Network Working Group
Request for Comments: 2252
Category: Standards Track

M. Wahl
Critical Angle Inc.
A. Coulbeck
Isode Inc.
T. Howes
Netscape Communications Corp.
S. Kille
Isode Limited
December 1997

Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions

1. Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

IESG Note

This document describes a directory access protocol that provides both read and update access. Update access requires secure authentication, but this document does not mandate implementation of any satisfactory authentication mechanisms.

In accordance with RFC 2026, section 4.4.1, this specification is being approved by IESG as a Proposed Standard despite this limitation, for the following reasons:

a. to encourage implementation and interoperability testing of these protocols (with or without update access) before they are deployed, and

b. to encourage deployment and use of these protocols in read-only applications. (e.g. applications where LDAPv3 is used as a query language for directories which are updated by some secure mechanism other than LDAP), and

c. to avoid delaying the advancement and deployment of other Internet standards-track protocols which require the ability to query, but not update, LDAPv3 directory servers.

Readers are hereby warned that until mandatory authentication mechanisms are standardized, clients and servers written according to this specification which make use of update functionality are UNLIKELY TO INTEROPERATE, or MAY INTEROPERATE ONLY IF AUTHENTICATION IS REDUCED TO AN UNACCEPTABLY WEAK LEVEL.

Implementors are hereby discouraged from deploying LDAPv3 clients or servers which implement the update functionality, until a Proposed Standard for mandatory authentication in LDAPv3 has been approved and published as an RFC.

2. Abstract

The Lightweight Directory Access Protocol (LDAP) [1] requires that the contents of AttributeValue fields in protocol elements be octet strings. This document defines a set of syntaxes for LDAPv3, and the rules by which attribute values of these syntaxes are represented as octet strings for transmission in the LDAP protocol. The syntaxes defined in this document are referenced by this and other documents that define attribute types. This document also defines the set of attribute types which LDAP servers should support.

3. Overview

This document defines the framework for developing schemas for directories accessible via the Lightweight Directory Access Protocol.

Schema is the collection of attribute type definitions, object class definitions and other information which a server uses to determine how to match a filter or attribute value assertion (in a compare operation) against the attributes of an entry, and whether to permit add and modify operations.

Section 4 states the general requirements and notations for attribute types, object classes, syntax and matching rule definitions.

Section 5 lists attributes, section 6 syntaxes and section 7 object classes.

Additional documents define schemas for representing real-world objects as directory entries.

4. General Issues

This document describes encodings used in an Internet protocol.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [4].

Attribute Type and Object Class definitions are written in a string representation of the AttributeTypeDescription and ObjectClassDescription data types defined in X.501(93) [3]. Implementors are strongly advised to first read the description of how schema is represented in X.500 before reading the rest of this document.

4.1. Common Encoding Aspects

For the purposes of defining the encoding rules for attribute syntaxes, the following BNF definitions will be used. They are based on the BNF styles of RFC 822 [13].

```
a     = "a" / "b" / "c" / "d" / "e" / "f" / "g" / "h" / "i" /
        "j" / "k" / "l" / "m" / "n" / "o" / "p" / "q" / "r" /
        "s" / "t" / "u" / "v" / "w" / "x" / "y" / "z" / "A" /
        "B" / "C" / "D" / "E" / "F" / "G" / "H" / "I" / "J" /
        "K" / "L" / "M" / "N" / "O" / "P" / "Q" / "R" / "S" /
        "T" / "U" / "V" / "W" / "X" / "Y" / "Z"

d               = "0" / "1" / "2" / "3" / "4" /
                  "5" / "6" / "7" / "8" / "9"

hex-digit       =  d / "a" / "b" / "c" / "d" / "e" / "f" /
                       "A" / "B" / "C" / "D" / "E" / "F"

k               = a / d / "-" / ";"

p               = a / d / "'" / "(" / ")" / "+" / "," /
                  "-" / "." / "/" / ":" / "?" / " "

letterstring    = 1*a

numericstring   = 1*d

anhstring       = 1*k

keystring       = a [ anhstring ]

printablestring = 1*p
```

```
space           = 1*" "

whsp            = [ space ]

utf8            = <any sequence of octets formed from the UTF-8 [9]
                   transformation of a character from ISO10646 [10]>

dstring         = 1*utf8

qdstring        = whsp "'" dstring "'" whsp

qdstringlist    = [ qdstring *( qdstring ) ]

qdstrings       = qdstring / ( whsp "(" qdstringlist ")" whsp )
```

In the following BNF for the string representation of OBJECT IDENTIFIERs, descr is the syntactic representation of an object descriptor, which consists of letters and digits, starting with a letter. An OBJECT IDENTIFIER in the numericoid format should not have leading zeroes (e.g. "0.9.3" is permitted but "0.09.3" should not be generated).

When encoding 'oid' elements in a value, the descr encoding option SHOULD be used in preference to the numericoid. An object descriptor is a more readable alias for a number OBJECT IDENTIFIER, and these (where assigned and known by the implementation) SHOULD be used in preference to numeric oids to the greatest extent possible. Examples of object descriptors in LDAP are attribute type, object class and matching rule names.

```
  oid             = descr / numericoid

  descr           = keystring

  numericoid      = numericstring *( "." numericstring )

  woid            = whsp oid whsp

  ; set of oids of either form
  oids            = woid / ( "(" oidlist ")" )

  oidlist         = woid *( "$" woid )

  ; object descriptors used as schema element names
  qdescrs         = qdescr / ( whsp "(" qdescrlist ")" whsp )

  qdescrlist      = [ qdescr *( qdescr ) ]
```

```
qdescr          = whsp "'" descr "'" whsp
```

4.2. Attribute Types

The attribute types are described by sample values for the subschema "attributeTypes" attribute, which is written in the AttributeTypeDescription syntax. While lines have been folded for readability, the values transferred in protocol would not contain newlines.

