        id-at-ldapUrl }
```

6.13.11 Tag location

An attribute of type **tagLocation** is used for holding the position of a tag as expressed in coordinates.

```
tagLocation ATTRIBUTE ::= {
  SUBTYPE OF                utmCoordinates
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               utmCoords.&id
  LDAP-NAME                 {"tagLocation"}
  ID                        id-at-tagLocation }
```

6.14 Notification attributes

Notification attributes have the syntax of attributes, but are defined to carry additional information in **CommonResults** (or **CommonResultsSeq**) and **PartialOutcomeQualifier** elements (as described in clauses 7.4 and 10.1 of Rec. ITU-T X.511 | ISO/IEC 9594-3). They are usually defined with matching rules so that returned values can be tested against locally known values.

6.14.1 DSA Problem

The *DSA Problem* notification attribute is used in conjunction with a **serviceError** or a **PartialOutcomeQualifier** and is defined as follows:

```
dSAProblem ATTRIBUTE ::= {
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  ID                        id-not-dSAProblem }
```

Values defined for **dsaProblem** are:

a) **id-pr-targetDsaUnavailable** – A request has to be chained to another DSA during name resolution, but no association can be established with this DSA.

b) **id-pr-dataSourceUnavailable** – A DSA cannot complete an operation as part of the DIB is not available.

c) **id-pr-administratorImposedLimit** – An operation has exceeded a limit set by the administrator.

d) **id-pr-permanentRestriction** – An operation has caused the DSA to exceed a limit that causes the process to stop and a repeated operation is judged to encounter the same problem.

e) **id-pr-temporaryRestriction** – An operation has caused the DSA to exceed a limit that causes the process to stop, but the reason is judged to be a temporary problem, e.g., resources depletion.

6.14.2 Search Service Problem

The *Search Service Problem* notification attribute describes problems in applying search-rule policies, and is used in conjunction with service-errors or **PartialOutcomeQualifier**. It is defined as follows:

```
searchServiceProblem ATTRIBUTE ::= {
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  SINGLE VALUE              TRUE
  ID                        id-not-searchServiceProblem }
```

Values defined for **searchServiceProblem** are:

a) **id-pr-unidentifiedOperation** – The attempted operation does not correspond to one of those identified for this service.

b) **id-pr-unavailableOperation** – The attempted operation only complies with a search-rule that is not available to the requester.

c) **id-pr-searchAttributeViolation** – One or more attribute types required to be in the filter were not present.

d) **id-pr-searchAttributeCombinationViolation** – The filter of the **search** request did include the required combination of attribute types.

e) **id-pr-searchValueNotAllowed** – Attribute values were specified for attribute types where only the attribute types can be specified in **present** and **contextPresent** filter item types.

f) **id-pr-missingSearchAttribute** – The identified attributes, which were not present in the requested search, are required for the relevant search-rule.

g) **id-pr-searchValueViolation** – The identified attribute values for the identified attribute types are not allowed when searching using the relevant search-rule.

h) **id-pr-attributeNegationViolation** – The identified attribute type is not allowed in negated form in the search filter.

i) **id-pr-searchValueRequired** – The identified attribute type is not allowed in filter items that do not require value matching.

j) **id-pr-invalidSearchValue** – The identified attribute values are not valid for the identified attribute types for the relevant search-rule.

k) **id-pr-searchContextViolation** – The identified context types in the attempted search are not allowed for the attribute type.

l) **id-pr-searchContextCombinationViolation** – The identified combinations of context types, which were not present in the requested search, are required for the relevant search-rule.

m) **id-pr-missingSearchContext** – The identified context types, which were not present in the requested search, are required for the attribute type.

n) **id-pr-searchContextValueViolation** – The identified context values for the identified context types are not allowed for the attribute type.

o) **id-pr-searchContexValueRequired** – The identified attribute type is not allowed in filter items that do not require value matching.

p) **id-pr-invalidContextSearchValue** – The identified attribute values are not valid for the identified attribute types for the relevant search-rule.

q) **id-pr-unsupportedMatchingRule** – The identified requested matching rule is not supported.

r) **id-pr-attributeMatchingViolation** – The identified requested matching rule, or its particular use, is not allowed for the identified attributes for the relevant search-rule.

s) **id-pr-unsupportedMatchingUse** – The way a matching rule is suggested to be used in a search filter is not supported.

t) **id-pr-matchingUseViolation** – The way a matching rule is suggested to be used in a search filter is not allowed, e.g., as specified in a search-rule.

u) **id-pr-hierarchySelectForbidden** – Hierarchy selection, except for **self**, is not allowed for the type of request.

v) **id-pr-invalidHierarchySelect** – One or more invalid hierarchy selection options were specified in the request.

w) **id-pr-unavailableHierarchySelect** – One or more hierarchy selections are not supported by the implementation.

x) **id-pr-invalidSearchControlOptions** – One or more invalid search options were specified in the request.

y) **id-pr-invalidServiceControlOptions** – One or more invalid service control options were specified in the request.

z) **id-pr-searchSubsetViolation** – The requested search subset is not allowed for the relevant search rule.

aa) **id-pr-unmatchedKeyAttributes** – A mapping-based matching rule was selected, but the mappable filter items did not provide any match against the relevant mapping table.

bb) **id-pr-ambiguousKeyAttributes** – A mapping-based matching rule was selected, but the mappable filter items provided multiple matches against the relevant mapping table.

cc) **id-pr-unavailableRelaxationLevel** – The DSA does not support a requested relaxation extension level.

dd) **id-pr-emptyHierarchySelection** – A hierarchy selection was specified that resulted in no entry returned although there were one or more entries that matched the search filter.

ee) **id-pr-relaxationNotSupported** – Relaxation was specified in the user request, but it is not supported.

6.14.3 Service-type

The *Service-type* notification attribute gives the service-type for the failing search.

```
serviceType ATTRIBUTE ::= {
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  SINGLE VALUE              TRUE
  ID                        id-not-serviceType }
```

6.14.4 Attribute Type List

The *Attribute Type List* notification attribute gives a list of attribute types to further qualify a search service problem.

```
attributeTypeList ATTRIBUTE ::= {
```

```
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  ID                        id-not-attributeTypeList }
```

6.14.5 Matching Rule List

The *Matching Rule List* notification attribute gives a list of matching rules to further qualify a search service problem.

```
matchingRuleList ATTRIBUTE ::= {
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  ID                        id-not-matchingRuleList }
```

6.14.6 Filter Item

The *Filter Item* notification attribute gives a list of invalid filter items in a search filter.

```
filterItem ATTRIBUTE ::= {
  WITH SYNTAX               FilterItem
  ID                        id-not-filterItem }
```

6.14.7 Attribute Combinations

The *Attribute Combinations* notification attribute gives a list of attribute combinations that were required to be presented in a filter, but were not provided.

```
attributeCombinations ATTRIBUTE ::= {
  WITH SYNTAX               AttributeCombination
  ID                        id-not-attributeCombinations }
```

6.14.8 Context Type List

The *Context Type List* notification attribute gives a list of context types to further qualify a search service problem.

```
contextTypeList ATTRIBUTE ::= {
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  ID                        id-not-contextTypeList }
```

6.14.9 Context List

The *Context List* notification attribute gives a list of contexts to further qualify a search service problem.

```
contextList ATTRIBUTE ::= {
  WITH SYNTAX               ContextAssertion
  ID                        id-not-contextList }
```

A value of this attribute type represents a context type and some context values of this type which are not allowed in the particular situation that resulted in this attribute being generated.

6.14.10 Context Combinations

The *Context Combinations* notification attribute gives a list of context combinations required to be presented in a filter, but were not provided.

```
contextCombinations ATTRIBUTE ::= {
  WITH SYNTAX               ContextCombination
  ID                        id-not-contextCombinations }
```

6.14.11 Hierarchy Select List

The *Hierarchy Select List* notification attribute gives a bitstring identifying one or more hierarchy selection options as defined by the **HierarchySelections** construct defined in clause 10.2.1 of Rec. ITU-T X.511 | ISO/IEC 9594-3.

```
hierarchySelectList ATTRIBUTE ::= {
  WITH SYNTAX               HierarchySelections
  SINGLE VALUE              TRUE
  ID                        id-not-hierarchySelectList }
```

When a bit is set in the **HierarchySelection** bitstring, it indicates that the corresponding hierarchy selection is invalid. Either a forbidden or unsupported selection has been requested, or the selection has not been requested when it is required.

6.14.12 Search Control Options List

The *Search Control Options List* notification attribute gives a bitstring identifying one or more search control options as defined by the **SearchControlOptions** ASN.1 data type in clause 10.2.1 of Rec. ITU-T X.511 | ISO/IEC 9594-3.

```
searchControlOptionsList ATTRIBUTE ::= {
  WITH SYNTAX               SearchControlOptions
  SINGLE VALUE              TRUE
  ID                        id-not-searchControlOptionsList }
```

When a bit is set in the **SearchControlOptions**, it indicates that the corresponding search control option selection is invalid. Either a forbidden or unsupported option has been requested, or the option has not been requested when it is required.

6.14.13 Service Control Options List

The *Service Control Options List* notification attribute gives a bitstring identifying one or more service control options as defined by the **ServiceControlOptions** ASN.1 data type defined in clause 7.5 of Rec. ITU-T X.511 | ISO/IEC 9594-3.

```
serviceControlOptionsList ATTRIBUTE ::= {
  WITH SYNTAX               ServiceControlOptions
  SINGLE VALUE              TRUE
  ID                        id-not-serviceControlOptionsList }
```

When a bit is set in the **ServiceControlOptions**, it indicates that the corresponding service control option selection is invalid. Either a forbidden or unsupported option has been requested, or the option has not been requested when it is required.

6.14.14 Multiple Matching Localities

The *Multiple Matching Localities* notification attribute specifies in each value a set of attribute assertions that if applied against the gazetteer will give a unique match.

```
multipleMatchingLocalities ATTRIBUTE ::= {
  WITH SYNTAX               MultipleMatchingLocalities
  ID                        id-not-multipleMatchingLocalities }

MultipleMatchingLocalities ::= SEQUENCE {
  matchingRuleUsed  MATCHING-RULE.&id OPTIONAL,
  attributeList     SEQUENCE OF AttributeValueAssertion,
  ... }
```

The **matchingRuleUsed** element is optionally present, and can be used to indicate the mapping-based matching rule that was used.

No matching rule is defined for this attribute; multiple identical or nearly identical values are tolerated.

6.14.15 Proposed Relaxation

The *Proposed Relaxation* notification attribute gives sequence-of **MRMapping** elements that can be supplied as part of the **RelaxationPolicy** supplied in the **relaxation** component of a subsequent **search** request.

```
proposedRelaxation ATTRIBUTE ::= {
  WITH SYNTAX               MRMappings
  ID                        id-not-proposedRelaxation }

MRMappings ::= SEQUENCE OF MRMapping
```

The sequence-of **MRMapping** has no significance.

6.14.16 Applied Relaxation

The *Applied Relaxation* notification attribute is used to list the attributes of the filter which have been subject to relaxation or tightening, other than those made by the **basic** element of a relaxation policy.

```
appliedRelaxation ATTRIBUTE ::= {
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  ID                        id-not-appliedRelaxation }
```

6.14.17 Password response

The Password Response notification attribute is used to give additional information in a password compare result.

```
pwdResponseValue ATTRIBUTE ::= {
  WITH SYNTAX               PwdResponse
  ID                        id-not-pwdResponse }

PwdResponse ::= SEQUENCE {
  warning CHOICE {
    timeleft        [0] INTEGER(0..MAX),
    graceRemaining  [1] INTEGER(0..MAX),
    ... } OPTIONAL,
  error ENUMERATED {
    passwordExpired  (0),
    changeAfterReset (1),
    ... } OPTIONAL}
```

6.14.18 LDAP diagnostic message

A value of type **ldapDiagnosticMsg** is used to carry the **diagnosticMessage** from an LDAP result.

```
ldapDiagnosticMsg ATTRIBUTE ::= {
  WITH SYNTAX               UTF8String
  SINGLE VALUE              TRUE
  ID                        id-not-ldapDiagnosticMsg }
```

6.15 LDAP defined attribute types

6.15.1 User ID attribute type

```
uid ATTRIBUTE ::= {
  WITH SYNTAX               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"uid"}
  ID                        id-coat-uid }
```

6.15.2 Domain Component attribute type

```
dc ATTRIBUTE ::= {
  WITH SYNTAX               IA5String
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               ia5String.&id
  LDAP-NAME                 {"dc"}
  ID                        id-coat-dc }
```

SECTION 3 – MATCHING RULES AND SYNTAXES

7 String preparation

The following six-step process shall be applied to each presented and attribute value in preparation for string match rule evaluation.

1) Transcode;
2) Map;
3) Normalize;
4) Prohibit;
5) Check bidi; and
6) Insignificant Character Removal.

Failure in any step shall cause the assertion to be UNDEFINED.

Comparison values created during the string preparation process are ephemeral, and shall not affect the attribute value stored in the Directory.

7.1 Transcode

Each non-Unicode string value is transcoded to Unicode.

`PrintableString` values are transcoded directly to Unicode.

`UniversalString`, **`UTF8String`**, and **`BMPString`** values do not need to be transcoded as they are Unicode-based strings (in the case of **`BMPString`**, restricted to a subset of Unicode).

If the implementation is unable or unwilling to perform the transcoding as described above, or the transcoding fails, this step fails and the assertion is evaluated to UNDEFINED.

The transcoded string is the output string.

7.2 Map

SOFT HYPHEN (U+00AD) and MONGOLIAN TODO SOFT HYPHEN (U+1806) code points are mapped to nothing. COMBINING GRAPHEME JOINER (U+034F) and VARIATION SELECTORs (U+180B-180D,FF00-FE0F) code points are also mapped to nothing. The OBJECT REPLACEMENT CHARACTER (U+FFFC) is mapped to nothing.

CHARACTER TABULATION (U+0009), LINE FEED (LF) (U+000A), LINE TABULATION (U+000B), FORM FEED (FF) (U+000C), CARRIAGE RETURN (CR) (U+000D), and NEXT LINE (NEL) (U+0085) are mapped to SPACE (U+0020).

All other control code points (e.g., Cc) or code points with a control function (e.g., Cf) are mapped to nothing.

ZERO WIDTH SPACE (U+200B) is mapped to nothing. All other code points with Separator (space, line, or paragraph) property (e.g., Zs, Zl, or Zp) are mapped to SPACE (U+0020).

For case ignore, numeric, and stored prefix string matching rules, characters are case-folded as per B.2 of IETF RFC 3454.

7.3 Normalize

The input string is normalized to Unicode Form KC (compatibility composed) as described in Unicode Standard Annex #15.

7.4 Prohibit

All Unassigned, Private Use, and non-character code points are prohibited. Surrogate codes (U+D800-DFFFF) are prohibited.

The REPLACEMENT CHARACTER (U+FFFD) code is prohibited. The first code point of a string is prohibited from being a combining character. Empty strings are prohibited. The step fails and the assertion is evaluated to UNDEFINED if the input string contains any prohibited code point. The output string is the input string.

7.5 Check bidi

There are no bidirectional restrictions. The output string is the input string.

7.6 Insignificant Character Removal

In this step, characters insignificant to the matching rule are to be removed. The characters to be removed differ from matching rule to matching rule. Clause 6.6.1 applies to case ignore and exact string matching.

7.6.1 Insignificant Space Removal

For the purposes of this clause, a space is defined to be the SPACE (U+0020) code point followed by no combining marks.

NOTE – The previous steps ensure that the string cannot contain any code points in the separator class, other than SPACE (U+0020).

The following spaces are regarded as not significant and shall be removed:

- 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.) For example, the removal of spaces from the Form KC string: "<SPACE><SPACE>foo<SPACE><SPACE>bar<SPACE><SPACE>" would result in the output string: "foo<SPACE>bar", and the Form KC string: "<SPACE><SPACE><SPACE>" would result in the output string: "<SPACE>".

7.6.2 NumericString Insignificant Character Removal

For the purposes of this clause, a space is defined to be the SPACE (U+0020) code point followed by no combining marks. All spaces are regarded as not significant and are to be removed. For example, the removal of spaces from the Form KC string: "<SPACE><SPACE>123<SPACE><SPACE>456<SPACE><SPACE>" would result in the output string: "123456", and the Form KC string: "<SPACE><SPACE><SPACE>" would result in an empty output string.

8 Definition of matching rules

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

8.1 String matching rules

In the matching rules specified in clauses 8.1.1 to 8.1.9, all presented and stored string values are to be prepared for matching as described in clause 7. String preparation produces strings suitable for character-by-character matching.

8.1.1 Case Exact Match and Case Ignore Match

The *Case Exact Match* rule compares for equality a presented string with an attribute value of type **UnboundedDirectoryString** or **DirectoryString** or one of the data types appearing in the choice type **UnboundedDirectoryString** or (equivalently) **DirectoryString**, e.g., **UTF8String** without regard to insignificant spaces (see clause 7.6).

