The drawings contained in this Recommendation have been done in AUTOCAD

```
VALUENOTATION ::=
    value(VALUE OBJECT IDENTIFIER)

SubclassOf ::=
    "SUBCLASS OF" Subclasses |
    empty

Subclasses ::=  Subclass | subclass "," 
                Subclasses
Subclass ::=  value (OBJECT-CLASS)

MandatoryAttributes ::=
    "MUST CONTAIN {"Attributes"}" | empty

OptionalAttributes ::=
    "MAY CONTAIN {"Attributes"}" | empty

Attributes      ::= AttributeTerm | AttributeTerm "," Attributes

AttributeTerm   ::= Attribute | AttributeSet

Attribute       ::= value(ATTRIBUTE)

AttributeSet    ::= value(ATTRIBUTE-SET)

END
```

The correspondence between the parts of the definition, as listed in 9.4.1, and the various pieces of the notation introduced by the macro, is as follows:

a) the object identifier to the object class is the value supplied in the value assignment of the macro;

b) the superclasses of which this object class is a subclass are those identified by the **SubclassOf** production, i.e. that following **"SUBCLASS OF";**

c) the mandatory attributes are those identified by the list of object identifiers produced by the **MandatoryAttributes** production, i.e. those following **"MUST CONTAIN";**

d) the optional attributes are those identified by the list of object identifiers produced by the **OptionalAttributes** production, i.e. those following **"MAY CONTAIN".**

Note 1 - The object identifiers in c) and d) identify both individual attributes and sets of attributes (see 9.4.7). The effective list in both cases is the set union of these. If an attribute appears in both the mandatory set and the optional set, it shall be considered mandatory.

Note 2 - The macro is used in defining selected object classes in Recommendation X.521.

Should all of the pieces of notation introduced by the macro and described in b), c), and d) above be empty, the resulting notation (**"OBJECT-CLASS"**) can be used to denote any possible object class.

9.4.7 An attribute set is a set of attributes identified by an object identifier. The definition of an attribute set involves:

a) assigning an object identifier to the set;

b) listing the object identifiers of the attributes and other attribute sets whose members together form the set.

The following ASN.1 macro may (but need not) be used to define a set of attributes for use with the **OBJECT-CLASS** macro:

```
ATTRIBUTE-SET-MACRO ::=

    BEGIN

    TYPE NOTATION ::= "CONTAINS" {"Attributes"}" | empty

    VALUE NOTATION ::= value(VALUE OBJECT IDENTIFIER)
```

Attributes ::=
AttributeTerm | AttributeTerm "," Attributes

AttributeTerm ::= Attribute | AttributeSet

Attribute ::= value(ATTRIBUTE)

AttributeSet ::= value(ATTRIBUTE-SET)

END

The correspondence between the parts of the definition of an attribute set and the notation introduced by the macro is as follows:

a) the object identifier assigned to the attribute set is the value supplied in the value assignment of the macro;

b) the set of attributes comprising the attribute set is that formed by the set union of the attributes and sets of attributes identified by the Attributes production, i.e. following **"CONTAINS".**

Should the "empty" alternative of the notation be selected, the resulting notation (**"ATTRIBUTE- SET"**) can be used to denote any possible attribute set.

9.4.8 The object classes previously mentioned are defined in 9.4.8.1, 9.4.8.2.

Note - These are partial definitions: the object identifiers are actually allocated for these object classes in Recommendation X.521 so as to provide a single point of allocation of these object identifiers in this series of Recommendations.

9.4.8.1 The object class "Top" is defined as follows:

Top ::=
OBJECT-CLASS
MUST CONTAIN {ObjectClass}

9.4.8.2 The object class **"Alias"** is defined as follows:

Alias ::=
OBJECT-CLASS
SUBCLASS OF top
MUST CONTAIN {aliasedObjectName}

Note 1 - The object class **"Alias"** does not specify appropriate attribute types for the RDN of an alias entry. Administrative Authorities may specify subclasses of the class **"Alias"** which specify useful attribute types for RDNs of alias entries (see Recommendation X.521).