The AttributeTypeDescription is encoded according to the following BNF, and the productions for oid, qdescrs and qdstring are given in section 4.1. Implementors should note that future versions of this document may have expanded this BNF to include additional terms. Terms which begin with the characters "X-" are reserved for private experiments, and MUST be followed by a <qdstrings>.

```
AttributeTypeDescription = "(" whsp
      numericoid whsp              ; AttributeType identifier
    [ "NAME" qdescrs ]             ; name used in AttributeType
    [ "DESC" qdstring ]            ; description
    [ "OBSOLETE" whsp ]
    [ "SUP" woid ]                 ; derived from this other
                                   ; AttributeType
    [ "EQUALITY" woid              ; Matching Rule name
    [ "ORDERING" woid              ; Matching Rule name
    [ "SUBSTR" woid ]              ; Matching Rule name
    [ "SYNTAX" whsp noidlen whsp ] ; see section 4.3
    [ "SINGLE-VALUE" whsp ]        ; default multi-valued
    [ "COLLECTIVE" whsp ]          ; default not collective
    [ "NO-USER-MODIFICATION" whsp ]; default user modifiable
    [ "USAGE" whsp AttributeUsage ]; default userApplications
    whsp ")"

AttributeUsage =
    "userApplications"     /
    "directoryOperation"   /
    "distributedOperation" / ; DSA-shared
    "dSAOperation"          ; DSA-specific, value depends on server
```

Servers are not required to provide the same or any text in the description part of the subschema values they maintain. Servers SHOULD provide at least one of the "SUP" and "SYNTAX" fields for each AttributeTypeDescription.

Servers MUST implement all the attribute types referenced in sections 5.1, 5.2 and 5.3.

Servers MAY recognize additional names and attributes not listed in this document, and if they do so, MUST publish the definitions of the types in the attributeTypes attribute of their subschema entries.

Schema developers MUST NOT create attribute definitions whose names conflict with attributes defined for use with LDAP in existing standards-track RFCs.

An AttributeDescription can be used as the value in a NAME part of an AttributeTypeDescription. Note that these are case insensitive.

Note that the AttributeTypeDescription does not list the matching rules which can can be used with that attribute type in an extensibleMatch search filter. This is done using the matchingRuleUse attribute described in section 4.5.

This document refines the schema description of X.501 by requiring that the syntax field in an AttributeTypeDescription be a string representation of an OBJECT IDENTIFIER for the LDAP string syntax definition, and an optional indication of the maximum length of a value of this attribute (defined in section 4.3.2).

4.3. Syntaxes

This section defines general requirements for LDAP attribute value syntax encodings. All documents defining attribute syntax encodings for use with LDAP are expected to conform to these requirements.

The encoding rules defined for a given attribute syntax must produce octet strings. To the greatest extent possible, encoded octet strings should be usable in their native encoded form for display purposes. In particular, encoding rules for attribute syntaxes defining non-binary values should produce strings that can be displayed with little or no translation by clients implementing LDAP. There are a few cases (e.g. audio) however, when it is not sensible to produce a printable representation, and clients MUST NOT assume that an unrecognized syntax is a string representation.

In encodings where an arbitrary string, not a Distinguished Name, is used as part of a larger production, and other than as part of a Distinguished Name, a backslash quoting mechanism is used to escape the following separator symbol character (such as "’", "$" or "#") if it should occur in that string. The backslash is followed by a pair of hexadecimal digits representing the next character. A backslash itself in the string which forms part of a larger syntax is always transmitted as ’\5C’ or ’\5c’. An example is given in section 6.27.

Syntaxes are also defined for matching rules whose assertion value syntax is different from the attribute value syntax.

4.3.1 Binary Transfer of Values

This encoding format is used if the binary encoding is requested by the client for an attribute, or if the attribute syntax name is "1.3.6.1.4.1.1466.115.121.1.5". The contents of the LDAP AttributeValue or AssertionValue field is a BER-encoded instance of the attribute value or a matching rule assertion value ASN.1 data type as defined for use with X.500. (The first byte inside the OCTET STRING wrapper is a tag octet. However, the OCTET STRING is still encoded in primitive form.)

All servers MUST implement this form for both generating attribute values in search responses, and parsing attribute values in add, compare and modify requests, if the attribute type is recognized and the attribute syntax name is that of Binary. Clients which request that all attributes be returned from entries MUST be prepared to receive values in binary (e.g. userCertificate;binary), and SHOULD NOT simply display binary or unrecognized values to users.

4.3.2. Syntax Object Identifiers

Syntaxes for use with LDAP are named by OBJECT IDENTIFIERs, which are dotted-decimal strings. These are not intended to be displayed to users.

```
noidlen = numericoid [ "{" len "}" ]

len     = numericstring
```

The following table lists some of the syntaxes that have been defined for LDAP thus far. The H-R column suggests whether a value in that syntax would likely be a human readable string. Clients and servers need not implement all the syntaxes listed here, and MAY implement other syntaxes.

Other documents may define additional syntaxes. However, the definition of additional arbitrary syntaxes is strongly deprecated since it will hinder interoperability: today’s client and server implementations generally do not have the ability to dynamically recognize new syntaxes. In most cases attributes will be defined with the syntax for directory strings.