```
caseExactMatch MATCHING-RULE ::= {
  SYNTAX        UnboundedDirectoryString
  LDAP-SYNTAX   directoryString.&id
  LDAP-NAME     {"caseExactMatch"}
  ID            id-mr-caseExactMatch }
```

The *Case Ignore Match* rule compares for equality a presented string with an attribute value of type **UnboundedDirectoryString** or one of the data types appearing in the choice type **DirectoryString**, e.g., **UTF8String**, without regard to the case (upper or lower) of the strings (e.g., "Dundee" and "DUNDEE" match) and insignificant spaces (see clause 7.6). The rule is identical to the **caseExactMatch** rule except upper-case characters are folded to lower case during string preparation as discussed in clause 7.2. After taking white space into account, caseless matching shall be performed by performing case folding as described in The Unicode Standard and applying

Normalization Form D or Form KC as described in Unicode Technical Report 15, depending on the character repertoire commonly examined and performance requirements.

```
caseIgnoreMatch MATCHING-RULE ::= {
  SYNTAX        UnboundedDirectoryString
  LDAP-SYNTAX   directoryString.&id
  LDAP-NAME     {"caseIgnoreMatch"}
  ID            id-mr-caseIgnoreMatch }
```

Both rules return TRUE if the prepared strings are the same length and corresponding characters in the prepared strings are identical.

8.1.2 Case Exact Ordering Match and Case Ignore Ordering Match

The *Case Exact Ordering Match* rule compares the collation order of a presented string with an attribute value of type **UnboundedDirectoryString** or one of the data types appearing in the choice type **DirectoryString**, e.g., **UTF8String** without regard to insignificant spaces (see clause 7.6).

```
caseExactOrderingMatch MATCHING-RULE ::= {
  SYNTAX        UnboundedDirectoryString
  LDAP-SYNTAX   directoryString.&id
  LDAP-NAME     {"caseExactOrderingMatch"}
  ID            id-mr-caseExactOrderingMatch }
```

The *Case Ignore Ordering Match* rule compares the collation order of a presented string with an attribute value of type **UnboundedDirectoryString** or one of the data types appearing in the choice type **UnboundedDirectoryString**, e.g., **UTF8String**, without regard to the case (upper or lower) of the strings and insignificant spaces (see clause 7.6). The rule is identical to the **caseExactOrderingMatch** rule except upper-case characters are folded to lower case during string preparation as discussed in clause 7.2.

```
caseIgnoreOrderingMatch MATCHING-RULE ::= {
  SYNTAX        UnboundedDirectoryString
  LDAP-SYNTAX   directoryString.&id
  LDAP-NAME     {"caseIgnoreOrderingMatch"}
  ID            id-mr-caseIgnoreOrderingMatch }
```

Both rules return TRUE if the attribute value is "less" or appears earlier than the presented value, when the strings are compared using the Unicode code point collation order.

NOTE – Collation order provides language and culture-specific information about how the characters of a given language are sorted. A Directory system can support several configurable collation orders. Implementation of this capability is outside the scope of this Directory Specification.

8.1.3 Case Exact Substrings Match and Case Ignore Substrings Match

The *Case Exact Substrings Match* rule determines whether a presented value is a substring of an attribute value of type **DirectoryString** or one of the data types appearing in the choice type **UnboundedDirectoryString**, e.g., **UTF8String** without regard to insignificant spaces (see clause 7.6).

```
caseExactSubstringsMatch MATCHING-RULE ::= {
  SYNTAX        SubstringAssertion -- only the PrintableString choice
  LDAP-SYNTAX   substringAssertion.&id
  LDAP-NAME     {"caseExactSubstringsMatch"}
  ID            id-mr-caseExactSubstringsMatch }
```

The *Case Ignore Substrings Match* rule determines whether a presented value is a substring of an attribute value of type **UnboundedDirectoryString** or one of the data types appearing in the choice type **UnboundedDirectoryString**, e.g., **UTF8String**, without regard to the case (upper or lower) of the strings and insignificant spaces (see clause 7.6). The rule is identical to the **caseExactSubstringsMatch** rule except upper-case characters are folded to lower case during string preparation as discussed in clause 7.2.

```
caseIgnoreSubstringsMatch MATCHING-RULE ::= {
  SYNTAX        SubstringAssertion
  LDAP-SYNTAX   substringAssertion.&id
  LDAP-NAME     {"caseIgnoreSubstringsMatch"}
  ID            id-mr-caseIgnoreSubstringsMatch }

caseIgnoreSubstringsMatch MATCHING-RULE ::= {
  SYNTAX        SubstringAssertion
  LDAP-SYNTAX   substringAssertion.&id
  LDAP-NAME     {"caseIgnoreSubstringsMatch"}
```

```
  ID            id-mr-caseIgnoreSubstringsMatch }

SubstringAssertion ::= SEQUENCE OF CHOICE {
  initial  [0]  UnboundedDirectoryString,
  any      [1]  UnboundedDirectoryString,
  final    [2]  UnboundedDirectoryString,
    -- at most one initial and one final component
  control       Attribute{{SupportedAttributes}},
    -- Used to specify interpretation of the following items
  ... }
```

Both rules return 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;
- **control** is not used for the **caseIgnoreSubstringsMatch**, **telephoneNumberSubstringsMatch**, or any other form of substring match for which only initial, any, or final elements are used in the matching algorithm; if a **control** element is encountered, it is ignored. The control element is only used for matching rules that explicitly specify its use in the matching algorithm. Such a matching rule may also redefine the semantics of the **initial**, **any** and **final** substrings.

NOTE – The **generalWordMatch** matching rule is an example of such a matching rule.

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

For a component of substrings to match a portion of the attribute value, corresponding characters must be identical (including all combining characters in the combining character sequences).

8.1.4 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
  LDAP-SYNTAX   numericString.&id
  LDAP-NAME     {"numericStringMatch"}
  ID            id-mr-numericStringMatch }
```

The rule is identical to the **caseIgnoreMatch** rule (case is irrelevant as characters are numeric) except that all space characters are removed during preparation as detailed in clause 7.6.2.

8.1.5 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
  LDAP-SYNTAX   numericString.&id
  LDAP-NAME     {"numericStringOrderingMatch"}
  ID            id-mr-numericStringOrderingMatch }
```

The rule is identical to the **caseIgnoreOrderingMatch** rule (case is irrelevant as characters are numeric) except that all space characters are removed during string preparation as detailed in clause 7.6.2.

8.1.6 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**.

```
numericStringOrderingMatch MATCHING-RULE ::= {
  SYNTAX        NumericString
  LDAP-SYNTAX   numericString.&id
  LDAP-NAME     {"numericStringOrderingMatch"}
  ID            id-mr-numericStringOrderingMatch }
```

The rule is identical to the **caseIgnoreSubstringsMatch** rule (case is irrelevant as characters are numeric) except that all space characters are removed during string preparation as detailed in clause 7.6.2.

8.1.7 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 **UnboundedDirectoryString**, without regard to the case (upper or lower) of the strings and significant spaces (see clause 7.6).

```
caseIgnoreListMatch MATCHING-RULE ::= {
  SYNTAX       CaseIgnoreList
  LDAP-SYNTAX  postalAddr.&id
  LDAP-NAME    {"caseIgnoreListMatch"}
  ID           id-mr-caseIgnoreListMatch }

CaseIgnoreList ::= SEQUENCE OF UnboundedDirectoryString
```

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

8.1.8 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 **UnboundedDirectoryString**, but without regard for the case (upper or lower) of the strings and insignificant spaces (see clause 7.6).

```
caseIgnoreListSubstringsMatch MATCHING-RULE ::= {
  SYNTAX       SubstringAssertion
  LDAP-SYNTAX  substringAssertion.&id
  LDAP-NAME    {"caseIgnoreListSubstringsMatch"}
  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.

8.1.9 Stored Prefix Match

The *Stored Prefix Match* rule determines whether an attribute value, whose syntax is **UnboundedDirectoryString**, is a prefix (i.e., initial substring) of the presented value, without regard to the case (upper or lower) of the strings and insignificant spaces (see clause 7.6).

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

```
storedPrefixMatch MATCHING-RULE ::= {
  SYNTAX       UnboundedDirectoryString
  ID           id-mr-storedPrefixMatch }
```

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

8.2 Syntax-based matching rules

8.2.1 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
  LDAP-SYNTAX  bitString.&id
  LDAP-NAME    {"booleanMatch"}
  ID           id-mr-booleanMatch }
```

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

8.2.2 Integer Match

The *Integer Match* rule compares for equality a presented integer value or enumerated value with an attribute value of type **INTEGER** or **ENUMERATED**, respectively.

```
integerMatch MATCHING-RULE ::= {
  SYNTAX        INTEGER
  LDAP-SYNTAX   integer.&id
  LDAP-NAME     {"integerMatch"}
  ID            id-mr-integerMatch }
```

The rule returns TRUE if the presented integer value or the presented enumerated value is equal to the attribute value.

8.2.3 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
  LDAP-SYNTAX   integer.&id
  LDAP-NAME     {"integerOrderingMatch"}
  ID            id-mr-integerOrderingMatch }
```

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

8.2.4 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
  LDAP-SYNTAX   bitString.&id
  LDAP-NAME     {"bitStringMatch"}
  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. If the attribute syntax is defined with a "NamedBitList", the trailing zero bits in the attribute value and presented value are ignored.

8.2.5 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
  LDAP-SYNTAX   octetString.&id
  LDAP-NAME     {"octetStringMatch"}
  ID            id-mr-octetStringMatch }
```

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

8.2.6 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
  LDAP-SYNTAX   octetString.&id
  LDAP-NAME     {"octetStringOrderingMatch"}
  ID            id-mr-octetStringOrderingMatch }
```

The rule compares octet strings from the first octet to the 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.

8.2.7 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**.

8.2.8 Telephone Number Match

The *Telephone Number Match* rule compares for equality a presented value with an attribute value of type **TelephoneNumber** (see clause 6.7.1).

```
telephoneNumberMatch MATCHING-RULE ::= {
  SYNTAX        TelephoneNumber
  LDAP-SYNTAX   telephoneNr.&id
  LDAP-NAME     {"telephoneNumberMatch"}
  ID            id-mr-telephoneNumberMatch }
```

The rules for matching are identical to those for **caseIgnoreMatch**, except that all hyphens and spaces are insignificant and removed during the insignificant character removal step.

8.2.9 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
  LDAP-SYNTAX   substringAssertion.&id
  LDAP-NAME     {"telephoneNumberSubstringsMatch"}
  ID            id-mr-telephoneNumberSubstringsMatch }
```

The rules for matching are identical to those for **caseExactSubstringsMatch**, except that all hyphens and spaces are insignificant and removed during the insignificant character removal step.

8.2.10 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.

8.2.11 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
  LDAP-SYNTAX   nameAndOptionalUID.&id
  LDAP-NAME     {"uniqueMemberMatch"}
  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.

8.2.12 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.

8.2.13 Facsimile Number Match

The *Facsimile Number Match* rule compares for equality a presented value with the first element of the attribute value sequence. That element, **telephoneNumber**, is of type **TelephoneNumber** (see clause 6.7.1). The **parameters** element of the facsimile number sequence is not evaluated.

```
facsimileNumberMatch MATCHING-RULE ::= {
  SYNTAX        TelephoneNumber
  ID            id-mr-facsimileNumberMatch }
```

The rules for matching are identical to those for **telephoneNumberMatch**.

8.2.14 Facsimile Number Substrings Match

The *Facsimile Number Substrings Match* rule determines if a presented substring is a substring of the first element of the attribute value sequence. That element, **telephoneNumber,** is of type **TelephoneNumber** and is a telephone number. The **parameters** element of the facsimile number sequence is not evaluated.

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

The rules for matching are identical to those for **telephoneNumberMatch**.

8.2.15 UUID Pair Match

The *UUID Pair Match* rule compares presented values of **UUIDPair** for equality, and is defined as follows:

```
uUIDPairMatch MATCHING-RULE ::= {
  SYNTAX        UUIDPair
  ID            id-mr-uuidpairmatch }
```

A presented value of the type **UUIDPair** matches a target value of the type **UUIDPair** if, and only if each component of the first is equal to the corresponding component of the second, the corresponding components are of the same length, and the corresponding octets are equal.

8.2.16 Component Match

The syntaxes of attributes in a Directory system range from simple data types, such as text string, integer or Boolean, to complex structured data types, such as the syntaxes of the directory schema operational attributes. Matching rules defined for the complex syntaxes usually only provide the most immediately useful matching capability. IETF RFC 3687 specifies generic matching rules that can match any user-selected component parts in an attribute value of any arbitrarily complex attribute syntax. IETF RFC 3727 specifies an ASN.1 module useful for reference by other specifications. This matching rule specification is imported into **SelectedAttributeTypes** within this Directory Specification, and may be selected for use by means of the **extensibleMatch** component of **FilterItem**, as specified in Rec. ITU-T X.511 | ISO/IEC 9594-3.

8.3 Time matching rules

8.3.1 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.

8.3.2 UTC Time Ordering Match

The *UTC Time Ordering Match* 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 to 49 inclusive, the value shall have 2000 added to it; and

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

8.3.3 Generalized Time Match

The *Generalized Time Match* rule compares for equality a presented value with an attribute value of type **GeneralizedTime** (as per 46.3 b) or c) of Rec. ITU-T X.680 | ISO/IEC 8824-1).