Note 2 - Entries of a subclass of the class **"Alias"** are alias entries.

9.5 Attribute type definition

9.5.1 The definition of an attribute type involves:

a) assigning an object identifier to the attribute type:

b) indicating or defining the attribute syntax for the attribute type;

c) indicating whether an attribute of this type may have only one or may have more than one value (recur).

9.5.2 The Directory ensures that the indicated attribute syntax is used for every attribute of this type. The Directory also ensures that attributes of this type will have one and only one value in entries if attributes of this type are defined to have only one value.

9.5.3 The following ASN.1 macro may (but need not) be used to define an attribute type:

ATTRIBUTE MACRO ::=
BEGIN

TYPENOTATION ::= AttributeSyntax Multivalued | empty

VALUENOTATION ::= value (VALUE OBJECT IDENTIFIER)

AttributeSyntax ::=

"WITH ATTRIBUTE-SYNTAX" SyntaxChoice

Multivalued **::= "SINGLE VALUE"**
|"MULTIVALUE" | empty
SyntaxChoice **::= value(ATTRIBUTE-SYNTAX)**
Constraint | type MatchTypes
Constraint **::= "("ConstraintAlternative")" | empty**
ConstraintAlternative **::= StringConstraint | IntegerConstraint**
StringConstraint **::= "SIZE" "("SizeConstraint")"**
SizeConstraint **::= SingleValue | Range**
SingleValue **::= value(INTEGER)**
Range **::= value(INTEGER) ".." value**
(INTEGER)
IntegerConstraint **::= Range**
MatchTypes **::= "MATCHES FOR" Matches | empty**
Matches **::= Match Matches | Match**
Match **::= "EQUALITY" | "SUBSTRINGS" |**
"ORDERING"

END

The correspondence between the parts of the definition, as listed in 9.5.1, and the various pieces of the notation introduced by the macro, is as follows:

a) the object identifier assigned to the attribute type is the value supplied in the value assignment of the MACRO;

b) the attribute syntax for the attribute type is that identified by the **AttributeSyntax** production. This either points to a separately defined attribute syntax, or explicitly defines an attribute syntax by giving its ASN.1 type and matching rules (see 9.6). If a separately identified attribute syntax is employed, a size constraint for underlying string types or a value range for an underlying integer type may optionally be indicated;

c) the attribute is single valued if the MultiValued production is **"SINGLE VALUE"**, and may have one or more values if it is **"MULTI VALUE"** or empty.

Note - The macro is used in defining selected attribute types in Recommendation X.520.

Should the "empty" alternative of the type notation be selected, the resulting notation (**"ATTRIBUTE"**) can be used to denote any possible attribute type.

9.5.4 The attribute types identified in 7.3.3 which are known to and used by the Directory for its own purposes are defined as follows:

ObjectClass ::= ATTRIBUTE
WITH ATTRIBUTE-SYNTAX objectIdentifierSyntax

AliasedObjectName ::= ATTRIBUTE
WITH ATTRIBUTE-SYNTAX distinguishedNameSyntax
SINGLE VALUE

Note 1 - These are partial definitions: the object identifiers are actually allocated for these attribute types in Recommendation X.520 so as to provide a single point of allocation of these object identifiers in this series of Recommendations.

Note 2 - The attribute syntaxes referred to in these definitions are themselves defined in 9.6.5.

9.6 Attribute syntax definition

9.6.1 The definition of an attribute syntax involves:

a) optionally, assigning an object identifier to the attribute syntax;

b) indicating the data type, in ASN.1 of the attribute syntax;

c) defining appropriate rules for matching a presented value with a target attribute value held in the DIB. None, some, or all of the following matching rules may be defined for a particular attribute syntax:

i) equality. Applicable to any attribute syntax. The presented value must conform to the data type of the attribute syntax;

ii) substrings. Applicable to any attribute syntax with a string data type. The presented value must be a sequence (**"SEQUENCE OF"**), each of whose elements conforms to the data type;

iii) ordering. Applicable to any attribute syntax for which a rule can be defined that will allow a presented value to be described as less than, equal to, or greater than a target value. The presented value must conform to the data type of the attribute syntax.