Value being represented	H-R	OBJECT IDENTIFIER
ACI Item	N	1.3.6.1.4.1.1466.115.121.1.1
Access Point	Y	1.3.6.1.4.1.1466.115.121.1.2
Attribute Type Description	Y	1.3.6.1.4.1.1466.115.121.1.3
Audio	N	1.3.6.1.4.1.1466.115.121.1.4
Binary	N	1.3.6.1.4.1.1466.115.121.1.5
Bit String	Y	1.3.6.1.4.1.1466.115.121.1.6
Boolean	Y	1.3.6.1.4.1.1466.115.121.1.7
Certificate	N	1.3.6.1.4.1.1466.115.121.1.8
Certificate List	N	1.3.6.1.4.1.1466.115.121.1.9
Certificate Pair	N	1.3.6.1.4.1.1466.115.121.1.10
Country String	Y	1.3.6.1.4.1.1466.115.121.1.11
DN	Y	1.3.6.1.4.1.1466.115.121.1.12
Data Quality Syntax	Y	1.3.6.1.4.1.1466.115.121.1.13
Delivery Method	Y	1.3.6.1.4.1.1466.115.121.1.14
Directory String	Y	1.3.6.1.4.1.1466.115.121.1.15
DIT Content Rule Description	Y	1.3.6.1.4.1.1466.115.121.1.16
DIT Structure Rule Description	Y	1.3.6.1.4.1.1466.115.121.1.17
DL Submit Permission	Y	1.3.6.1.4.1.1466.115.121.1.18
DSA Quality Syntax	Y	1.3.6.1.4.1.1466.115.121.1.19
DSE Type	Y	1.3.6.1.4.1.1466.115.121.1.20
Enhanced Guide	Y	1.3.6.1.4.1.1466.115.121.1.21
Facsimile Telephone Number	Y	1.3.6.1.4.1.1466.115.121.1.22
Fax	N	1.3.6.1.4.1.1466.115.121.1.23
Generalized Time	Y	1.3.6.1.4.1.1466.115.121.1.24
Guide	Y	1.3.6.1.4.1.1466.115.121.1.25
IA5 String	Y	1.3.6.1.4.1.1466.115.121.1.26
INTEGER	Y	1.3.6.1.4.1.1466.115.121.1.27
JPEG	N	1.3.6.1.4.1.1466.115.121.1.28
LDAP Syntax Description	Y	1.3.6.1.4.1.1466.115.121.1.54
LDAP Schema Definition	Y	1.3.6.1.4.1.1466.115.121.1.56
LDAP Schema Description	Y	1.3.6.1.4.1.1466.115.121.1.57
Master And Shadow Access Points	Y	1.3.6.1.4.1.1466.115.121.1.29
Matching Rule Description	Y	1.3.6.1.4.1.1466.115.121.1.30
Matching Rule Use Description	Y	1.3.6.1.4.1.1466.115.121.1.31
Mail Preference	Y	1.3.6.1.4.1.1466.115.121.1.32
MHS OR Address	Y	1.3.6.1.4.1.1466.115.121.1.33
Modify Rights	Y	1.3.6.1.4.1.1466.115.121.1.55
Name And Optional UID	Y	1.3.6.1.4.1.1466.115.121.1.34
Name Form Description	Y	1.3.6.1.4.1.1466.115.121.1.35
Numeric String	Y	1.3.6.1.4.1.1466.115.121.1.36
Object Class Description	Y	1.3.6.1.4.1.1466.115.121.1.37
Octet String	Y	1.3.6.1.4.1.1466.115.121.1.40
OID	Y	1.3.6.1.4.1.1466.115.121.1.38
Other Mailbox	Y	1.3.6.1.4.1.1466.115.121.1.39
Postal Address	Y	1.3.6.1.4.1.1466.115.121.1.41
Protocol Information	Y	1.3.6.1.4.1.1466.115.121.1.42

Presentation Address	Y	1.3.6.1.4.1.1466.115.121.1.43
Printable String	Y	1.3.6.1.4.1.1466.115.121.1.44
Substring Assertion	Y	1.3.6.1.4.1.1466.115.121.1.58
Subtree Specification	Y	1.3.6.1.4.1.1466.115.121.1.45
Supplier Information	Y	1.3.6.1.4.1.1466.115.121.1.46
Supplier Or Consumer	Y	1.3.6.1.4.1.1466.115.121.1.47
Supplier And Consumer	Y	1.3.6.1.4.1.1466.115.121.1.48
Supported Algorithm	N	1.3.6.1.4.1.1466.115.121.1.49
Telephone Number	Y	1.3.6.1.4.1.1466.115.121.1.50
Teletex Terminal Identifier	Y	1.3.6.1.4.1.1466.115.121.1.51
Telex Number	Y	1.3.6.1.4.1.1466.115.121.1.52
UTC Time	Y	1.3.6.1.4.1.1466.115.121.1.53

A suggested minimum upper bound on the number of characters in value with a string-based syntax, or the number of bytes in a value for all other syntaxes, may be indicated by appending this bound count inside of curly braces following the syntax name's OBJECT IDENTIFIER in an Attribute Type Description. This bound is not part of the syntax name itself. For instance, "1.3.6.4.1.1466.0{64}" suggests that server implementations should allow a string to be 64 characters long, although they may allow longer strings. Note that a single character of the Directory String syntax may be encoded in more than one byte since UTF-8 is a variable-length encoding.

4.3.3. Syntax Description

The following BNF may be used to associate a short description with a syntax OBJECT IDENTIFIER. Implementors should note that future versions of this document may expand this definition to include additional terms. Terms whose identifier begins with "X-" are reserved for private experiments, and MUST be followed by a <qdstrings>.

```
SyntaxDescription = "(" whsp
    numericoid whsp
    [ "DESC" qdstring ]
    whsp ")"
```

4.4. Object Classes

The format for representation of object classes is defined in X.501 [3]. In general every entry will contain an abstract class ("top" or "alias"), at least one structural object class, and zero or more auxiliary object classes. Whether an object class is abstract, structural or auxiliary is defined when the object class identifier is assigned. An object class definition should not be changed without having a new identifier assigned to it.

Object class descriptions are written according to the following BNF. Implementors should note that future versions of this document may expand this definition to include additional terms. Terms whose identifier begins with "X-" are reserved for private experiments, and MUST be followed by a <qdstrings> encoding.

```
ObjectClassDescription = "(" whsp
    numericoid whsp      ; ObjectClass identifier
    [ "NAME" qdescrs ]
    [ "DESC" qdstring ]
    [ "OBSOLETE" whsp ]
    [ "SUP" oids ]       ; Superior ObjectClasses
    [ ( "ABSTRACT" / "STRUCTURAL" / "AUXILIARY" ) whsp ]
                         ; default structural
    [ "MUST" oids ]      ; AttributeTypes
    [ "MAY" oids ]       ; AttributeTypes
whsp ")"
```

These are described as sample values for the subschema "objectClasses" attribute for a server which implements the LDAP schema. While lines have been folded for readability, the values transferred in protocol would not contain newlines.