```
generalizedTimeMatch MATCHING-RULE ::= {
  SYNTAX        GeneralizedTime
  -- as per 46.3 b) or c) of Rec. ITU-T X.680 | ISO/IEC 8824-1
  LDAP-SYNTAX   generalizedTime.&id
  LDAP-NAME     {"generalizedTimeMatch"}
  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.

8.3.4 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 46.3 b) and c) of Rec. ITU-T X.680 | ISO/IEC 8824-1).

```
generalizedTimeOrderingMatch MATCHING-RULE ::= {
  SYNTAX        GeneralizedTime
  -- as per 46.3 b) or c) of Rec. ITU-T X.680 | ISO/IEC 8824-1
  LDAP-SYNTAX   generalizedTime.&id
  LDAP-NAME     {"generalizedTimeOrderingMatch"}
  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.

8.3.5 System Proposed Match

The *System Proposed Match* rule is a dummy matching rule, defined as follows:

```
systemProposedMatch MATCHING-RULE ::= {
  ID  id-mr-systemProposedMatch }
```

This matching rule can by a requester to be included in the **RelaxationPolicy** within a **search** request to indicate that the Directory should determine which matching rule should be used in a matching rule substitution.

8.4 First component matching rules

8.4.1 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
  LDAP-SYNTAX   integer.&id
  LDAP-NAME     {"integerFirstComponentMatch"}
  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**.

8.4.2 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
  LDAP-SYNTAX   oid.&id
  LDAP-NAME     {"objectIdentifierFirstComponentMatch"}
  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**.

8.4.3 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        UnboundedDirectoryString
  LDAP-SYNTAX   directoryString.&id
  LDAP-NAME     {"directoryStringFirstComponentMatch"}
  ID            id-mr-directoryStringFirstComponentMatch }
```

The rule returns TRUE if the attribute value has a first component whose value matches the presented **UnboundedDirectoryString** 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**.

8.5 Word matching rules

8.5.1 Word Match

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

```
wordMatch MATCHING-RULE ::= {
  SYNTAX        UnboundedDirectoryString
  LDAP-SYNTAX   directoryString.&id
  LDAP-NAME     {"wordMatch"}
  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.

8.5.2 Keyword Match

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

```
keywordMatch MATCHING-RULE ::= {
  SYNTAX        UnboundedDirectoryString
  LDAP-SYNTAX   directoryString.&id
  LDAP-NAME     {"keywordMatch"}
  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.

8.5.3 General Word Match

The *General Word Match* rule compares words in a presented string with words in an attribute value of type **UnboundedDirectoryString**. The matching rule can also be used for attribute values of a type that explicitly specifies one of the **UnboundedDirectoryString** choices as its syntax.

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

This matching rule is differentiated from a normal substring matching rule by the interposition of control attributes before or between the **initial**, **any**, or **final** elements. If there are no control attributes in the filter item, the matching shall be performed as for the **caseExactSubstringsMatch** matching rule with the semantics of **initial**, **any** and **final** elements as defined by that matching rule. However, if the equality matching rule (if any) for the attribute type subject to the matching is **caseIgnoreMatch**, then the **caseIgnoreSubstringsMatch** shall be used instead.

Four types of control attribute are defined for general word match (restrictions on their placement are defined below); any other control attributes shall be ignored:

```
sequenceMatchType ATTRIBUTE ::= {
  WITH SYNTAX   SequenceMatchType
  SINGLE VALUE  TRUE
  ID            id-cat-sequenceMatchType } -- defaulting to sequenceExact

SequenceMatchType ::= ENUMERATED {
  sequenceExact                  (0),
  sequenceDeletion               (1),
  sequenceRestrictedDeletion     (2),
  sequencePermutation            (3),
  sequencePermutationAndDeletion (4),
  sequenceProviderDefined        (5),
  ... }

wordMatchTypes ATTRIBUTE ::= {
  WITH SYNTAX   WordMatchTypes
  SINGLE VALUE  TRUE
  ID            id-cat-wordMatchType } -- defaulting to wordExact

WordMatchTypes ::= ENUMERATED {
  wordExact           (0),
  wordTruncated       (1),
  wordPhonetic        (2),
  wordProviderDefined (3),
  ... }

characterMatchTypes ATTRIBUTE ::= {
  WITH SYNTAX   CharacterMatchTypes
  SINGLE VALUE  TRUE
  ID            id-cat-characterMatchTypes }

CharacterMatchTypes ::= ENUMERATED {
  characterExact      (0),
  characterCaseIgnore (1),
  characterMapped     (2),
  ... }

selectedContexts ATTRIBUTE ::= {
  WITH SYNTAX  ContextAssertion
  ID           id-cat-selectedContexts }
```

Each attribute affects all following **initial**, **any**, or **final** elements, and the values that it provides supersede those that were previously applicable.

Prior to the first **sequenceMatchType** attribute, if any, the value that is to be taken as applicable for the **sequenceMatchType** attribute shall be taken as **sequenceExact**. The attribute does not affect the evaluation of the **initial** and **final** elements, which shall always be taken as matching the initial and final words; it only affects the remaining unmatched words. The **initial** word, if present, shall match the first word of the stored text; if both are noise words, the two words shall be taken as matching. The positioning of **sequenceMatchType** attributes defines the words to which the form of match applies.

NOTE 1 – For many practical purposes it will be suffice to place the **sequenceMatchType** before the first **initial** element; particular implementations may not support the full generality of the definition.

Prior to the first **wordMatchType** attribute, if any, the value that is to be taken as applicable for the **wordMatchType** attribute shall be taken as **wordExact**. Prior to the first **characterMatchType** attribute, if any, the value that is to be taken as applicable for the **characterMatchType** attribute shall be taken as **characterExact**. However, if the equality matching rule (if any) for the attribute type subject to the matching is **caseIgnoreMatch**, then it shall instead be taken as **characterCaseIgnore**.

If **selectedContexts** control attribute is present, it shall be the first element; there shall only be one such control attribute; it shall be taken as a restriction on the stored value (see below).

The rule returns TRUE if the presented value contains a non-empty sequence of words which matches the specified initial and final words, and in addition the sequence of remaining unmatched words in the attribute value according to the specified **sequenceMatchType**, where corresponding words are matched according to the specified **wordMatchTypes** and corresponding characters within words are matched according to the specified **characterMatchTypes**, except that if the **selectedContexts** component is present in the presented value, all **ContextAssertion** elements are also required to evaluate to TRUE (as specified in Rec. ITU-T X.501 | ISO/IEC 9594-2). The rule returns FALSE for a given stored attribute when the words do not match, or when some **ContextAssertion** element does not match.

A word is a non-empty sequence of non-space characters bounded by the start or end of the string or by space or punctuation characters. Punctuation characters are defined as those that do not affect the semantics of word tokens, and normally include commas, quotes, full stops at the end of sentences, parentheses, etc. The determination of what characters are punctuation characters shall be a local matter.

NOTE 2 – For example, the character '!' is sometimes used in text to denote a clicking sound, as used in certain African languages, and is thus sometimes part of a word rather than an exclamation-mark (which would be a punctuation character).

Similarly, the **final** word, if present, shall match the last word of the stored text; if both are noise words, the two words shall be taken as matching.

Noise words, which are words which match one of the words on an implementation-defined list of semantically weak words (e.g., articles and prepositions) according to the specified **characterMatchTypes** are discarded from the sequence of words prior to matching, except to match **initial** and **final** words, and the corresponding rule in **wordMatchTypes** is discarded from the sequence of rules provided it is not the last such rule.

The sequence of words in the presented value matches the sequence of words in the attribute value if the latter can be transformed according to the specified **sequenceMatchType** into a sequence containing the same number of words as the first sequence and whose corresponding words match. If **sequenceMatchType** is **sequenceExact**, the transform leaves the sequence unchanged. If it is **sequenceDeletion**, it deletes zero or more words from the sequence. If it is **sequenceRestrictedDeletion**, it deletes zero or more words but not the first word from the sequence. If it is **sequencePermutation**, it permutes zero or more words in the sequence. If it is **sequencePermutationAndDeletion**, it deletes zero or more words in the sequence and permutes zero or more of the remaining words. If it is **sequenceProviderDefined**, it deletes, permutes, or inserts words in accordance with an implementation-defined rule.

A word in the presented value matches a word in the attribute value if the latter word can be transformed according to the corresponding rule from the specified **wordMatchTypes** into a sequence of characters which match in turn the characters of the word in the presented value. Each word is matched using the corresponding rule in **wordMatchTypes** where the correspondence is determined prior to applying any deletions or permutations from sequence matching; any words in excess of the number of rules in **wordMatchTypes** is matched using the last rule. If the rule is exact, the transform leaves the word unchanged. If it is **wordTruncated**, then zero or more characters are removed from the end of the word, up to an implementation-defined minimum word length. If it is **wordPhonetic**, the word is replaced with a word that matches it according to an implementation-defined phonetic matching algorithm. If it is **wordProviderDefined**, the word is matched in accordance with an implementation-defined rule.

The characters in each word are compared using the corresponding rule in **characterMatchTypes** where the correspondence is determined prior to applying any deletions or permutations from sequence matching; the characters of any words in excess of the number of rules in **characterMatchTypes** are matched using the last rule. If **characterMatchTypes** is **characterExact**, then the corresponding characters within the words match if they are the same. If it is **characterCaseIgnore,** then the corresponding characters within the words match if they are the same when differences in case are ignored. If it is **characterMapped**, the characters match if they map to the same character according to an implementation-defined mapping table. This table shall be such as to allow national characters listed in Figure A.2/T.51 of Rec. ITU-T T.51 to be matched using only the characters A-Z and 0-9 in presented values, and may map short sequences of characters onto a single character, e.g., ae to a-e-diphthong or ue to u-umlaut.

8.6 Approximate Matching Rules

8.6.1 Approximate String Match

The *Approximate String Match* rule compares a presented value with an attribute value according to a locally-defined approximate matching algorithm (e.g., spelling variations, phonetic match, etc.). The algorithm shall be the same as that invoked in response to processing a filter item of type **approximateMatch** (see Rec. ITU-T X.511 | ISO/IEC 9594-3).

```
approximateStringMatch MATCHING-RULE ::= {
  ID      id-mr-approximateStringMatch }
```

The assertion syntax for this matching rule is the same as the assertion syntax of the equality matching rule for the attribute to which it is applied. If no equality matching rule is defined for the attribute, any assertion syntax is permitted but the rule always evaluates to undefined.

8.7 Special Matching Rules

8.7.1 Ignore if Absent Match

The *Ignore if Absent Match* rule compares a value for any purpose and for any attribute.

```
ignoreIfAbsentMatch MATCHING-RULE ::= {
  ID      id-mr-ignoreIfAbsentMatch }
```

The rule returns as follows:

a) If the attribute is absent, the rule returns the value TRUE;

b) If the attribute is present, the rule returns the value undefined.

This match can only be used as a parent matching-rule. It is then used in conjunction with a matching rule which matches values when the attribute is present. See also clause 13.5.2 of Rec. ITU-T X.501 | ISO/IEC 9594-2.

NOTE – Within a service-specific administrative area, the same effect can be achieved by specifying an empty **defaultValues** subcomponent of the appropriate request-attribute-profile.

8.7.2 Null Match

The *Null Match* rule compares a value for any purpose and for any attribute, with the special rule:

```
nullMatch MATCHING-RULE ::= {
  ID      id-mr-nullMatch }
```

The rule returns as follows:

a) if the filter-item is non-negated, the rule returns the value TRUE; and

b) if the filter-item is negated, the rule returns the value FALSE.

This match can be used formally to cause a filter-item to be ignored. A filter item using null match shall be considered absent when evaluating compatibility with search-rules.

8.8 Zonal Match

A *Zonal Match* is primarily applicable to **search** requests that make use of geographical related mappable filter items. Such filter items could be assertions for **localityName**, **stateOrProvinceName**, **postalCode**, etc.

Zonal matching uses combinable filter items for the matching against the mapping table.

The zonal matching can take into account that users' perception of localities may be different from the locality model used within a DMD. The mapping between the users' perception and the model used within a DMD should take into account that a user may use localities that are not directly reflected in Directory entries or their names. Such localities may be fuzzy in the sense that they do not relate exactly to localities that are more official. Also, a user may guess slightly wrong on locality names when making a search if the object being looked for lives close to the border of a neighbouring locality. For this purpose, a region, e.g., a country, is divided into *zones*. Zones are areas that are completely contained within any locality referenced in a **search** request. The result of a mapping of the mappable filter items is a list of zones. For further explanation of zonal matching, see Annex E.

When using zonal match, the mapping table is called a *gazetteer* (i.e., a geographical dictionary). Within the filter, a set of combinable locality filter items may be able together to define a single *named place* (that is, a unique, usually contiguous local area), or, when this is permitted, a small number of named places that match the filter items. A named place is a distinct named real-world place, such as a town, village, county, etc.

A gazetteer will in general cover (i.e., provide a geographical database relating to) a domain comprising a single country or region. A geographical search inquiry shall be interpreted in terms of a specific gazetteer. How the scope of a search is determined, and an appropriate gazetteer selected, is a local matter, but the selection can be done by using a default gazetteer for the DSA, or be based on one or more attributes, e.g., **countryName**, **stateOrProvinceName** or **localityName** associated with the search operation (e.g., present as part of the distinguished name of the **baseObject**, or as part of the filter).

The first step of a zonal match is to use one or more filter items together to identify one or more named places. For this purpose, combinable locality filter items (i.e., all locality filter items within a single subfilter) are used together.

Otherwise, the procedure so far identifies one or more named places. At this stage, no reference at all has been made to information within the DIT. The remainder of the filter can then be used to identify all of the entries within the search scope that have positions corresponding to those named places, as defined later. Relaxation may be applicable so that named places will match more entry positions if inadequate results would be returned otherwise.

Zonal matching does not support tightening.

Each entry that is to be considered eligible for matching shall have a position that is identified either by a uniquely named place, perhaps using more than one place-name value, e.g., ("Newton" "Chester" "Cheshire"), or by one or more *zones* (see next paragraph), represented by values placed in a zone attribute. If an entry has zones to define its position, it may also have locality values, but the latter, in this case, are informational. The administrative authority is responsible for ensuring that locality information does indeed identify a named place.

Zones are primitive non-overlapping geographical components, distinct in kind from places, such that a place is precisely composed of one or more zones, as listed within the gazetteer. Zones are identified by string values that are unique within a gazetteer's region. Thus, two overlapping places would share one or more zones that correspond to the overlapping area. Zones are represented within entries as attributes, possibly as operational attributes. In this case, zonal information would never be returned as attribute values unless the attribute representing the zone is specifically requested as an operational attribute. Alternatively, a zone could be a standard attribute (e.g., **postalCode**). Locality values are returned as usual, subject to access control.