9.6.2 If no equality matching rule is defined, the Directory:

a) treats values as attributes of this attribute syntax as having type **ANY**, i.e. the Directory does not check that those values conform with the data type indicated for the attribute syntax;

b) will not attempt to match presented values against target values of such an attribute type.

Note - It follows that the Directory will not permit such an attribute to be used in a distinguished name, nor allow for a specific value to be modified.

9.6.3 If an equality matching rule is defined, the Directory:

a) treats values of attributes of this attribute syntax as having type **ANY DEFINED BY** the data type indicated for the attribute syntax;

b) will only match according to the matching rules defined for that attribute syntax;

c) will only match a presented value of a suitable data type as specified in 9.6.1 c).

9.6.4 The following ASN.1 macro may, but need not, be used to define attribute syntaxes:

```
ATTRIBUTE-SYNTAX MACRO ::=
BEGIN

TYPE NOTATION ::=  Syntax
                   MatchTypes | empty

VALUE NOTATION ::=
    value (VALUE OBJECT IDENTIFIER)

Syntax ::= type

MatchTypes ::= "MATCHES FOR" Matches | empty

Matches ::= Match Matches | Match

Match ::=    "EQUALITY" | "SUBSTRINGS" | "ORDERING"

END
```

The correspondence between the parts of the definition, as listed in 9.6.1, and the various pieces of the notation introduced by the macro, is as follows:

a) the object identifier assigned to the attribute syntax is a value supplied in the value assignment of the macro;

b) the data type of the attribute syntax is that identified by the Syntax production, i.e. that following macro name;

c) the defined matching rules are equality, if **"EQUALITY"** appears in the **MatchTypes** production, substrings if **"SUBSTRINGS"** appears, and ordering if **"ORDERING"** appears. If the production is empty, then no matching rules are defined.

Should the "empty" alternative of the notation be selected, the resulting notation (**"ATTRIBUTE - SYNTAX"**) can be used to denote any possible attribute syntax.

Note 1 - No support is provided in the macro for actually defining the matching rules themselves: this must be done by natural language or by other means.

Note 2 - The macro is used in defining selected attribute syntaxes in Recommendation X.520.

9.6.5 The attribute syntaxes used in 9.5.4 are defined in 9.6.5.1 and 9.6.5.2.

Note - These are partial definitions: the object identifiers are actually allocated for these attribute syntaxes in Recommendation X.520 so as to provide a single point of allocation of these object identifiers in this series of Recommendations.

9.6.5.1 **ObjectIdentifierSyntax** is defined as follows:

```
ObjectIdentifierSyntax ::=
    ATTRIBUTE-SYNTAX
        OBJECT IDENTIFIER
        MATCHES FOR EQUALITY
```

The matching rule for equality is inherent in the definition of the ASN.1 type object identifier.

9.6.5.2 **DistinguishedNameSyntax** is defined as follows:

```
DistinguishedNameSyntax ::=
    ATTRIBUTE-SYNTAX
        DistinguishedName
        MATCHES FOR EQUALITY
```

A presented distinguished name value is equal to a target distinguished name value if and only if all of the following are true:

a) the number of RDNs in each is the same;

b) corresponding RDNs have the same number of AVAs;

c) corresponding AVAs (i.e. those with identical attribute types) have attribute values which match for equality (in such a match, the attribute values take the same roles - i.e. as presented or target value - as the distinguished name which contains them does in the overall match).

SECTION 3 - Security model

10 Security

10.1 The directory exists in an environment where various authorities provide access to their fragment of the DIB. Such access shall be in conformance to the security policy (see Recommendation X.509) of the security domain in which the fragment of the DIB exists.