Servers SHOULD implement all the object classes referenced in section 7, except for extensibleObject, which is optional. Servers MAY implement additional object classes not listed in this document, and if they do so, MUST publish the definitions of the classes in the objectClasses attribute of their subschema entries.

Schema developers MUST NOT create object class definitions whose names conflict with attributes defined for use with LDAP in existing standards-track RFCs.

4.5. Matching Rules

Matching rules are used by servers to compare attribute values against assertion values when performing Search and Compare operations. They are also used to identify the value to be added or deleted when modifying entries, and are used when comparing a purported distinguished name with the name of an entry.

Most of the attributes given in this document will have an equality matching rule defined.

Matching rule descriptions are written according to the following BNF. Implementors should note that future versions of this document may have expanded this BNF to include additional terms. Terms whose identifier begins with "X-" are reserved for private experiments, and

MUST be followed by a <qdstrings> encoding.

```
MatchingRuleDescription = "(" whsp
    numericoid whsp  ; MatchingRule identifier
    [ "NAME" qdescrs ]
    [ "DESC" qdstring ]
    [ "OBSOLETE" whsp ]
    "SYNTAX" numericoid
whsp ")"
```

Values of the matchingRuleUse list the attributes which are suitable for use with an extensible matching rule.

```
MatchingRuleUseDescription = "(" whsp
    numericoid whsp  ; MatchingRule identifier
    [ "NAME" qdescrs ]
    [ "DESC" qdstring ]
    [ "OBSOLETE" ]
   "APPLIES" oids    ; AttributeType identifiers
whsp ")"
```

Servers which support matching rules and the extensibleMatch SHOULD implement all the matching rules in section 8.

Servers MAY implement additional matching rules not listed in this document, and if they do so, MUST publish the definitions of the matching rules in the matchingRules attribute of their subschema entries. If the server supports the extensibleMatch, then the server MUST publish the relationship between the matching rules and attributes in the matchingRuleUse attribute.

For example, a server which implements a privately-defined matching rule for performing sound-alike matches on Directory String-valued attributes would include the following in the subschema entry (1.2.3.4.5 is an example, the OID of an actual matching rule would be different):

```
matchingRule: ( 1.2.3.4.5 NAME 'soundAlikeMatch'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

If this matching rule could be used with the attributes 2.5.4.41 and 2.5.4.15, the following would also be present:

```
matchingRuleUse: ( 1.2.3.4.5 APPLIES (2.5.4.41 $ 2.5.4.15) )
```

A client could then make use of this matching rule by sending a search operation in which the filter is of the extensibleMatch choice, the matchingRule field is "soundAlikeMatch", and the type field is "2.5.4.41" or "2.5.4.15".

5. Attribute Types

All LDAP server implementations MUST recognize the attribute types defined in this section.

Servers SHOULD also recognize all the attributes from section 5 of [12].

5.1. Standard Operational Attributes

Servers MUST maintain values of these attributes in accordance with the definitions in X.501(93).

5.1.1. createTimestamp

This attribute SHOULD appear in entries which were created using the Add operation.

```
( 2.5.18.1 NAME 'createTimestamp' EQUALITY generalizedTimeMatch
  ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
  SINGLE-VALUE NO-USER-MODIFICATION USAGE directoryOperation )
```

5.1.2. modifyTimestamp

This attribute SHOULD appear in entries which have been modified using the Modify operation.

```
( 2.5.18.2 NAME 'modifyTimestamp' EQUALITY generalizedTimeMatch
  ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
  SINGLE-VALUE NO-USER-MODIFICATION USAGE directoryOperation )
```

5.1.3. creatorsName

This attribute SHOULD appear in entries which were created using the Add operation.

```
( 2.5.18.3 NAME 'creatorsName' EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  SINGLE-VALUE NO-USER-MODIFICATION USAGE directoryOperation )
```

5.1.4. modifiersName

This attribute SHOULD appear in entries which have been modified using the Modify operation.

```
( 2.5.18.4 NAME 'modifiersName' EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  SINGLE-VALUE NO-USER-MODIFICATION USAGE directoryOperation )
```

5.1.5. subschemaSubentry

The value of this attribute is the name of a subschema entry (or subentry if the server is based on X.500(93)) in which the server makes available attributes specifying the schema.

```
( 2.5.18.10 NAME 'subschemaSubentry'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 NO-USER-MODIFICATION
  SINGLE-VALUE USAGE directoryOperation )
```

5.1.6. attributeTypes

This attribute is typically located in the subschema entry.

```
( 2.5.21.5 NAME 'attributeTypes'
  EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.3 USAGE directoryOperation )
```

5.1.7. objectClasses

This attribute is typically located in the subschema entry.

```
( 2.5.21.6 NAME 'objectClasses'
  EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.37 USAGE directoryOperation )
```

5.1.8. matchingRules

This attribute is typically located in the subschema entry.

```
( 2.5.21.4 NAME 'matchingRules'
  EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.30 USAGE directoryOperation )
```

5.1.9. matchingRuleUse

This attribute is typically located in the subschema entry.

```
( 2.5.21.8 NAME 'matchingRuleUse'
  EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.31 USAGE directoryOperation )
```

5.2. LDAP Operational Attributes

These attributes are only present in the root DSE (see [1] and [3]).

Servers MUST recognize these attribute names, but it is not required that a server provide values for these attributes, when the attribute corresponds to a feature which the server does not implement.

5.2.1. namingContexts

The values of this attribute correspond to naming contexts which this server masters or shadows. If the server does not master any information (e.g. it is an LDAP gateway to a public X.500 directory) this attribute will be absent. If the server believes it contains the entire directory, the attribute will have a single value, and that value will be the empty string (indicating the null DN of the root). This attribute will allow a client to choose suitable base objects for searching when it has contacted a server.