NOTE 1 – The exact nature of a zone, and its mapping to a specific attribute, is a local matter, and would probably depend on the capabilities of a specific implementation. In the United Kingdom, a good candidate for a zone would be a postal code, like "RG12 2JL", which often defines a small area such as one side of a street. Zones in city areas would then be small; those in country areas would be correspondingly large. In unpopulated and featureless areas (e.g., deserts), a zone could be very large indeed.

An entry's position (defined by zones) matches a named place, as defined by the gazetteer, if there is overlap between the zones defined for the named place and the zones defined for the entry (i.e., an overlap-based matching rule is used). If the entry's position is defined as a named place, the position is considered to be composed of the zones constituting the named place.

Zonal matching permits extended (i.e., relaxed) matching, where level 0 corresponds to the basic definition of objects in the gazetteer. Levels 1 and greater levels correspond to a gradual and systematic enlargement of the zones comprising a place so that more entry locations match.

The following is a more formal statement of the model underlying zonal match:

a) Zonal matching is based on the existence of one or more *gazetteers* that are supported for the purpose by DSAs. A gazetteer is a geographical dictionary covering, as its domain, a country or named *region*, supported by a suitable database. The selection of the domain for a specific search is carried out by local means. The gazetteer contains place-names and their properties, including lists of matching named places. It is supported by mechanisms for finding and collating the properties of place-names as given by combinable locality attributes, and is quite independent of the DIT.

b) The region covered by a gazetteer contains *places*. A place is a recognizable named geographical area; places can overlap, and can even extend somewhat beyond the boundary of the region. Places that are identifiable by reference to the gazetteer are called *named places*.

c) The gazetteer itself is based on strings that are *place-names*. These are used to identify (or name) named places. The name of a named place can be:

   – a single place-name, possibly in more than one word;

   – a collection of place-names, where in general one place-name corresponds to a larger area and qualifies a place-name that corresponds (within the context) to a smaller area.

d) The concept of larger and smaller areas may sometimes be usefully represented in the characteristic of scale as applied to a place. Informal examples of places of varying scale are plots, spots, villages, towns, cities, counties, provinces and countries. In general, a named place should be associated in the gazetteer with the names of encompassing places of a larger scale, even if these are not required for unique identification.

e) Place-names may also have synonyms associated with a particular place, which could (for example) represent abbreviations or alternative names. It is convenient to define a canonical name for each place, to which synonyms of component place-names may be mapped.

f) Place-names may sometimes be derived from simpler place names by using semantic components such as "Near" (e.g., "Near Tenterden"). This may conceivably be taken to define a ring-shaped place around the town of Tenterden in Kent, England, but would probably be best taken as a place-name that does not by itself define a place.

g) All places covered by the gazetteer shall have a unique canonical name consisting of a distinct set of place-names, where these names can be ordered in terms of the scale that each place-name implies in the context.

h) Places are broken down into zones in such a way that zones are always nested inside each place, and each part of a place has a corresponding zone. A zone is the building block of places in a gazetteer; every point in a region has a single zone in which it is contained.

i) Zones usually have neighbouring zones (e.g., unless effectively blocked by a geographical or major political feature such as a lake, river, sea, or mountain, or country boundary). Thus, the area defined for a place can usually be extended by including zones that are neighbours to the zones that comprise it; the extension can be carried on indefinitely a step at a time. The inclusion of a single level of neighbour extension is called the 1-extension of a place; a further level of extension is called a 2-extension, and so on. The scope of an extension may be locally adjustable (extended or reduced) to represent a practical situation, but such adjustments should be relatively scarce.

j) An entry representing a physical object may be defined to have a *location*. A location can be defined in terms of a set of zones in an appropriate zone attribute, or by identifying it as a named place by the use of one or more place-names using a locality attribute such as **locationName**, which can also be represented as a set of zones. An entry will match a place if the set of zones that comprise its locality overlap the set of zones that represent the place (possibly n-extended) that is the result of consulting the gazetteer, as described above.

k) The selection of zones, places, place-names and the compilation of their relationships is a local matter.

l) Entries that would match by equality match on the basis of strings that they contain shall continue to match (in effect bypassing zonal match).

To further qualify zonal matching, the **ZONAL-MATCHING** information non-generic object class is defined as a specialization of the **MAPPING-BASED-MATCHING** generic information object class. An instance of this information object class determines the characteristics of zonal matching.

```
ZONAL-MATCHING ::=
  MAPPING-BASED-MATCHING{ZonalSelect, TRUE, ZonalResult, zonalMatch.&id}
```

An instance of this information object class is characterized by:

a) The **&selectBy** dummy reference, if present, is by this information object class replaced by a set-of attribute types. The selection of an instance of this information object class is based on these attributes and on the attribute types represented in the search filter. An information object instance may be selected if all the attribute types represented by this component are represented in the filter. Attribute subtypes are not considered (i.e., the selection shall be based on explicitly named attributes). However, local criteria that are not defined by this Directory Specification may also be taken into account for selecting an instance. For example, the selection may partly be determined by the **baseObject** of the search argument. If this component is absent, selection is based wholly on local decision-making.

b) The **&ApplicableTo** shall specify a set of locality related attribute types as determined by local requirements, such as **localityName**, **stateOrProvinceName**, **streetName**, **postalCode**, etc.

c) The **&subtypeIncluded** component is set according to local requirements.

d) The **&combinable** dummy value reference is unconditionally replaced by TRUE.

e) The **&mappingResults** dummy type reference is by this information object class replaced by the **ZonalResult** data type.

f) The **&userControl** is set according to local requirements.

NOTE 2 – This field should in most cases take the value **TRUE**.

g) The **&exclusive** is set according to local requirements.

NOTE 3 – An information object instance of this information object class is a candidate for exclusive relaxation.

h) The **&matching-rule** is by this derived information object class set to **zonalMatch**.

i) The **&id** gives a unique identification of the instance of zonal matching algorithm.

The **ZonalSelect** data type is:

```
ZonalSelect ::= SEQUENCE OF AttributeType
```

The **ZonalResult** data type is used for indicating exception conditions for zonal matching.

```
ZonalResult ::= ENUMERATED {
  cannot-select-mapping (0),
  zero-mappings         (2),
  multiple-mappings     (3),
   ... }
```

The values:

a) **cannot-select-mapping** is the result when the information provided in the base object name and subfilter is insufficient to identify the mapping that is to be used in the zonal matching rule. The corresponding match produces a result of undefined. None of the subfilters which have mappable filter items, according to the **&applicableTo** specification, will accordingly not evaluate to **TRUE**.

NOTE 4 – Within a service-specific administrative area and for properly designed search-rules, the analysis of the search argument should have detected insufficient information in the search argument.

b) **zero-mappings** is the result when the information provided in the filter item(s) to be mapped cannot be mapped, either because no corresponding item exists in the mapping table, or because the mapping process produced zero filter items to be matched against entries. In this situation, a **serviceError** with problem **requestedServiceNotAvailable** shall be returned. The notification **component** of **CommonResults** shall contain:

i) a **searchServiceProblem** notification attribute with the value **id-pr-unmatchedKeyAttributes**; and

ii) a **filterItem** notification attribute indicating the mappable filter items unable to provide a match.

c) **multiple-mappings** is the result when the information provided in the filter item(s) can successfully be mapped to multiple entries of the gazetteer. The corresponding match produces a value TRUE, but can, nevertheless, cause the search to be abandoned with an error. In this situation, a **serviceError** with problem **requestedServiceNotAvailable** shall be returned. The notification **component** of **CommonResults** shall contain:

i) a **searchServiceProblem** notification attribute with the value **id-pr-ambiguousKeyAttributes**; and

ii) a **multipleMatchingLocalities** notification attribute as indicated by the **zonalMatch** matching rule.

The **zonalMatch** matching rule is the mapping-based matching rule associated with any instance of the **ZONAL-MATCHING** information object class.

```
zonalMatch MATCHING-RULE ::= {
  UNIQUE-MATCH-INDICATOR  multipleMatchingLocalities
  ID                      id-mr-zonalMatch }
```

This mapping-based matching rule includes the **UNIQUE-MATCH-INDICATOR** field, which implies that matching against the gazetteer shall give an unambiguous result. If several table entries match in the mapping process, a **serviceError** with problem **ambiguousKeyAttributes** shall be returned. The notification component of **CommonResults** shall contain a **multipleMatchingLocalities** notification attribute (see clause 6.14.14). A value of the **multipleMatchingLocalities** notification attribute is included for each table entry matched on the gazetteer. Each such value shall be a set-of **AttributeValueAssertion** specification that, if supplied in AND'ed **equality** filter items in each subfilter, would give a unique match against the corresponding table entry. This will allow the user in a subsequent **search** request to select one of the returned notification attribute values to be reflected in the filter.

8.9 uri Match

The *uri Match* rule compares a presented value with an attribute value and is defined as:

```
uriMatch MATCHING-RULE ::= {
  SYNTAX       UTF8String
  LDAP-SYNTAX  directoryString.&id
  LDAP-NAME    {"uriMatch"}
  ID           id-mr-uriMatch }
```

This rule conforms to IETF RFC 3986 clause 6.2.2: the two UTF8String values are normalized as described in IETF RFC 3986:

a) Case normalization: the hexadecimal digits within a percent-encoding triplet shall be normalized to use uppercase letters for digits A-F;

b) Percent-encoding normalization: any percent-encoded octet that corresponds to an unreserved character (uppercase letters, lowercase letters, digits, HYPHEN MINUS, PERIOD, LOW LINE and TILDE) shall be decoded;

c) Path segment normalization: path normalization permits the simplification of a path containing "." or ".." complete path segments. This normalization uses two buffers (an input buffer containing the path and an empty output buffer which will contain the result) and loops as follows until the input buffer is empty:

   – If the input buffer begins with a prefix of "../" or "./", then remove the prefix from the input buffer; otherwise,

   – if the input buffer begins with a prefix of "/./" or "/.", where "." is complete path segment, then replace that prefix with "/" in the input buffer; otherwise,

   – if the input buffer begins with a prefix of "/../" or "/..", where ".." is a complete path segment, then replace that prefix with "/" in the input buffer and remove the last segment and its preceding "/" (if any) from the output buffer; otherwise,

   – if the input buffer consists only of "." or "..", then remove that from the input buffer; otherwise,

   – move the first path segment in the input buffer to the end of the output buffer, including the initial "/" character (if any) and any subsequent characters up to, but not including the next "/" character of the end of the input buffer.

d) Scheme based normalization: components which are empty or equal to the default for the scheme shall be removed.

8.10 LDAP defined matching rules

8.10.1 Case exact IA5 match

The **caseExactIA5Match** rule compares an assertion value of the **IA5String** syntax to an attribute value of the same syntax after string preparation as discussed in clause 7.2.

```
caseExactIA5Match MATCHING-RULE ::= {
  SYNTAX        IA5String
  LDAP-SYNTAX   ia5String.&id
  LDAP-NAME     {"caseExactIA5Match"}
  ID            id-lmr-caseExactIA5Match }
```

The rule evaluates to TRUE if, and only if the prepared attribute value character string and the prepared assertion value character string have the same number of characters and the corresponding characters have the same code point.

8.10.2 Case ignore IA5 match

The **caseIgnoreIA5Match** rule compares an assertion value of the **IA5String** syntax to an attribute value of the same syntax after string preparation as discussed in clause 7.2.

```
caseIgnoreIA5Match MATCHING-RULE ::= {
  SYNTAX        IA5String
  LDAP-SYNTAX   ia5String.&id
  LDAP-NAME     {"caseIgnoreIA5Match"}
  ID            id-lmr-caseIgnoreIA5Match }
```

The rule evaluates to TRUE if, and only if the prepared attribute value character string and the prepared assertion value character string have the same number of characters and the corresponding characters have the same code point after case mapping.

8.10.3 Case ignore IA5 substrings match

The **caseIgnoreIA5SubstringsMatch** rule compares an assertion value of **SubstringAssertion** syntax to an attribute value of **IA5String** syntax after string preparation as discussed in clause 7.2.

```
caseIgnoreIA5SubstringsMatch MATCHING-RULE ::= {
  SYNTAX        SubstringAssertion
  LDAP-SYNTAX   substringAssertion.&id
  LDAP-NAME     {"caseIgnoreIA5SubstringsMatch"}
```

```
  ID            id-lmr-caseIgnoreIA5Match }
```

The rule is identical to the **caseIgnoreSubstringsMatch** as discussed in clause 8.1.3.

9 Definition of syntaxes

9.1 Directory syntaxes

9.1.1 UTM Coordinates syntax

```
utmCoords SYNTAX-NAME ::= {
  LDAP-DESC          "UTM Coordinates"
  DIRECTORY SYNTAX   UtmCoordinates
  ID                 id-asx-utmCoords }
```

A value which has an LDAP UTM Coordinate syntax is an encoded value of the **UtmCoordinates** data type using the General String Encoding Rules as defined by the IETF RFCs 3641, 3642 and 4792.

9.1.2 UII Format syntax

```
uiiForm SYNTAX-NAME ::= {
  LDAP-DESC          "UII Format"
  DIRECTORY SYNTAX   UiiFormat
  ID                 id-asx-uiiForm }
```

A value which has an LDAP **UiiFormat** syntax is an encoded value of the **UiiFormat** data type using the General String Encoding Rules as defined by the IETF RFCs 3641, 3642 and 4792.

9.1.3 EPC Format syntax

```
epcForm SYNTAX-NAME ::= {
  LDAP-DESC          "EPC Format"
  DIRECTORY SYNTAX   EpcFormat
  ID                 id-asx-epcForm }
```

A value which has an LDAP **EpcFormat** syntax is an encoded value of the **EpcFormat** data type using the General String Encoding Rules as defined by the IETF RFCs 3641, 3642 and 4792.

9.2 IETF syntaxes

9.2.1 Descriptors

Descriptors are defined in clause 3.4 of IETF RFC 4520. In contrast to other syntaxes, a descriptor syntax is not identified by an object identifier (as it a short descriptive name of an object identifier). The syntax of a descriptor is case insensitive and consists of a leading alphabetic character followed by alphanumeric characters and hyphens.

9.2.2 AttributeType Description syntax

```
attributeTypeDescription SYNTAX-NAME ::= {
  LDAP-DESC          "Attribute Type Description"
  DIRECTORY SYNTAX   AttributeTypeDescription
  ID                 id-lsx-attributeTypeDescription }
```

A value which has an LDAP Attribute Type Description is an ABNF encoding of an attribute type description, as specified in IETF RFC 4512 without line breaks.