10.2 Two specific components of a security policy are addressed here:

a) the definition of an authorization policy;

b) the definition of an authentication policy.

10.3 The definition of authorization in the context of the Directory includes the methods to:

a) specify access rights;

b) enforce access rights (access control);

c) maintain access rights.

10.4 The definition of authentication in the context of the Directory includes the methods to verify:

a) the identity of DSAs and directory users;

b) the identity of the origin of received information at an access point.

The integrity of received information is a local matter and shall be in conformance to the security policy in force.

10.5 This Recommendation does not define a Security Policy.

10.6 Annex F describes guidelines for specifying access rights.

10.7 Recommendation X.509 defines authentication procedures. The DAP and DSP may provide strong authentication of the initiator by the signing of the request, data integrity of the request by signing of the request, strong authentification of the responder and data integrity of the result by signing the result. The DAP may provide simple authentication between a DUA and a DSA. The DSP may provide simple authentication between two DSAs.

10.8 Administrative authorities of applications which make use of the Directory can use their own security policy. The directory can support applications by holding authentication information (e.g. distinguished names, passwords, certificates) about communication entities. This is further described in Recommendation X.509.

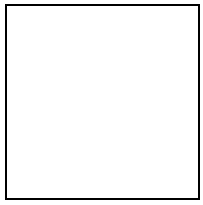
ANNEX A

(to Recommendation X.501)

THE MATHEMATICS OF TREES

This Annex is not part of the standard.

FIGURE - -T0704360-88



A tree is a set of points, called vertices, and a set of directed lines, called arcs; each arc a leads from a vertex V to a vertex Vw'. For example, the tree in the Figure has seven vertices, labelled V^1 through V^7, and six arcs, labelled a^1 through a^6.

Two vertices V and Vw' are said to be the initial and final vertices, respectively, of an arc a from V to Vw'.For example, V^2 and V^3 are the initial and final vertices, respectively, of arc a^2. Several different arcs may have the same initial vertex, but not the same final vertex. For example, arcs a^1 and a^3 have the same initial vertex, V^1, but no two arcs in the Figure have the same final vertex.

The vertex that is not the final vertex of any arc is often referred to as the root vertex, or even more informally as the "root" of the tree. For example, in the Figure, V^1 is the root.

A vertex that is not the initial vertex of any arc is often referred to informally as a leaf vertex, or even more informally, as a "leaf" of the tree graph. For example, vertices V^3, V^6, and V^7 are leaves.

An oriented path from a vertex V to a vertex Vw' is a set of arcs $(a^1, a^2, ..., a^n)$ (n w_ 1) such that V is the initial vertex of arc a^1, Vw' is the final vertex of arc a^n, and the final vertex of arc a^k is also the initial vertex of arc a^{k+1} for 1 w_ k < n. For example, the oriented path from vertex V^1 to vertex V^6 is the set of arcs (a^3, a^4, a^5). The term "path" should be understood to denote an oriented path from the root to a vertex.

ANNEX B

(to Recommendation X.501)

OBJECT IDENTIFIER USAGE

This Annex is part of the standard.

This Annex documents the upper reaches of the object identifier subtree in which all of the object identifiers assigned in this series of Recommendations reside. It does so by providing an ASN.1 module called "UsefulDefinitions" in which all non-leaf nodes in the subtree are assigned names.