```
( 1.3.6.1.4.1.1466.101.120.5 NAME 'namingContexts'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE dSAOperation )
```

5.2.2. altServer

The values of this attribute are URLs of other servers which may be contacted when this server becomes unavailable. If the server does not know of any other servers which could be used this attribute will be absent. Clients may cache this information in case their preferred LDAP server later becomes unavailable.

```
( 1.3.6.1.4.1.1466.101.120.6 NAME 'altServer'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE dSAOperation )
```

5.2.3. supportedExtension

The values of this attribute are OBJECT IDENTIFIERs identifying the supported extended operations which the server supports.

If the server does not support any extensions this attribute will be absent.

```
    ( 1.3.6.1.4.1.1466.101.120.7 NAME 'supportedExtension'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 USAGE dSAOperation )
```

5.2.4. supportedControl

The values of this attribute are the OBJECT IDENTIFIERs identifying controls which the server supports. If the server does not support any controls, this attribute will be absent.

```
    ( 1.3.6.1.4.1.1466.101.120.13 NAME 'supportedControl'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 USAGE dSAOperation )
```

5.2.5. supportedSASLMechanisms

The values of this attribute are the names of supported SASL mechanisms which the server supports. If the server does not support any mechanisms this attribute will be absent.

```
    ( 1.3.6.1.4.1.1466.101.120.14 NAME 'supportedSASLMechanisms'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE dSAOperation )
```

5.2.6. supportedLDAPVersion

The values of this attribute are the versions of the LDAP protocol which the server implements.

```
    ( 1.3.6.1.4.1.1466.101.120.15 NAME 'supportedLDAPVersion'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE dSAOperation )
```

5.3. LDAP Subschema Attribute

This attribute is typically located in the subschema entry.

5.3.1. ldapSyntaxes

Servers MAY use this attribute to list the syntaxes which are implemented. Each value corresponds to one syntax.

```
    ( 1.3.6.1.4.1.1466.101.120.16 NAME 'ldapSyntaxes'
      EQUALITY objectIdentifierFirstComponentMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.54 USAGE directoryOperation )
```

5.4. X.500 Subschema attributes

These attributes are located in the subschema entry. All servers SHOULD recognize their name, although typically only X.500 servers will implement their functionality.

5.4.1. dITStructureRules

```
 ( 2.5.21.1 NAME 'dITStructureRules' EQUALITY integerFirstComponentMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.17 USAGE directoryOperation )
```

5.4.2. nameForms

```
    ( 2.5.21.7 NAME 'nameForms'
      EQUALITY objectIdentifierFirstComponentMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.35 USAGE directoryOperation )
```

5.4.3. ditContentRules

```
    ( 2.5.21.2 NAME 'dITContentRules'
      EQUALITY objectIdentifierFirstComponentMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.16 USAGE directoryOperation )
```

6. Syntaxes

Servers SHOULD recognize all the syntaxes described in this section.

6.1. Attribute Type Description

(1.3.6.1.4.1.1466.115.121.1.3 DESC 'Attribute Type Description')

Values in this syntax are encoded according to the BNF given at the start of section 4.2. For example,

```
     ( 2.5.4.0 NAME 'objectClass'
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
```

6.2. Binary

(1.3.6.1.4.1.1466.115.121.1.5 DESC 'Binary')

Values in this syntax are encoded as described in section 4.3.1.

6.3. Bit String

(1.3.6.1.4.1.1466.115.121.1.6 DESC 'Bit String')

Values in this syntax are encoded according to the following BNF:

```
   bitstring = "'" *binary-digit "'B"

   binary-digit = "0" / "1"
```

Example:

'0101111101'B

6.4. Boolean

(1.3.6.1.4.1.1466.115.121.1.7 DESC 'Boolean')

Values in this syntax are encoded according to the following BNF:

boolean = "TRUE" / "FALSE"

Boolean values have an encoding of "TRUE" if they are logically true, and have an encoding of "FALSE" otherwise.

6.5. Certificate

(1.3.6.1.4.1.1466.115.121.1.8 DESC 'Certificate')

Because of the changes from X.509(1988) and X.509(1993) and additional changes to the ASN.1 definition to support certificate extensions, no string representation is defined, and values in this syntax MUST only be transferred using the binary encoding, by requesting or returning the attributes with descriptions "userCertificate;binary" or "caCertificate;binary". The BNF notation in RFC 1778 for "User Certificate" is not recommended to be used.

6.6. Certificate List

(1.3.6.1.4.1.1466.115.121.1.9 DESC 'Certificate List')

Because of the incompatibility of the X.509(1988) and X.509(1993) definitions of revocation lists, values in this syntax MUST only be transferred using a binary encoding, by requesting or returning the attributes with descriptions "certificateRevocationList;binary" or "authorityRevocationList;binary". The BNF notation in RFC 1778 for "Authority Revocation List" is not recommended to be used.

6.7. Certificate Pair

(1.3.6.1.4.1.1466.115.121.1.10 DESC 'Certificate Pair')

Because the Certificate is being carried in binary, values in this syntax MUST only be transferred using a binary encoding, by requesting or returning the attribute description "crossCertificatePair;binary". The BNF notation in RFC 1778 for "Certificate Pair" is not recommended to be used.

6.8. Country String

(1.3.6.1.4.1.1466.115.121.1.11 DESC 'Country String')

A value in this syntax is encoded the same as a value of Directory String syntax. Note that this syntax is limited to values of exactly two printable string characters, as listed in ISO 3166 [14].