9.2.3 Bit string syntax

```
bitstring SYNTAX-NAME ::= {
  LDAP-DESC          "Bit String"
  DIRECTORY SYNTAX   BIT STRING
  ID                 id-lsx-bitstring }
```

A value which has an LDAP bit string syntax is a string of "0" and"1" characters enclosed by single quotes followed by character "B", as specified by IETF RFC 4517.

When converting from a BER encoded bit string, the trailing bits added for octet alignment shall not be included.

9.2.4 Boolean syntax

```
boolean SYNTAX-NAME ::= {
  LDAP-DESC          "Boolean"
  DIRECTORY SYNTAX   BOOLEAN
  ID                 id-lsx-boolean }
```

A value of the LDAP Boolean syntax is either the character string "TRUE" or "FALSE", as specified IETF RFC 4517.

9.2.5 Country string

```
countryString SYNTAX-NAME ::= {
  LDAP-DESC          "Country String"
  DIRECTORY SYNTAX   CountryName
  ID                 id-lsx-countryString }
```

A value which has an LDAP Country String syntax is a two printable character string, as specified by IETF RFC 4517.

9.2.6 DN syntax

```
dn SYNTAX-NAME ::= {
  LDAP-DESC          "DN"
  DIRECTORY SYNTAX   DistinguishedName
  ID                 id-lsx-dn }
```

A value which has an LDAP DN syntax is a distinguished name in the LDAP format using the string representation, as specified by IETF RFC 4514.

9.2.7 Delivery method

```
deliveryMethod SYNTAX-NAME ::= {
  LDAP-DESC          "Delevery Method"
  DIRECTORY SYNTAX   PreferredDeliveryMethod
  ID                 id-lsx-deliverylMethod }
```

A value which has an LDAP **DeliveryMethod** syntax is a specification of a preferable delivery method, as specified by IETF RFC 4517.

9.2.8 Directory string syntax

```
directoryString SYNTAX-NAME ::= {
  DESC               "Directory String"
  DIRECTORY SYNTAX   UnboundedDirectoryString
  ID                 id-lsx-directoryString }
```

The LDAP **DirectoryString** syntax is a **UTF8String** syntax. When converting to the LDAP syntax, a directory string has to be converted to the UTF8 encoding of the UCS. When an LDAP directory string is converted to a directory string as defined by this Directory Specification, it is the local choice of the LDAP requester as to which of the directory string alternatives to take.

9.2.9 DIT Content Rule Description syntax

```
dITContentRuleDescription SYNTAX-NAME ::= {
  DESC               "DIT Content Rule Description"
  DIRECTORY SYNTAX   DITContentRuleDescription
  ID                 id-lsx-dITContentRuleDescription }
```

A value which has an LDAP **DITContentRuleDescription** syntax is a specification of a content rule. It is expressed in an ABNF encoding, as specified by IETF RFC 4512.

9.2.10 DIT Structure Rule Description syntax

```
dITStructureRuleDescription SYNTAX-NAME ::= {
  DESC               "DIT StructureRule Description"
  DIRECTORY SYNTAX   DITStructureRuleDescription
  ID                 id-lsx-dITStructureRuleDescription }
```

A value which has an LDAP **DITStructureRuleDescription** syntax is a specification of a structure rule. It is expressed in an ABNF encoding, as specified by IETF RFC 4512.

9.2.11 Enhanced Guide syntax

```
enhancedGuide SYNTAX-NAME ::= {
  DESC               "Enhanced Guide"
  DIRECTORY SYNTAX   EnhancedGuide
  ID                 id-lsx-enhancedGuide }
```

A value which has an LDAP **EnhancedGuide** syntax is a specification of an enhanced guide. It is expressed in an ABNF encoding, as specified by IETF RFC 4517.

9.2.12 Facsimile Telephone Number syntax

```
facsimileTelephoneNr SYNTAX-NAME ::= {
  DESC               "Facsimile Telephone Number"
  DIRECTORY SYNTAX   FacsimileTelephoneNumber
  ID                 id-lsx-facsimileTelephoneNr }
```

A value which has an LDAP **FacsimileTelephoneNr** syntax is a specification of a facsimile telephone number. It is expressed in an ABNF encoding, as specified by IETF RFC 4517.

9.2.13 Fax syntax

```
fax SYNTAX-NAME ::= {
  DESC               "Fax"
  DIRECTORY SYNTAX   NULL
  ID                 id-lsx-fax }
```

A value which has an LDAP **Fax** syntax is a specification of a fax image. Its encoding is specified by IETF RFC 4517.

9.2.14 Generalized Time syntax

```
generalizedTime SYNTAX-NAME ::= {
  DESC               "Generalized Time"
  DIRECTORY SYNTAX   GeneralizedTime
  ID                 id-lsx-generalizedTime }
```

A value which has an LDAP **GeneralizedTime** syntax is a specification of a time value in the generalized time format. It is expressed in an ABNF encoding, as specified by IETF RFC 4517.

9.2.15 Guide syntax

```
guide SYNTAX-NAME ::= {
  DESC               "Guide"
  DIRECTORY SYNTAX   Guide
  ID                 id-lsx-guide }
```

A value which has an LDAP **Guide** syntax is a specification of a suggested criterion for constructing a filter. It is expressed in an ABNF encoding, as specified by IETF RFC 4517.

9.2.16 IA5 String syntax

```
ia5String SYNTAX-NAME ::= {
  DESC               "IA5 String"
  DIRECTORY SYNTAX   IA5String
  ID                 id-lsx-ia5String }
```

A value which has an LDAP **Ia5string** syntax consisting of characters from the International Alphabet 5 (the ASCII character set), as specified by Rec. ITU-T X.680 | ISO/IEC 8824-1.

9.2.17 INTEGER syntax

```
integer SYNTAX-NAME ::= {
  DESC               "INTEGER"
  DIRECTORY SYNTAX   INTEGER
  ID                 id-lsx-integer }
```

A value which has an LDAP **Integer** syntax consisting of numeric characters, as specified by IETF RFC 4517.

9.2.18 JPEG syntax

```
jpeg SYNTAX-NAME ::= {
  DESC               "JPEG"
  DIRECTORY SYNTAX   NULL
  ID                 id-lsx-jpeg }
```

A value which has an LDAP **IJPEG** syntax is an octet string constrained to a JPEG image, as specified by IETF RFC 4517.

9.2.19 Matching Rule Description syntax

```
matchingRuleDescription SYNTAX-NAME ::= {
  DESC               "Matching Rule Description"
  DIRECTORY SYNTAX   MatchingRuleDescription
  ID                 id-lsx-MatchingRuleDescription }
```

A value which has an LDAP **MatchingRuleDescription** syntax is a specification of a matching rule description. It is expressed in an ABNF encoding, as specified by IETF RFC 4512.

9.2.20 Matching Rule Use Description syntax

```
matchingRuleUseDescription SYNTAX-NAME ::= {
  DESC               "Matching Rule Use Description"
  DIRECTORY SYNTAX   MatchingRuleUseDescription
  ID                 id-lsx-matchingRuleUseDescription }
```

A value which has an LDAP **MatchingRuleUseDescription** syntax is a specification of a matching rule use description. It is expressed in an ABNF encoding, as specified by IETF RFC 4512.

9.2.21 Name and Optional UID syntax

```
nameAndOptionalUID SYNTAX-NAME ::= {
  DESC               "Name And Optional UID"
  DIRECTORY SYNTAX   NameAndOptionalUID
  ID                 id-lsx-nameAndOptionalUID }
```

A value which has an LDAP **NameAndOptionalUID** is a distinguished name optionally followed by bit string forming a unique ID, as specified by IETF RFC 4517.

9.2.22 Name Form Description syntax

```
nameFormDescription SYNTAX-NAME ::= {
  DESC               "Name Form Description"
  DIRECTORY SYNTAX   NameFormDescription
  ID                 id-lsx-nameFormDescription }
```

A value which has an LDAP **NameFormDescription** syntax is a specification of a name form. It is expressed in an ABNF encoding, as specified by IETF RFC 4512.

9.2.23 Numeric String syntax

```
numericString SYNTAX-NAME ::= {
  DESC               "Numeric String"
  DIRECTORY SYNTAX   NumericString
  ID                 id-lsx-numericString }
```

A value which has an LDAP **NumericString** syntax consisting of numeric characters and spaces, as specified by IETF RFC 4517.

9.2.24 Object Class Description syntax

```
objectClassDescription SYNTAX-NAME ::= {
  DESC               "Object Class Description"
  DIRECTORY SYNTAX   ObjectClassDescription
  ID                 id-lsx-objectClassDescription }
```

A value which has an LDAP **ObjectClassDescription** syntax is a specification of an object class. It is expressed in an ABNF encoding, as specified by IETF RFC 4512.

9.2.25 OID syntax

```
oid SYNTAX-NAME ::= {
  DESC               "OID"
  DIRECTORY SYNTAX   OBJECT IDENTIFIER
  ID                 id-lsx-oid }
```

A value which has an LDAP **OID** syntax is an object identifier either defined using the dot-decimal format or a descriptor, as specified by IETF RFC 4512.

9.2.26 Other Mailbox syntax

```
otherMailbox SYNTAX-NAME ::= {
  DESC               "Other Mailbox"
  DIRECTORY SYNTAX   NULL
  ID                 id-lsx-otherMailbox }
```

A value which has an LDAP **OtherMailbox** syntax is a specification of an electronic mailbox. It is expressed in an ABNF encoding, as specified by IETF RFC 4517.

9.2.27 Octet String syntax

```
octetString SYNTAX-NAME ::= {
  DESC               "Octet String"
  DIRECTORY SYNTAX   OCTET STRING
  ID                 id-lsx-octetString }
```

A value which has an LDAP **OctetString** syntax is a string of zero or more octets encoded, as specified by IETF RFC 4512.

9.2.28 Postal Address syntax

```
postalAddr SYNTAX-NAME ::= {
  DESC               "Postal Address"
  DIRECTORY SYNTAX   PostalAddress
  ID                 id-lsx-postalAddr }
```

A value which has an LDAP **postalAddr** syntax is a specification of a postal address. It is expressed in an ABNF encoding, as specified by IETF RFC 4517.

9.2.29 Printable String syntax

```
printableString SYNTAX-NAME ::= {
  DESC               "Printable String"
  DIRECTORY SYNTAX   PrintableString
  ID                 id-lsx-printableString }
```

A value which has an LDAP **PrintableString** syntax is a string of one or more characters with a repertoire equal to PrintableString, as specified by Rec. ITU-T X.680 | ISO/IEC 8824-1.

9.2.30 Subtree Specification syntax

```
subtreeSpec SYNTAX-NAME ::= {
  DESC               "SubtreeSpecification"
  DIRECTORY SYNTAX   SubtreeSpecification
  ID                 id-lsx-subtreeSpec }
```

The LDAP Subtree Specification syntax is specified in IETF RFC 3672.

The LDAP syntax is based on the Generic String Encoding Rules (GSER) for ASN.1as defined in IETF RFCs 3641, 3642 and 4792.

9.2.31 Telephone Number syntax

```
telephoneNr SYNTAX-NAME ::= {
  DESC               "Telephone Number"
  DIRECTORY SYNTAX   TelephoneNumber
  ID                 id-lsx-telephoneNr }
```

A value which has an LDAP **TelephoneNumber** syntax is a specification of a telephone number, as specified by IETF RFC 4517.

9.2.32 Telex Number syntax

```
telexNr SYNTAX-NAME ::= {
  DESC               "Telex Number"
  DIRECTORY SYNTAX   TelexNumber
  ID                 id-lsx-telexNr }
```

A value of the LDAP **TelexNumber** syntax is a telex number, country code and answerback code of a telex terminal, as specified by IETF RFC 4517.

9.2.33 UTC Time syntax

```
utcTime SYNTAX-NAME ::= {
  DESC               "UTC Time"
  DIRECTORY SYNTAX   UTCTime
  ID                 id-lsx-utcTime }
```

A value of the LDAP **UTCTime** syntax is a character string giving time information, as specified by IETF RFC 4517.

9.2.34 LDAP Syntax Description syntax

```
ldapSyntaxDescription SYNTAX-NAME ::= {
  DESC               "LDAP Syntax Description"
  DIRECTORY SYNTAX   NULL
  ID                 id-lsx-ldapSyntaxDescription }
```

A value of the LDAP **LDAPSyntaxDescription** syntax is a description of an LDAP syntax, as specified by IETF RFC 4517. There is no corresponding syntax defined by this Directory Specification.

9.2.35 Substring Assertion syntax

```
substringAssertion SYNTAX-NAME ::= {
  DESC               "Substring Assertion"
  DIRECTORY SYNTAX   SubstringAssertion
  ID                 id-lsx-substringAssertion }
```

A value of the **SubstringAssertion** syntax is a sequence of zero, one, or more character substrings used as an argument for substring extensible matching of character string attribute values, as specified by IETF RFC 4517.

SECTION 4 – CONTEXTS

10 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.

10.1 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.

10.2 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 to 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 SIZE (1..MAX) OF Period},
  notThisTime    BOOLEAN DEFAULT FALSE,
  timeZone       TimeZone OPTIONAL,
  ... }

Period ::= SEQUENCE {
  timesOfDay  [0]  SET SIZE (1..MAX) 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 {
  intNamedDays 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 period, since this also acts as 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 may be present.

The final element in a **Period** is assumed to be valid for all time periods of a 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 **INTEGER**s 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 **INTEGER**s 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 **INTEGER**s 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 **INTEGER**s 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 **INTEGER**s 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 **INTEGER**s, the **INTEGER**s 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 the 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:

```
temporal1 TimeSpecification ::= {
  time periodic:{

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

b) Every Monday would be expressed as:

```
temporal2 TimeSpecification ::= {
  time 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:

```
temporal3 TimeSpecification ::= {
  time periodic:{

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

    { days       intDay:{7},
      weeks      intWeek:{1,2,3,4,5},
      months     intMonth:{1} },

    { days       intDay{3},
      weeks      intWeek:{1,2,3,4,5},
      months     intMonth:{2,3} }
  }
}
```

d) All of August 1996 would be expressed as:

```
temporal4 TimeSpecification ::= {
  time periodic:{

    { months     intMonth:{8},
      years      {1996} }
  }
}
```

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

```
temporal5 TimeSpecification ::= {
  time periodic:{

    { days       intDay:{1},
      months     allMonths:NULL }
  }
}
```

10.3 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  UnboundedDirectoryString,
  ... }
```

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 9945:2009, *Information technology – Portable Operating System Interface (POSIX) – Base Specifications, Issue 7*. ISO/IEC 15897 specifies procedures for registration of cultural elements.

10.4 LDAP Attribute Option Context

The *LDAP Attribute Option Context* is used to provide an alignment between ITU-T X.500 contexts and LDAP attribute options.