```
CountryString  = p p
```

Example:

```
US
```

6.9. DN

(1.3.6.1.4.1.1466.115.121.1.12 DESC 'DN')

Values in the Distinguished Name syntax are encoded to have the representation defined in [5]. Note that this representation is not reversible to an ASN.1 encoding used in X.500 for Distinguished Names, as the CHOICE of any DirectoryString element in an RDN is no longer known.

Examples (from [5]):

```
CN=Steve Kille,O=Isode Limited,C=GB
OU=Sales+CN=J. Smith,O=Widget Inc.,C=US
CN=L. Eagle,O=Sue\, Grabbit and Runn,C=GB
CN=Before\0DAfter,O=Test,C=GB
1.3.6.1.4.1.1466.0=#04024869,O=Test,C=GB
SN=Lu\C4\8Di\C4\87
```

6.10. Directory String

(1.3.6.1.4.1.1466.115.121.1.15 DESC 'Directory String')

A string in this syntax is encoded in the UTF-8 form of ISO 10646 (a superset of Unicode). Servers and clients MUST be prepared to receive encodings of arbitrary Unicode characters, including characters not presently assigned to any character set.

For characters in the PrintableString form, the value is encoded as the string value itself.

If it is of the TeletexString form, then the characters are transliterated to their equivalents in UniversalString, and encoded in UTF-8 [9].

If it is of the UniversalString or BMPString forms [10], UTF-8 is used to encode them.

Note: the form of DirectoryString is not indicated in protocol unless the attribute value is carried in binary. Servers which convert to DAP MUST choose an appropriate form. Servers MUST NOT reject values merely because they contain legal Unicode characters outside of the range of printable ASCII.

Example:

This is a string of DirectoryString containing #!%#@

6.11. DIT Content Rule Description

ues in this syntax are encoded according to the following BNF.
lementors should note that future versions of this document
have expanded this BNF to include additional terms.

0

```
DITContentRuleDescription = "("
    numericoid   ; Structural ObjectClass identifier
    [ "NAME" qdescrs ]
    [ "DESC" qdstring ]
    [ "OBSOLETE" ]
    [ "AUX" oids ]    ; Auxiliary ObjectClasses
    [ "MUST" oids ]   ; AttributeType identifiers
    [ "MAY" oids ]    ; AttributeType identifiers
    [ "NOT" oids ]    ; AttributeType identifiers
   ")"
```

0 2. Facsimile Telephone Number

3

(1.3.6.1.4.1.1466.115.121.1.22 DESC 'Facsimile Telephone Number')

Values in this syntax are encoded according to the following BNF:

```
fax-number    = printablestring [ "$" faxparameters ]

faxparameters = faxparm / ( faxparm "$" faxparameters )

faxparm = "twoDimensional" / "fineResolution" /
          "unlimitedLength" /
          "b4Length" / "a3Width" / "b4Width" / "uncompressed"
```

In the above, the first printablestring is the telephone number, based on E.123 [15], and the faxparm tokens represent fax parameters.

6.13. Fax

(1.3.6.1.4.1.1466.115.121.1.23 DESC 'Fax')

Values in this syntax are encoded as if they were octet strings containing Group 3 Fax images as defined in [7].

6.14. Generalized Time

(1.3.6.1.4.1.1466.115.121.1.24 DESC 'Generalized Time')

Values in this syntax are encoded as printable strings, represented as specified in X.208. Note that the time zone must be specified. It is strongly recommended that GMT time be used. For example,

199412161032Z

6.15. IA5 String

(1.3.6.1.4.1.1466.115.121.1.26 DESC 'IA5 String')

The encoding of a value in this syntax is the string value itself.

6.16. INTEGER

(1.3.6.1.4.1.1466.115.121.1.27 DESC 'INTEGER')

Values in this syntax are encoded as the decimal representation of their values, with each decimal digit represented by the its character equivalent. So the number 1321 is represented by the character string "1321".

6.17. JPEG

(1.3.6.1.4.1.1466.115.121.1.28 DESC 'JPEG')

Values in this syntax are encoded as strings containing JPEG images in the JPEG File Interchange Format (JFIF), as described in [8].

6.18. Matching Rule Description

(1.3.6.1.4.1.1466.115.121.1.30 DESC 'Matching Rule Description')

Values of type matchingRules are encoded as strings according to the BNF given in section 4.5.

6.19. Matching Rule Use Description

(1.3.6.1.4.1.1466.115.121.1.31 DESC 'Matching Rule Use Description')

Values of type matchingRuleUse are encoded as strings according to the BNF given in section 4.5.

6.20. MHS OR Address

(1.3.6.1.4.1.1466.115.121.1.33 DESC 'MHS OR Address')

Values in this syntax are encoded as strings, according to the format defined in [11].

6.21. Name And Optional UID

(1.3.6.1.4.1.1466.115.121.1.34 DESC 'Name And Optional UID')

Values in this syntax are encoded according to the following BNF:

```
NameAndOptionalUID = DistinguishedName [ "#" bitstring ]
```

Although the '#' character may occur in a string representation of a distinguished name, no additional special quoting is done. This syntax has been added subsequent to RFC 1778.

Example:

```
1.3.6.1.4.1.1466.0=#04024869,O=Test,C=GB#'0101'B
```

6.22. Name Form Description

(1.3.6.1.4.1.1466.115.121.1.35 DESC 'Name Form Description')

Values in this syntax are encoded according to the following BNF. Implementors should note that future versions of this document may have expanded this BNF to include additional terms.

```
NameFormDescription = "(" whsp
    numericoid whsp  ; NameForm identifier
    [ "NAME" qdescrs ]
    [ "DESC" qdstring ]
    [ "OBSOLETE" whsp ]
    "OC" woid        ; Structural ObjectClass
    "MUST" oids      ; AttributeTypes
    [ "MAY" oids ]   ; AttributeTypes
whsp ")"
```

6.23. Numeric String

(1.3.6.1.4.1.1466.115.121.1.36 DESC 'Numeric String')

The encoding of a string in this syntax is the string value itself. Example:

1997

6.24. Object Class Description

(1.3.6.1.4.1.1466.115.121.1.37 DESC 'Object Class Description')

Values in this syntax are encoded according to the BNF in section 4.4.

6.25. OID

(1.3.6.1.4.1.1466.115.121.1.38 DESC 'OID')

Values in the Object Identifier syntax are encoded according to the BNF in section 4.1 for "oid".

Example:

1.2.3.4
cn

6.26. Other Mailbox

(1.3.6.1.4.1.1466.115.121.1.39 DESC 'Other Mailbox')

Values in this syntax are encoded according to the following BNF:

otherMailbox = mailbox-type "$" mailbox

mailbox-type = printablestring

mailbox = <an encoded IA5 String>

In the above, mailbox-type represents the type of mail system in which the mailbox resides, for example "MCIMail"; and mailbox is the actual mailbox in the mail system defined by mailbox-type.

6.27. Postal Address

(1.3.6.1.4.1.1466.115.121.1.41 DESC 'Postal Address')

Values in this syntax are encoded according to the following BNF:

```
postal-address = dstring *( "$" dstring )
```

In the above, each dstring component of a postal address value is encoded as a value of type Directory String syntax. Backslashes and dollar characters, if they occur in the component, are quoted as described in section 4.3. Many servers limit the postal address to six lines of up to thirty characters.

Example:

```
1234 Main St.$Anytown, CA 12345$USA
\241,000,000 Sweepstakes$PO Box 1000000$Anytown, CA 12345$USA
```

6.28. Presentation Address

(1.3.6.1.4.1.1466.115.121.1.43 DESC 'Presentation Address')

Values in this syntax are encoded with the representation described in RFC 1278 [6].

6.29. Printable String

(1.3.6.1.4.1.1466.115.121.1.44 DESC 'Printable String')

The encoding of a value in this syntax is the string value itself. PrintableString is limited to the characters in production p of section 4.1.

Example:

```
This is a PrintableString
```

6.30. Telephone Number

(1.3.6.1.4.1.1466.115.121.1.50 DESC 'Telephone Number')

Values in this syntax are encoded as if they were Printable String types. Telephone numbers are recommended in X.520 to be in international form, as described in E.123 [15].

Example:

```
+1 512 305 0280
```

6.31. UTC Time

(1.3.6.1.4.1.1466.115.121.1.53 DESC 'UTC Time')

Values in this syntax are encoded as if they were printable strings with the strings containing a UTCTime value. This is historical; new attribute definitions SHOULD use GeneralizedTime instead.

6.32. LDAP Syntax Description

(1.3.6.1.4.1.1466.115.121.1.54 DESC 'LDAP Syntax Description')

Values in this syntax are encoded according to the BNF in section 4.3.3.

6.33. DIT Structure Rule Description

(1.3.6.1.4.1.1466.115.121.1.17 DESC 'DIT Structure Rule Description')

Values with this syntax are encoded according to the following BNF:

```
DITStructureRuleDescription = "(" whsp
    ruleidentifier whsp            ; DITStructureRule identifier
    [ "NAME" qdescrs ]
    [ "DESC" qdstring ]
    [ "OBSOLETE" whsp ]
    "FORM" woid whsp               ; NameForm
    [ "SUP" ruleidentifiers whsp ] ; superior DITStructureRules
")"

ruleidentifier = integer

ruleidentifiers = ruleidentifier |
    "(" whsp ruleidentifierlist whsp ")"

ruleidentifierlist = [ ruleidentifier *( ruleidentifier ) ]
```

7. Object Classes

Servers SHOULD recognize all the names of standard classes from section 7 of [12].

7.1. Extensible Object Class

The extensibleObject object class, if present in an entry, permits that entry to optionally hold any attribute. The MAY attribute list of this class is implicitly the set of all attributes.

```
( 1.3.6.1.4.1.1466.101.120.111 NAME 'extensibleObject'
  SUP top AUXILIARY )
```

The mandatory attributes of the other object classes of this entry are still required to be present.

Note that not all servers will implement this object class, and those which do not will reject requests to add entries which contain this object class, or modify an entry to add this object class.

7.2. subschema

This object class is used in the subschema entry.

```
( 2.5.20.1 NAME 'subschema' AUXILIARY
  MAY ( dITStructureRules $ nameForms $ ditContentRules $
  objectClasses $ attributeTypes $ matchingRules $
  matchingRuleUse ) )
```

The ldapSyntaxes operational attribute may also be present in subschema entries.

8. Matching Rules

Servers which implement the extensibleMatch filter SHOULD allow all the matching rules listed in this section to be used in the extensibleMatch. In general these servers SHOULD allow matching rules to be used with all attribute types known to the server, when the assertion syntax of the matching rule is the same as the value syntax of the attribute.

Servers MAY implement additional matching rules.

8.1. Matching Rules used in Equality Filters

Servers SHOULD be capable of performing the following matching rules.

For all these rules, the assertion syntax is the same as the value syntax.

```
( 2.5.13.0 NAME 'objectIdentifierMatch'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
```

If the client supplies a filter using an objectIdentifierMatch whose matchValue oid is in the "descr" form, and the oid is not recognized by the server, then the filter is Undefined.

```
( 2.5.13.1 NAME 'distinguishedNameMatch'
```

```
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )

    ( 2.5.13.2 NAME 'caseIgnoreMatch'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

    ( 2.5.13.8 NAME 'numericStringMatch'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.36 )

    ( 2.5.13.11 NAME 'caseIgnoreListMatch'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.41 )

    ( 2.5.13.14 NAME 'integerMatch'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )

    ( 2.5.13.16 NAME 'bitStringMatch'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.6 )

    ( 2.5.13.20 NAME 'telephoneNumberMatch'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.50 )

    ( 2.5.13.22 NAME 'presentationAddressMatch'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.43 )

    ( 2.5.13.23 NAME 'uniqueMemberMatch'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.34 )

    ( 2.5.13.24 NAME 'protocolInformationMatch'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.42 )

    ( 2.5.13.27 NAME 'generalizedTimeMatch'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )

    ( 1.3.6.1.4.1.1466.109.114.1 NAME 'caseExactIA5Match'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

    ( 1.3.6.1.4.1.1466.109.114.2 NAME 'caseIgnoreIA5Match'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

When performing the caseIgnoreMatch, caseIgnoreListMatch, telephoneNumberMatch, caseExactIA5Match and caseIgnoreIA5Match, multiple adjoining whitespace characters are treated the same as an individual space, and leading and trailing whitespace is ignored.