```
ldapAttributeOptionContext CONTEXT ::= {
  WITH SYNTAX  AttributeOptionList
  ASSERTED AS  AttributeOptionList
  ABSENT-MATCH FALSE
  ID           id-avc-ldapAttributeOption }

AttributeOptionList ::= SEQUENCE OF UTF8String
```

A list of options as the context value provides the closest, most natural fit of the context concept as defined by these Directory Specifications to ReLDAP attribute options. Each LDAP subtyping attribute option is mapped to a single **UTF8String** value in the list. Two **ldapAttributeOptionContext** values are equal if they contain the same list of strings, in any order, using a case ignore comparison. An **AttributeOptionList** in a **ContextAssertion** matches an **AttributeOptionList** in a stored context value if it is a subset of, or equal to, the stored list, ignoring letter case and the order of the options.

NOTE 1 – **AttributeOptionList** is implemented as a **SEQUENCE OF** to simplify DER encoding.

NOTE 2 – LDAP attribute options are restricted to the characters 'A' to 'Z', 'a' to 'z', '0' to '9' and hyphen, so **PrintableString** rather than **UTF8String** would be sufficient. However, the underlying character set for attribute options is UTF8 and a future LDAP extension might make use of the wider repertoire. Therefore, **UTF8String** was chosen to future-proof the specification.

An empty **AttributeOptionList** is specifically allowed. In LDAP, a particular value is permitted to simultaneously appear in the base attribute and in any of its optioned subtypes, e.g., (in LDIF format):

```
description: This is a string
description;lang-en: This is a string
description;lang-en;lang-en-us: This is a string
```

In Rec. ITU-T X.500 | ISO/IEC 9594-1, this is represented as the single value "This is a string" with a single Context which has the **contextType id-avc-ldapAttributeOption**, and three **contextValues**: { }, { "lang-en" } and { "lang-en", "lang-en-us" }. That is, an empty **AttributeOptionList**, an **AttributeOptionList** containing the single value "lang-en" and an **AttributeOptionList** containing the two values "lang-en" and "lang-en-us".

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) 7}
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

  -- from Rec. ITU-T X.501 | ISO/IEC 9594-2

  authenticationFramework, certificateExtensions,
  directoryAbstractService, id-at, id-avc, id, id-asx, id-cat, id-coat, id-lmr, id-lsx,
  id-mr, id-not, id-pr, informationFramework, schemaAdministration,
  serviceAdministration, passwordPolicy
    FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 7}

  Attribute{}, ATTRIBUTE, AttributeType, AttributeValueAssertion, CONTEXT,
  ContextAssertion, DistinguishedName, distinguishedNameMatch,
  MAPPING-BASED-MATCHING{}, MATCHING-RULE, OBJECT-CLASS,
  objectIdentifierMatch, SubtreeSpecification, SupportedAttributes, SYNTAX-NAME
    FROM InformationFramework informationFramework

  AttributeCombination, ContextCombination, MRMapping
    FROM ServiceAdministration serviceAdministration

  AttributeTypeDescription, DITContentRuleDescription, DITStructureRuleDescription,
  MatchingRuleDescription, MatchingRuleUseDescription, NameFormDescription,
  ObjectClassDescription
    FROM SchemaAdministration schemaAdministration

  -- from Rec. ITU-T X.509 | ISO/IEC 9594-8

  AlgorithmIdentifier{}, Certificate, CertificateList, CertificatePair,
  SupportedAlgorithm, SupportedAlgorithms
     FROM AuthenticationFramework authenticationFramework

PwdAlphabet, PwdVocabulary, UserPwd
    FROM PasswordPolicy passwordPolicy

 -- from Rec. ITU-T X.511 | ISO/IEC 9594-3

  FilterItem, HierarchySelections, SearchControlOptions, ServiceControlOptions
    FROM DirectoryAbstractService directoryAbstractService

  -- from Rec. ITU-T X.411 | ISO/IEC 10021-4
  G3FacsimileNonBasicParameters
    FROM MTSAbstractService {joint-iso-itu-t mhs(6) mts(3) modules(0)
      mts-abstract-service(1) version-1999(1)} ;

/*from IETF RFC 3727

The following import is provided for information only (see clause 7.2.16), it is not
referenced by any ASN.1 construct within these Directory Specifications. Note that
the ASN.1 module in RFC 3727 imports from the InformationFramework module of edition
```

```
4 of Rec. ITU-T X.501 | ISO/IEC 9594-2. A specification importing from both these
Directory Specifications and from RFC 3727 should take corrective actions, e.g.,
by making a copy of the ASN.1 module of RFC 3727 and then update the IMPORT statement.

  allComponentsMatch, componentFilterMatch, directoryComponentsMatch, presentMatch,
rdnMatch
    FROM ComponentMatching {iso(1) 2 36 79672281 xed(3) module (0) component-matching(4)}
*/

-- Directory string type

UnboundedDirectoryString ::= CHOICE {
  teletexString    TeletexString(SIZE (1..MAX)),
  printableString  PrintableString(SIZE (1..MAX)),
  bmpString        BMPString(SIZE (1..MAX)),
  universalString  UniversalString(SIZE (1..MAX)),
  uTF8String       UTF8String(SIZE (1..MAX)) }

DirectoryString{INTEGER:maxSize} ::= CHOICE {
  teletexString    TeletexString(SIZE (1..maxSize,...)),
  printableString  PrintableString(SIZE (1..maxSize,...)),
  bmpString        BMPString(SIZE (1..maxSize,...)),
  universalString  UniversalString(SIZE (1..maxSize,...)),
  uTF8String       UTF8String(SIZE (1..maxSize,...)) }

-- Attribute types

knowledgeInformation ATTRIBUTE ::= {
  WITH SYNTAX               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  OBSOLETE                  TRUE
  ID                        id-at-knowledgeInformation }

name ATTRIBUTE ::= {
  WITH SYNTAX               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"name"}
  ID                        id-at-name }

commonName ATTRIBUTE ::= {
  SUBTYPE OF                name
  WITH SYNTAX               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"cn", "commonName"}
  ID                        id-at-commonName }

surname ATTRIBUTE ::= {
  SUBTYPE OF                name
  WITH SYNTAX               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"sn"}
  ID                        id-at-surname }

givenName ATTRIBUTE ::= {
  SUBTYPE OF                name
  WITH SYNTAX               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"givenName"}
  ID                        id-at-givenName }

initials ATTRIBUTE ::= {
  SUBTYPE OF                name
  WITH SYNTAX               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"initials"}
  ID                        id-at-initials }

generationQualifier ATTRIBUTE ::= {
  SUBTYPE OF                name
```

```
  WITH SYNTAX                UnboundedDirectoryString
  LDAP-SYNTAX                directoryString.&id
  LDAP-NAME                  {"generationQualifier"}
  ID                         id-at-generationQualifier }

uniqueIdentifier ATTRIBUTE ::= {
  WITH SYNTAX                UniqueIdentifier
  EQUALITY MATCHING RULE     bitStringMatch
  LDAP-SYNTAX                bitString.&id
  LDAP-NAME                  {"x500UniqueIdentifier"}
  ID                         id-at-uniqueIdentifier }

UniqueIdentifier ::= BIT STRING

dnQualifier ATTRIBUTE ::= {
  WITH SYNTAX                PrintableString
  EQUALITY MATCHING RULE     caseIgnoreMatch
  ORDERING MATCHING RULE     caseIgnoreOrderingMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX                printableString.&id
  LDAP-NAME                  {"dnQualifier"}
  ID                         id-at-dnQualifier }

serialNumber ATTRIBUTE ::= {
  WITH SYNTAX                PrintableString(SIZE (1..MAX))
  EQUALITY MATCHING RULE     caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX                printableString.&id
  LDAP-NAME                  {"serialNumber"}
  ID                         id-at-serialNumber }

pseudonym ATTRIBUTE ::= {
  SUBTYPE OF                 name
  WITH SYNTAX                UnboundedDirectoryString
  ID                         id-at-pseudonym }

uUIDPair ATTRIBUTE ::= {
  WITH SYNTAX                UUIDPair
  EQUALITY MATCHING RULE     uUIDPairMatch
  ID                         id-at-uuidpair }

UUIDPair ::= SEQUENCE {
  issuerUUID   UUID,
  subjectUUID  UUID,
  ... }

UUID ::= OCTET STRING(SIZE (16)) -- UUID format only

uri ATTRIBUTE ::= {
  WITH SYNTAX                URI
  EQUALITY MATCHING RULE     uriMatch
  LDAP-SYNTAX                directoryString.&id
  LDAP-NAME                  {"uri"}
  ID                         id-at-uri }

URI ::= UTF8String

urn ATTRIBUTE ::= {
  SUBTYPE OF                 uri
  LDAP-SYNTAX                directoryString.&id
  LDAP-NAME                  {"urn"}
  ID                         id-at-urn }

url ATTRIBUTE ::= {
  SUBTYPE OF                 uri
  LDAP-SYNTAX                directoryString.&id
  LDAP-NAME                  {"url"}
  ID                         id-at-url }

countryName ATTRIBUTE ::= {
  SUBTYPE OF                 name
```

```
  WITH SYNTAX               CountryName
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               countryString.&id
  LDAP-NAME                 {"c"}
  ID                        id-at-countryName }

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

localityName ATTRIBUTE ::= {
  SUBTYPE OF                name
  WITH SYNTAX               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"l"}
  ID                        id-at-localityName }

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

stateOrProvinceName ATTRIBUTE ::= {
  SUBTYPE OF                name
  WITH SYNTAX               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"st"}
  ID                        id-at-stateOrProvinceName }

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

streetAddress ATTRIBUTE ::= {
  WITH SYNTAX               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"street"}
  ID                        id-at-streetAddress }

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

houseIdentifier ATTRIBUTE ::= {
  WITH SYNTAX               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"houseIdentifier"}
  ID                        id-at-houseIdentifier }

utmCoordinates  ATTRIBUTE  ::=  {
  WITH SYNTAX               UtmCoordinates
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               utmCoords.&id
  LDAP-NAME                 {"utmCoordinates"}
  ID                        id-at-utmCoordinates }

UtmCoordinates ::= SEQUENCE {
  zone      PrintableString,
  easting   NumericString,
  northing  NumericString }

organizationName ATTRIBUTE ::= {
  SUBTYPE OF                name
  WITH SYNTAX               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"o"}
  ID                        id-at-organizationName }
```

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

organizationalUnitName ATTRIBUTE ::= {
  SUBTYPE OF                name
  WITH SYNTAX               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"ou"}
  ID                        id-at-organizationalUnitName }

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

title ATTRIBUTE ::= {
  SUBTYPE OF                name
  WITH SYNTAX               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"title"}
  ID                        id-at-title }

organizationIdentifier ATTRIBUTE ::= {
  WITH SYNTAX               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"organizationIdentifier"}
  ID                        id-at-organizationIdentifier }

description ATTRIBUTE ::= {
  WITH SYNTAX               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"description"}
  ID                        id-at-description }

searchGuide ATTRIBUTE ::= {
  WITH SYNTAX               Guide
  LDAP-SYNTAX               guide.&id
  LDAP-NAME                 {"searchGuide"}
  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
  LDAP-SYNTAX               enhancedGuide.&id
```

```
  LDAP-NAME                {"enhancedSearchGuide"}
  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              UnboundedDirectoryString
  EQUALITY MATCHING RULE   caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX              directoryString.&id
  LDAP-NAME                {"businessCategory"}
  ID                       id-at-businessCategory }

postalAddress ATTRIBUTE ::= {
  WITH SYNTAX              PostalAddress
  EQUALITY MATCHING RULE   caseIgnoreListMatch
  SUBSTRINGS MATCHING RULE caseIgnoreListSubstringsMatch
  LDAP-SYNTAX              postalAddr.&id
  LDAP-NAME                {"postalAddress"}
  ID                       id-at-postalAddress }

PostalAddress ::= SEQUENCE SIZE (1..MAX) OF UnboundedDirectoryString

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

postalCode ATTRIBUTE ::= {
  WITH SYNTAX              UnboundedDirectoryString
  EQUALITY MATCHING RULE   caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX              directoryString.&id
  LDAP-NAME                {"postalCode"}
  ID                       id-at-postalCode }

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

postOfficeBox ATTRIBUTE ::= {
  WITH SYNTAX              UnboundedDirectoryString
  EQUALITY MATCHING RULE   caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX              directoryString.&id
  LDAP-NAME                {"postOfficeBox"}
  ID                       id-at-postOfficeBox }

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

physicalDeliveryOfficeName ATTRIBUTE ::= {
  WITH SYNTAX              UnboundedDirectoryString
  EQUALITY MATCHING RULE   caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX              directoryString.&id
  LDAP-NAME                {"physicalDeliveryOfficeName"}
  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
  LDAP-SYNTAX               printableString.&id
  LDAP-NAME                 {"telephoneNumber"}
  ID                        id-at-telephoneNumber }

TelephoneNumber ::= PrintableString(SIZE (1..ub-telephone-number))
-- String complying with Rec. ITU-T E.123 only

ub-telephone-number INTEGER ::= 32

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

telexNumber ATTRIBUTE ::= {
  WITH SYNTAX               TelexNumber
  LDAP-SYNTAX               telexNr.&id
  LDAP-NAME                 {"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)),
  ... }

ub-telex-number INTEGER ::= 14
ub-country-code INTEGER ::= 4
ub-answerback   INTEGER ::= 8

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

facsimileTelephoneNumber ATTRIBUTE ::= {
  WITH SYNTAX               FacsimileTelephoneNumber
  EQUALITY MATCHING RULE    facsimileNumberMatch
  SUBSTRINGS MATCHING RULE facsimileNumberSubstringsMatch
  LDAP-SYNTAX               facsimileTelephoneNr.&id
  LDAP-NAME                 {"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               X121Address
  EQUALITY MATCHING RULE    numericStringMatch
  SUBSTRINGS MATCHING RULE numericStringSubstringsMatch
  LDAP-SYNTAX               numericString.&id
  LDAP-NAME                 {"x121Address"}
  ID                        id-at-x121Address }