Clients MUST NOT assume that servers are capable of transliteration of Unicode values.

8.2. Matching Rules used in Inequality Filters

Servers SHOULD be capable of performing the following matching rules, which are used in greaterOrEqual and lessOrEqual filters.

```
( 2.5.13.28 NAME 'generalizedTimeOrderingMatch'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )

( 2.5.13.3 NAME 'caseIgnoreOrderingMatch'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

The sort ordering for a caseIgnoreOrderingMatch is implementation-dependent.

8.3. Syntax and Matching Rules used in Substring Filters

The Substring Assertion syntax is used only as the syntax of assertion values in the extensible match. It is not used as the syntax of attributes, or in the substring filter.

```
( 1.3.6.1.4.1.1466.115.121.1.58 DESC 'Substring Assertion' )
```

The Substring Assertion is encoded according to the following BNF:

```
substring = [initial] any [final]
initial = value
any = "*" *(value "*")
final = value
```

The <value> production is UTF-8 encoded string. Should the backslash or asterix characters be present in a production of <value>, they are quoted as described in section 4.3.

Servers SHOULD be capable of performing the following matching rules, which are used in substring filters.

```
( 2.5.13.4 NAME 'caseIgnoreSubstringsMatch'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 )

( 2.5.13.21 NAME 'telephoneNumberSubstringsMatch'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 )

( 2.5.13.10 NAME 'numericStringSubstringsMatch'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 )
```

8.4. Matching Rules for Subschema Attributes

Servers which allow subschema entries to be modified by clients MUST support the following matching rules, as they are the equality matching rules for several of the subschema attributes.

(2.5.13.29 NAME 'integerFirstComponentMatch'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27)

(2.5.13.30 NAME 'objectIdentifierFirstComponentMatch'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.38)

Implementors should note that the assertion syntax of these matching rules, an INTEGER or OID, is different from the value syntax of attributes for which this is the equality matching rule.

If the client supplies an extensible filter using an objectIdentifierFirstComponentMatch whose matchValue is in the "descr" form, and the OID is not recognized by the server, then the filter is Undefined.

9. Security Considerations

9.1. Disclosure

Attributes of directory entries are used to provide descriptive information about the real-world objects they represent, which can be people, organizations or devices. Most countries have privacy laws regarding the publication of information about people.

9.2. Use of Attribute Values in Security Applications

The transformations of an AttributeValue value from its X.501 form to an LDAP string representation are not always reversible back to the same BER or DER form. An example of a situation which requires the DER form of a distinguished name is the verification of an X.509 certificate.

For example, a distinguished name consisting of one RDN with one AVA, in which the type is commonName and the value is of the TeletexString choice with the letters 'Sam' would be represented in LDAP as the string CN=Sam. Another distinguished name in which the value is still 'Sam' but of the PrintableString choice would have the same representation CN=Sam.

Applications which require the reconstruction of the DER form of the value SHOULD NOT use the string representation of attribute syntaxes when converting a value to LDAP format. Instead it SHOULD use the

Binary syntax.

10. Acknowledgements

This document is based substantially on RFC 1778, written by Tim Howes, Steve Kille, Wengyik Yeong and Colin Robbins.

Many of the attribute syntax encodings defined in this and related documents are adapted from those used in the QUIPU and the IC R3 X.500 implementations. The contributions of the authors of both these implementations in the specification of syntaxes are gratefully acknowledged.

11. Authors' Addresses

Mark Wahl
Critical Angle Inc.
4815 West Braker Lane #502-385
Austin, TX 78759
USA

Phone: +1 512 372-3160
EMail: M.Wahl@critical-angle.com

Andy Coulbeck
Isode Inc.
9390 Research Blvd Suite 305
Austin, TX 78759
USA

Phone: +1 512 231-8993
EMail: A.Coulbeck@isode.com

Tim Howes
Netscape Communications Corp.
501 E. Middlefield Rd, MS MV068
Mountain View, CA 94043
USA

Phone: +1 650 937-3419
EMail: howes@netscape.com

Steve Kille
Isode Limited
The Dome, The Square
Richmond
TW9 1DT
UK

Phone: +44-181-332-9091
EMail: S.Kille@isode.com

12. Bibliography

[1] Wahl, M., Howes, T., and S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.

[2] The Directory: Selected Attribute Types. ITU-T Recommendation X.520, 1993.

[3] The Directory: Models. ITU-T Recommendation X.501, 1993.

[4] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[5] Wahl, M., Kille, S., and T. Howes, "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names", RFC 2253, December 1997.

[6] Kille, S., "A String Representation for Presentation Addresses", RFC 1278, November 1991.

[7] Terminal Equipment and Protocols for Telematic Services - Standardization of Group 3 facsimile apparatus for document transmission. CCITT, Recommendation T.4.

[8] JPEG File Interchange Format (Version 1.02). Eric Hamilton, C-Cube Microsystems, Milpitas, CA, September 1, 1992.

[9] Yergeau, F., "UTF-8, a transformation format of Unicode and ISO 10646", RFC 2044, October 1996.

[10] Universal Multiple-Octet Coded Character Set (UCS) - Architecture and Basic Multilingual Plane, ISO/IEC 10646-1 : 1993 (With amendments).

[11] Hardcastle-Kille, S., "Mapping between X.400(1988) / ISO 10021 and RFC 822", RFC 1327, May 1992.

[12] Wahl, M., "A Summary of the X.500(96) User Schema for use with LDAPv3", RFC 2256, December 1997.

[13] Crocker, D., "Standard of the Format of ARPA-Internet Text Messages", STD 11, RFC 822, August 1982.

[14] ISO 3166, "Codes for the representation of names of countries".

[15] ITU-T Rec. E.123, Notation for national and international telephone numbers, 1988.

13. Full Copyright Statement