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

```
ub-x121-address INTEGER ::= 15

internationalISDNNumber ATTRIBUTE ::= {
  WITH SYNTAX               InternationalISDNNumber
  EQUALITY MATCHING RULE    numericStringMatch
  SUBSTRINGS MATCHING RULE numericStringSubstringsMatch
  LDAP-SYNTAX               numericString.&id
  LDAP-NAME                 {"internationalISDNNumber"}
  ID                        id-at-internationalISDNNumber }

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

ub-international-isdn-number INTEGER ::= 16

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

registeredAddress ATTRIBUTE ::= {
  SUBTYPE OF                postalAddress
  WITH SYNTAX               PostalAddress
  LDAP-SYNTAX               postalAddr.&id
  LDAP-NAME                 {"registeredAddress"}
  ID                        id-at-registeredAddress }

destinationIndicator ATTRIBUTE ::= {
  WITH SYNTAX               DestinationIndicator
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               printableString.&id
  LDAP-NAME                 {"destinationIndicator"}
  ID                        id-at-destinationIndicator }

DestinationIndicator ::= PrintableString(SIZE (1..MAX))
-- alphabetical characters only

communicationsService ATTRIBUTE ::= {
  WITH SYNTAX              CommunicationsService
  EQUALITY MATCHING RULE   objectIdentifierMatch
  ID                       id-at-communicationsService }

CommunicationsService ::= OBJECT IDENTIFIER

communicationsNetwork ATTRIBUTE ::= {
  WITH SYNTAX               CommunicationsNetwork
  EQUALITY MATCHING RULE    objectIdentifierMatch
  SINGLE VALUE              TRUE
  ID                        id-at-communicationsNetwork }

CommunicationsNetwork ::= OBJECT IDENTIFIER

preferredDeliveryMethod ATTRIBUTE ::= {
  WITH SYNTAX               PreferredDeliveryMethod
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               deliveryMethod.&id
  LDAP-NAME                 {"preferredDeliveryMethod"}
  ID                        id-at-preferredDeliveryMethod }

PreferredDeliveryMethod ::= 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) }

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
  LDAP-SYNTAX               dn.&id
  LDAP-NAME                 {"distinguishedName"}
  ID                        id-at-distinguishedName }

member ATTRIBUTE ::= {
  SUBTYPE OF                distinguishedName
  LDAP-SYNTAX               dn.&id
  LDAP-NAME                 {"member"}
  ID                        id-at-member }

uniqueMember ATTRIBUTE ::= {
  WITH SYNTAX               NameAndOptionalUID
  EQUALITY MATCHING RULE    uniqueMemberMatch
  LDAP-SYNTAX               nameAndOptionalUID.&id
  LDAP-NAME                 {"uniqueMember"}
  ID                        id-at-uniqueMember }

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

owner ATTRIBUTE ::= {
  SUBTYPE OF                distinguishedName
  LDAP-SYNTAX               dn.&id
  LDAP-NAME                 {"owner"}
  ID                        id-at-owner }

roleOccupant ATTRIBUTE ::= {
  SUBTYPE OF                distinguishedName
  LDAP-SYNTAX               dn.&id
  LDAP-NAME                 {"roleOccupant"}
  ID                        id-at-roleOccupant }

seeAlso ATTRIBUTE ::= {
  SUBTYPE OF                distinguishedName
  LDAP-SYNTAX               dn.&id
  LDAP-NAME                 {"seeAlso"}
  ID                        id-at-seeAlso }
```

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

--  Hierarchical attribute types

oidC1 ATTRIBUTE ::= {
  WITH SYNTAX               INTEGER
  EQUALITY MATCHING RULE    integerMatch
  SINGLE VALUE              TRUE
  ID                        id-oidC1 }

oidC2 ATTRIBUTE ::= {
  WITH SYNTAX               INTEGER
  EQUALITY MATCHING RULE    integerMatch
  SINGLE VALUE              TRUE
  ID                        id-oidC2 }

oidC ATTRIBUTE ::= {
  WITH SYNTAX               INTEGER
  EQUALITY MATCHING RULE    integerMatch
  SINGLE VALUE              TRUE
  ID                        id-oidC }

urnC ATTRIBUTE ::= {
  WITH SYNTAX               PrintableString
  EQUALITY MATCHING RULE    caseExactMatch
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               printableString.&id
  LDAP-NAME                 {"urnC"}
  ID                        id-at-urnC }

--  Attribute types for tag-based identification

tagOid ATTRIBUTE ::= {
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               oid.&id
  LDAP-NAME                 {"tagOid"}
  ID                        id-at-tagOid }

uiiFormat ATTRIBUTE ::= {
  WITH SYNTAX               UiiFormat
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               uiiForm.&id
  LDAP-NAME                 {"uiiFormat"}
  ID                        id-at-uiiFormat }

UiiFormat ::= SEQUENCE {
  baseObject   URI  OPTIONAL,
  subset       ENUMERATED {
    baseObject    (0),
    oneLevel      (1),
    wholeSubtree (2) } DEFAULT baseObject,
  next         CHOICE {
    length       INTEGER,
    filter       UiiFilter } }

UiiFilter ::= CHOICE {
  item   [0]  UiiItem,
  and    [1]  SET OF UiiFilter,
  or     [2]  SET OF UiiFilter,
  not    [3]  UiiFilter }

UiiItem ::= SEQUENCE {
  type   ATTRIBUTE.&id,
  length INTEGER OPTIONAL }
```

```
uiiInUrn ATTRIBUTE ::= {
  WITH SYNTAX               UTF8String
  EQUALITY MATCHING RULE    caseExactMatch
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"uiiInUrn"}
  ID                        id-at-uiiInUrn }

contentUrl ATTRIBUTE ::= {
  SUBTYPE OF                url
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"contentUrl"}
  ID                        id-at-contentUrl }

uii ATTRIBUTE ::= {
  WITH SYNTAX               BIT STRING
  EQUALITY MATCHING RULE    bitStringMatch
  LDAP-SYNTAX               bitString.&id
  LDAP-NAME                 {"uii"}
  ID                        id-at-uii }

epc ATTRIBUTE ::= {
  WITH SYNTAX               BIT STRING
  EQUALITY MATCHING RULE    bitStringMatch
  LDAP-SYNTAX               bitString.&id
  LDAP-NAME                 {"epc"}
  ID                        id-at-epc }

tagAfi ATTRIBUTE ::= {
  WITH SYNTAX               OCTET STRING
  EQUALITY MATCHING RULE    octetStringMatch
  LDAP-SYNTAX               octetString.&id
  LDAP-NAME                 {"tagAfi"}
  ID                        id-at-tagAfi }

epcFormat  ATTRIBUTE  ::=  {
  WITH SYNTAX               EpcFormat
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               epcForm.&id
  LDAP-NAME                 {"epcFormat"}
  ID                        id-at-epcFormat }

EpcFormat ::= SEQUENCE {
  fields          SEQUENCE SIZE (1..MAX) OF SEQUENCE {
    bits            INTEGER,
    charField       CHOICE {
      characters  [0] INTEGER,
      maxValue    [1] INTEGER },
    result          ENUMERATED {
      numericPad      (0),
      numeric         (1),
      alpha7bits      (2) } DEFAULT numericPad },
  digitShift  [0] INTEGER                       OPTIONAL,
  checkCalc   [1] INTEGER                       OPTIONAL,
  urnPrefix       UTF8String                    OPTIONAL }

epcInUrn ATTRIBUTE ::= {
  SUBTYPE OF                urn
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"epcInUrn"}
  ID                        id-at-epcInUrn }

ldapUrl ATTRIBUTE ::= {
  SUBTYPE OF                url
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"ldapUrl"}
  ID                        id-at-ldapUrl }

tagLocation ATTRIBUTE ::= {
  SUBTYPE OF                utmCoordinates
```

```
  SINGLE VALUE              TRUE
  LDAP-SYNTAX               utmCoords.&id
  LDAP-NAME                 {"tagLocation"}
  ID                        id-at-tagLocation }

-- Notification attributes

dSAProblem ATTRIBUTE ::= {
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  ID                        id-not-dSAProblem }

searchServiceProblem ATTRIBUTE ::= {
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  SINGLE VALUE              TRUE
  ID                        id-not-searchServiceProblem }

serviceType ATTRIBUTE ::= {
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  SINGLE VALUE              TRUE
  ID                        id-not-serviceType }

attributeTypeList ATTRIBUTE ::= {
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  ID                        id-not-attributeTypeList }

matchingRuleList ATTRIBUTE ::= {
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  ID                        id-not-matchingRuleList }

filterItem ATTRIBUTE ::= {
  WITH SYNTAX               FilterItem
  ID                        id-not-filterItem }

attributeCombinations ATTRIBUTE ::= {
  WITH SYNTAX               AttributeCombination
  ID                        id-not-attributeCombinations }

contextTypeList ATTRIBUTE ::= {
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  ID                        id-not-contextTypeList }

contextList ATTRIBUTE ::= {
  WITH SYNTAX               ContextAssertion
  ID                        id-not-contextList }

contextCombinations ATTRIBUTE ::= {
  WITH SYNTAX               ContextCombination
  ID                        id-not-contextCombinations }

hierarchySelectList ATTRIBUTE ::= {
  WITH SYNTAX               HierarchySelections
  SINGLE VALUE              TRUE
  ID                        id-not-hierarchySelectList }

searchControlOptionsList ATTRIBUTE ::= {
  WITH SYNTAX               SearchControlOptions
  SINGLE VALUE              TRUE
  ID                        id-not-searchControlOptionsList }

serviceControlOptionsList ATTRIBUTE ::= {
  WITH SYNTAX               ServiceControlOptions
  SINGLE VALUE              TRUE
  ID                        id-not-serviceControlOptionsList }

multipleMatchingLocalities ATTRIBUTE ::= {
```

```
  WITH SYNTAX               MultipleMatchingLocalities
  ID                        id-not-multipleMatchingLocalities }

MultipleMatchingLocalities ::= SEQUENCE {
  matchingRuleUsed  MATCHING-RULE.&id OPTIONAL,
  attributeList     SEQUENCE OF AttributeValueAssertion,
  ... }

proposedRelaxation ATTRIBUTE ::= {
  WITH SYNTAX               MRMappings
  ID                        id-not-proposedRelaxation }

MRMappings ::= SEQUENCE OF MRMapping

appliedRelaxation ATTRIBUTE ::= {
  WITH SYNTAX               OBJECT IDENTIFIER
  EQUALITY MATCHING RULE    objectIdentifierMatch
  ID                        id-not-appliedRelaxation }

pwdResponseValue ATTRIBUTE ::= {
  WITH SYNTAX               PwdResponse
  ID                        id-not-pwdResponse }

PwdResponse ::= SEQUENCE {
  warning CHOICE {
    timeleft        [0] INTEGER(0..MAX),
    graceRemaining  [1] INTEGER(0..MAX),
    ... } OPTIONAL,
  error ENUMERATED {
    passwordExpired  (0),
    changeAfterReset (1),
    ... } OPTIONAL}

ldapDiagnosticMsg ATTRIBUTE ::= {
  WITH SYNTAX               UTF8String
  SINGLE VALUE              TRUE
  ID                        id-not-ldapDiagnosticMsg }

-- LDAP defined attribute types

uid ATTRIBUTE ::= {
  WITH SYNTAX               UnboundedDirectoryString
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"uid"}
  ID                        id-coat-uid }

dc ATTRIBUTE ::= {
  WITH SYNTAX               IA5String
  EQUALITY MATCHING RULE    caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  LDAP-SYNTAX               ia5String.&id
  LDAP-NAME                 {"dc"}
  ID                        id-coat-dc }

-- Matching rules

caseExactMatch MATCHING-RULE ::= {
  SYNTAX       UnboundedDirectoryString
  LDAP-SYNTAX  directoryString.&id
  LDAP-NAME    {"caseExactMatch"}
  ID           id-mr-caseExactMatch }

caseIgnoreMatch MATCHING-RULE ::= {
  SYNTAX       UnboundedDirectoryString
  LDAP-SYNTAX  directoryString.&id
  LDAP-NAME    {"caseIgnoreMatch"}
  ID           id-mr-caseIgnoreMatch }

caseExactOrderingMatch MATCHING-RULE ::= {
```

```
  SYNTAX        UnboundedDirectoryString
  LDAP-SYNTAX   directoryString.&id
  LDAP-NAME     {"caseExactOrderingMatch"}
  ID            id-mr-caseExactOrderingMatch }

caseIgnoreOrderingMatch MATCHING-RULE ::= {
  SYNTAX        UnboundedDirectoryString
  LDAP-SYNTAX   directoryString.&id
  LDAP-NAME     {"caseIgnoreOrderingMatch"}
  ID            id-mr-caseIgnoreOrderingMatch }

caseExactSubstringsMatch MATCHING-RULE ::= {
  SYNTAX        SubstringAssertion -- only the PrintableString choice
  LDAP-SYNTAX   substringAssertion.&id
  LDAP-NAME     {"caseExactSubstringsMatch"}
  ID            id-mr-caseExactSubstringsMatch }

caseIgnoreSubstringsMatch MATCHING-RULE ::= {
  SYNTAX        SubstringAssertion
  LDAP-SYNTAX   substringAssertion.&id
  LDAP-NAME     {"caseIgnoreSubstringsMatch"}
  ID            id-mr-caseIgnoreSubstringsMatch }

SubstringAssertion ::= SEQUENCE OF CHOICE {
  initial   [0]  UnboundedDirectoryString,
  any       [1]  UnboundedDirectoryString,
  final     [2]  UnboundedDirectoryString,
    -- at most one initial and one final component
  control        Attribute{{SupportedAttributes}},
    -- Used to specify interpretation of the following items
  ... }

numericStringMatch MATCHING-RULE ::= {
  SYNTAX        NumericString
  LDAP-SYNTAX   numericString.&id
  LDAP-NAME     {"numericStringMatch"}
  ID            id-mr-numericStringMatch }

numericStringOrderingMatch MATCHING-RULE ::= {
  SYNTAX        NumericString
  LDAP-SYNTAX   numericString.&id
  LDAP-NAME     {"numericStringOrderingMatch"}
  ID            id-mr-numericStringOrderingMatch }

numericStringSubstringsMatch MATCHING-RULE ::= {
  SYNTAX        SubstringAssertion
  LDAP-SYNTAX   substringAssertion.&id
  LDAP-NAME     {"numericStringSubstringsMatch"}
  ID            id-mr-numericStringSubstringsMatch }

caseIgnoreListMatch MATCHING-RULE ::= {
  SYNTAX        CaseIgnoreList
  LDAP-SYNTAX   postalAddr.&id
  LDAP-NAME     {"caseIgnoreListMatch"}
  ID            id-mr-caseIgnoreListMatch }

CaseIgnoreList ::= SEQUENCE OF UnboundedDirectoryString

caseIgnoreListSubstringsMatch MATCHING-RULE ::= {
  SYNTAX        SubstringAssertion
  LDAP-SYNTAX   substringAssertion.&id
  LDAP-NAME     {"caseIgnoreListSubstringsMatch"}
  ID            id-mr-caseIgnoreListSubstringsMatch }

storedPrefixMatch MATCHING-RULE ::= {
  SYNTAX        UnboundedDirectoryString
  ID            id-mr-storedPrefixMatch }

booleanMatch MATCHING-RULE ::= {
  SYNTAX        BOOLEAN
  LDAP-SYNTAX   bitString.&id
```

```
  LDAP-NAME     {"booleanMatch"}
  ID            id-mr-booleanMatch }

integerMatch MATCHING-RULE ::= {
  SYNTAX        INTEGER
  LDAP-SYNTAX   integer.&id
  LDAP-NAME     {"integerMatch"}
  ID            id-mr-integerMatch }

integerOrderingMatch MATCHING-RULE ::= {
  SYNTAX        INTEGER
  LDAP-SYNTAX   integer.&id
  LDAP-NAME     {"integerOrderingMatch"}
  ID            id-mr-integerOrderingMatch }

bitStringMatch MATCHING-RULE ::= {
  SYNTAX        BIT STRING
  LDAP-SYNTAX   bitString.&id
  LDAP-NAME     {"bitStringMatch"}
  ID            id-mr-bitStringMatch }

octetStringMatch MATCHING-RULE ::= {
  SYNTAX        OCTET STRING
  LDAP-SYNTAX   octetString.&id
  LDAP-NAME     {"octetStringMatch"}
  ID            id-mr-octetStringMatch }

octetStringOrderingMatch MATCHING-RULE ::= {
  SYNTAX        OCTET STRING
  LDAP-SYNTAX   octetString.&id
  LDAP-NAME     {"octetStringOrderingMatch"}
  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        TelephoneNumber
  LDAP-SYNTAX   telephoneNr.&id
  LDAP-NAME     {"telephoneNumberMatch"}
  ID            id-mr-telephoneNumberMatch }

telephoneNumberSubstringsMatch MATCHING-RULE ::= {
  SYNTAX        SubstringAssertion
  LDAP-SYNTAX   substringAssertion.&id
  LDAP-NAME     {"telephoneNumberSubstringsMatch"}
  ID            id-mr-telephoneNumberSubstringsMatch }

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

uniqueMemberMatch MATCHING-RULE ::= {
  SYNTAX        NameAndOptionalUID
  LDAP-SYNTAX   nameAndOptionalUID.&id
  LDAP-NAME     {"uniqueMemberMatch"}
  ID            id-mr-uniqueMemberMatch }

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

facsimileNumberMatch MATCHING-RULE ::= {
  SYNTAX        TelephoneNumber
```

```
  ID            id-mr-facsimileNumberMatch }

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

uUIDPairMatch MATCHING-RULE ::= {
  SYNTAX        UUIDPair
  ID            id-mr-uuidpairmatch }

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

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

generalizedTimeMatch MATCHING-RULE ::= {
  SYNTAX        GeneralizedTime
  -- as per 46.3 b) or c) of Rec. ITU-T X.680 | ISO/IEC 8824-1
  LDAP-SYNTAX   generalizedTime.&id
  LDAP-NAME     {"generalizedTimeMatch"}
  ID            id-mr-generalizedTimeMatch }

generalizedTimeOrderingMatch MATCHING-RULE ::= {
  SYNTAX        GeneralizedTime
  -- as per 46.3 b) or c) of Rec. ITU-T X.680 | ISO/IEC 8824-1
  LDAP-SYNTAX   generalizedTime.&id
  LDAP-NAME     {"generalizedTimeOrderingMatch"}
  ID            id-mr-generalizedTimeOrderingMatch }

systemProposedMatch MATCHING-RULE ::= {
  ID  id-mr-systemProposedMatch }

integerFirstComponentMatch MATCHING-RULE ::= {
  SYNTAX        INTEGER
  LDAP-SYNTAX   integer.&id
  LDAP-NAME     {"integerFirstComponentMatch"}
  ID            id-mr-integerFirstComponentMatch }

objectIdentifierFirstComponentMatch MATCHING-RULE ::= {
  SYNTAX        OBJECT IDENTIFIER
  LDAP-SYNTAX   oid.&id
  LDAP-NAME     {"objectIdentifierFirstComponentMatch"}
  ID            id-mr-objectIdentifierFirstComponentMatch }

directoryStringFirstComponentMatch MATCHING-RULE ::= {
  SYNTAX        UnboundedDirectoryString
  LDAP-SYNTAX   directoryString.&id
  LDAP-NAME     {"directoryStringFirstComponentMatch"}
  ID            id-mr-directoryStringFirstComponentMatch }

wordMatch MATCHING-RULE ::= {
  SYNTAX        UnboundedDirectoryString
  LDAP-SYNTAX   directoryString.&id
  LDAP-NAME     {"wordMatch"}
  ID            id-mr-wordMatch }

keywordMatch MATCHING-RULE ::= {
  SYNTAX        UnboundedDirectoryString
  LDAP-SYNTAX   directoryString.&id
  LDAP-NAME     {"keywordMatch"}
  ID            id-mr-keywordMatch }

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

sequenceMatchType ATTRIBUTE ::= {
  WITH SYNTAX   SequenceMatchType
```

```
  SINGLE VALUE  TRUE
  ID            id-cat-sequenceMatchType } -- defaulting to sequenceExact

SequenceMatchType ::= ENUMERATED {
  sequenceExact                  (0),
  sequenceDeletion               (1),
  sequenceRestrictedDeletion     (2),
  sequencePermutation            (3),
  sequencePermutationAndDeletion (4),
  sequenceProviderDefined        (5),
  ... }

wordMatchTypes ATTRIBUTE ::= {
  WITH SYNTAX   WordMatchTypes
  SINGLE VALUE  TRUE
  ID            id-cat-wordMatchType } -- defaulting to wordExact

WordMatchTypes ::= ENUMERATED {
  wordExact           (0),
  wordTruncated       (1),
  wordPhonetic        (2),
  wordProviderDefined (3),
  ... }

characterMatchTypes ATTRIBUTE ::= {
  WITH SYNTAX   CharacterMatchTypes
  SINGLE VALUE  TRUE
  ID            id-cat-characterMatchTypes }

CharacterMatchTypes ::= ENUMERATED {
  characterExact      (0),
  characterCaseIgnore (1),
  characterMapped     (2),
  ... }

selectedContexts ATTRIBUTE ::= {
  WITH SYNTAX  ContextAssertion
  ID           id-cat-selectedContexts }

approximateStringMatch MATCHING-RULE ::= {
  ID      id-mr-approximateStringMatch }

ignoreIfAbsentMatch MATCHING-RULE ::= {
  ID      id-mr-ignoreIfAbsentMatch }

nullMatch MATCHING-RULE ::= {
  ID      id-mr-nullMatch }

ZONAL-MATCHING ::=
  MAPPING-BASED-MATCHING{ZonalSelect, TRUE, ZonalResult, zonalMatch.&id}

ZonalSelect ::= SEQUENCE OF AttributeType

ZonalResult ::= ENUMERATED {
  cannot-select-mapping (0),
  zero-mappings         (2),
  multiple-mappings     (3),
   ... }

zonalMatch MATCHING-RULE ::= {
  UNIQUE-MATCH-INDICATOR  multipleMatchingLocalities
  ID                      id-mr-zonalMatch }

uriMatch MATCHING-RULE ::= {
  SYNTAX       UTF8String
  LDAP-SYNTAX  directoryString.&id
  LDAP-NAME    {"uriMatch"}
  ID           id-mr-uriMatch }

-- LDAP defined matching rules
```

```
caseExactIA5Match MATCHING-RULE ::= {
  SYNTAX        IA5String
  LDAP-SYNTAX   ia5String.&id
  LDAP-NAME     {"caseExactIA5Match"}
  ID            id-lmr-caseExactIA5Match }

caseIgnoreIA5Match MATCHING-RULE ::= {
  SYNTAX        IA5String
  LDAP-SYNTAX   ia5String.&id
  LDAP-NAME     {"caseIgnoreIA5Match"}
  ID            id-lmr-caseIgnoreIA5Match }

caseIgnoreIA5SubstringsMatch MATCHING-RULE ::= {
  SYNTAX        SubstringAssertion
  LDAP-SYNTAX   substringAssertion.&id
  LDAP-NAME     {"caseIgnoreIA5SubstringsMatch"}
  ID            id-lmr-caseIgnoreIA5Match }

-- Syntaxes defined by this Directory Specification

utmCoords SYNTAX-NAME ::= {
  LDAP-DESC          "UTM Coordinates"
  DIRECTORY SYNTAX   UtmCoordinates
  ID                 id-asx-utmCoords }

uiiForm SYNTAX-NAME ::= {
  LDAP-DESC          "UII Format"
  DIRECTORY SYNTAX   UiiFormat
  ID                 id-asx-uiiForm }

epcForm SYNTAX-NAME ::= {
  LDAP-DESC          "EPC Format"
  DIRECTORY SYNTAX   EpcFormat
  ID                 id-asx-epcForm }

-- Syntaxes defined under the ldap-syntax OID arc

attributeTypeDescription SYNTAX-NAME ::= {
  LDAP-DESC          "Attribute Type Description"
  DIRECTORY SYNTAX   AttributeTypeDescription
  ID                 id-lsx-attributeTypeDescription }

bitString SYNTAX-NAME ::= {
  LDAP-DESC          "Bit String"
  DIRECTORY SYNTAX   BIT STRING
  ID                 id-lsx-bitString }

boolean SYNTAX-NAME ::= {
  LDAP-DESC          "Boolean"
  DIRECTORY SYNTAX   BOOLEAN
  ID                 id-lsx-boolean }

countryString SYNTAX-NAME ::= {
  LDAP-DESC          "Country String"
  DIRECTORY SYNTAX   CountryName
  ID                 id-lsx-countryString }

dn SYNTAX-NAME ::= {
  LDAP-DESC          "DN"
  DIRECTORY SYNTAX   DistinguishedName
  ID                 id-lsx-dn }

deliveryMethod SYNTAX-NAME ::= {
  LDAP-DESC          "Delevery Method"
  DIRECTORY SYNTAX   PreferredDeliveryMethod
  ID                 id-lsx-deliveryMethod }

directoryString SYNTAX-NAME ::= {
  LDAP-DESC          "Directory String"
  DIRECTORY SYNTAX   UnboundedDirectoryString
  ID                 id-lsx-directoryString }
```

```
dITContentRuleDescription SYNTAX-NAME ::= {
  LDAP-DESC          "DIT Content Rule Description"
  DIRECTORY SYNTAX   DITContentRuleDescription
  ID                 id-lsx-dITContentRuleDescription }

dITStructureRuleDescription SYNTAX-NAME ::= {
  LDAP-DESC          "DIT StructureRule Description"
  DIRECTORY SYNTAX   DITStructureRuleDescription
  ID                 id-lsx-dITStructureRuleDescription }

enhancedGuide SYNTAX-NAME ::= {
  LDAP-DESC          "Enhanced Guide"
  DIRECTORY SYNTAX   EnhancedGuide
  ID                 id-lsx-enhancedGuide }

facsimileTelephoneNr SYNTAX-NAME ::= {
  LDAP-DESC          "Facsimile Telephone Number"
  DIRECTORY SYNTAX   FacsimileTelephoneNumber
  ID                 id-lsx-facsimileTelephoneNr }

fax SYNTAX-NAME ::= {
  LDAP-DESC          "Fax"
  DIRECTORY SYNTAX   NULL
  ID                 id-lsx-fax }

generalizedTime SYNTAX-NAME ::= {
  LDAP-DESC          "Generalized Time"
  DIRECTORY SYNTAX   GeneralizedTime
  ID                 id-lsx-generalizedTime }

guide SYNTAX-NAME ::= {
  LDAP-DESC          "Guide"
  DIRECTORY SYNTAX   Guide
  ID                 id-lsx-guide }

ia5String SYNTAX-NAME ::= {
  LDAP-DESC          "IA5 String"
  DIRECTORY SYNTAX   IA5String
  ID                 id-lsx-ia5String }

integer SYNTAX-NAME ::= {
  LDAP-DESC          "INTEGER"
  DIRECTORY SYNTAX   INTEGER
  ID                 id-lsx-integer }

jpeg SYNTAX-NAME ::= {
  LDAP-DESC          "JPEG"
  DIRECTORY SYNTAX   NULL
  ID                 id-lsx-jpeg }

matchingRuleDescription SYNTAX-NAME ::= {
  LDAP-DESC          "Matching Rule Description"
  DIRECTORY SYNTAX   MatchingRuleDescription
  ID                 id-lsx-matchingRuleDescription }

matchingRuleUseDescription SYNTAX-NAME ::= {
  LDAP-DESC          "Matching Rule Use Description"
  DIRECTORY SYNTAX   MatchingRuleUseDescription
  ID                 id-lsx-matchingRuleUseDescription }

nameAndOptionalUID SYNTAX-NAME ::= {
  LDAP-DESC          "Name And Optional UID"
  DIRECTORY SYNTAX   NameAndOptionalUID
  ID                 id-lsx-nameAndOptionalUID }

nameFormDescription SYNTAX-NAME ::= {
  LDAP-DESC          "Name Form Description"
  DIRECTORY SYNTAX   NameFormDescription
  ID                 id-lsx-nameFormDescription }
```

```
numericString SYNTAX-NAME ::= {
  LDAP-DESC          "Numeric String"
  DIRECTORY SYNTAX   NumericString
  ID                 id-lsx-numericString }

objectClassDescription SYNTAX-NAME ::= {
  LDAP-DESC          "Object Class Description"
  DIRECTORY SYNTAX   ObjectClassDescription
  ID                 id-lsx-objectClassDescription }

oid SYNTAX-NAME ::= {
  LDAP-DESC          "OID"
  DIRECTORY SYNTAX   OBJECT IDENTIFIER
  ID                 id-lsx-oid }

otherMailbox SYNTAX-NAME ::= {
  LDAP-DESC         "Other Mailbox"
  DIRECTORY SYNTAX   NULL
  ID                 id-lsx-otherMailbox }

octetString SYNTAX-NAME ::= {
  LDAP-DESC          "Octet String"
  DIRECTORY SYNTAX   OCTET STRING
  ID                 id-lsx-octetString }

postalAddr SYNTAX-NAME ::= {
  LDAP-DESC          "Postal Address"
  DIRECTORY SYNTAX   PostalAddress
  ID                 id-lsx-postalAddr }

printableString SYNTAX-NAME ::= {
  LDAP-DESC          "Printable String"
  DIRECTORY SYNTAX   PrintableString
  ID                 id-lsx-printableString }

subtreeSpec SYNTAX-NAME ::= {
  LDAP-DESC          "SubtreeSpecification"
  DIRECTORY SYNTAX   SubtreeSpecification
  ID                 id-lsx-subtreeSpec }

telephoneNr SYNTAX-NAME ::= {
  LDAP-DESC          "Telephone Number"
  DIRECTORY SYNTAX   TelephoneNumber
  ID                 id-lsx-telephoneNr }

telexNr SYNTAX-NAME ::= {
  LDAP-DESC          "Telex Number"
  DIRECTORY SYNTAX   TelexNumber
  ID                 id-lsx-telexNr }

utcTime SYNTAX-NAME ::= {
  LDAP-DESC          "UTC Time"
  DIRECTORY SYNTAX   UTCTime
  ID                 id-lsx-utcTime }

ldapSyntaxDescription SYNTAX-NAME ::= {
  LDAP-DESC          "LDAP Syntax Description"
  DIRECTORY SYNTAX   NULL
  ID                 id-lsx-ldapSyntaxDescription }

substringAssertion SYNTAX-NAME ::= {
  LDAP-DESC          "Substring Assertion"
  DIRECTORY SYNTAX   SubstringAssertion
  ID                 id-lsx-substringAssertion }

-- 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 SIZE (1..MAX) OF Period},
  notThisTime   BOOLEAN DEFAULT FALSE,
  timeZone      TimeZone OPTIONAL,
  ... }

Period ::= SEQUENCE {
  timesOfDay  [0]  SET SIZE (1..MAX) 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 {
  intNamedDays ENUMERATED {
```