UsefulDefinitions {joint-iso-ccitt ds(5) modules(1)
usefulDefinitions(0)}

```
DEFINITIONS ::=

BEGIN

EXPORTS
    module, serviceElement, applicationContext, attributeType, attributeSyntax, objectClass,
    algorithm, abstractSyntax, attributeSet,

    usefulDefinitions, informationFramework, directoryAbstractService,
    directoryObjectIdentifiers, algorithmObjectIdentifiers, distributedOperations,
    protocolObjectIdentifiers, selectedAttributeTypes, selectedObjectClasses,
    authenticationFramework, upperBounds,
    dap,dsp

    id-ac, id-ase, id-as, id-ot, id-pt;

ds              OBJECT IDENTIFIER ::=  {joint-iso-ccitt ds(5)}

 -- categories of information object --

module                  OBJECT IDENTIFIER ::= {ds 1}
serviceElement          OBJECT IDENTIFIER ::= {ds 2}
applicationContext      OBJECT IDENTIFIER ::= {ds 3}
attributeType           OBJECT IDENTIFIER ::= {ds 4}
attributeSyntax         OBJECT IDENTIFIER ::= {ds 5}
objectClass             OBJECT IDENTIFIER ::= {ds 6}
attributeSet            OBJECT IDENTIFIER ::= {ds 7}
algorithm               OBJECT IDENTIFIER ::= {ds 8}
abstractSyntax          OBJECT IDENTIFIER ::= {ds 9}
object                  OBJECT IDENTIFIER ::= {ds 10}
port                    OBJECT IDENTIFIER ::= {ds 11}

 -- modules --

usefulDefinitions                       OBJECT IDENTIFIER ::= {module 0}
informationFramework                    OBJECT IDENTIFIER ::= {module 1}
directoryAbstractService                OBJECT IDENTIFIER ::= {module 2}
distributedOperations                   OBJECT IDENTIFIER ::= {module 3}
protocolObjectIdentifier                OBJECT IDENTIFIER ::= {module 4}
selectedAttributeTypes                  OBJECT IDENTIFIER ::= {module 5}
selectedObjectClasses                   OBJECT IDENTIFIER ::= {module 6}
authenticationFramework                 OBJECT IDENTIFIER ::= {module 7}
algorithmObjectIdentifiers              OBJECT IDENTIFIER ::= {module 8}
directoryObjectIdentifiers              OBJECT IDENTIFIER ::= {module 9}
upperBounds                             OBJECT IDENTIFIER ::= {module 10}
dap                                     OBJECT IDENTIFIER ::= {module 11}
dsp                                     OBJECT IDENTIFIER ::= {module 12}
distributedDirectoryObjectIdentifiers   OBJECT IDENTIFIER ::= {module 13}

 -- synonyms --

id-ac           OBJECT IDENTIFIER ::= applicationContext
id-ase          OBJECT IDENTIFIER ::= serviceElement
id-as           OBJECT IDENTIFIER ::= abstractSyntax
id-ot           OBJECT IDENTIFIER ::= object
id-pt           OBJECT IDENTIFIER ::= port

END
```

ANNEX C

(to Recommendation X.501)

INFORMATION FRAMEWORK IN ASN.1

This Annex is part of the standard.

This Annex provides a summary of all of the ASN.1 type, value, and macro definitions contained in this Recommendation. The definitions form the ASN.1 module **"InformationFramework".**

```
InformationFramework{joint-iso-ccitt ds(5) modules(1)
                     informationFramework(1)}

DEFINITIONS ::=
BEGIN

EXPORTS
    Attribute, AttributeType, AttributeValue, AttributeValueAssertion,
    DistinguishedName, Name, RelativeDistinguishedName,
    OBJECT-CLASS,ATTRIBUTE,ATTRIBUTE-SET,ATTRIBUTE-SYNTAX,
    Top, Alias,
    ObjectClass, AliasedObjectName,
    ObjectIdentifierSyntax, DistinguishedNameSyntax;

IMPORTS
    selectedAttributeTypes, selectedObjectClasses
        FROM    UsefulDefinitions {joint-iso-ccitt ds(5) modules(1)
                usefulDefinitions(0)}

    top
        FROM SelecedObjectClasses selectedObjectClasses
    objectIdentifierSyntax, distinguishedNameSyntax, objectClass, aliasedObjectName
        FROM SelectedAttributeTypes selectedAttributeTypes;

-- attribute data types --

Attribute                   ::= SEQUENCE{
                                    type        AttributeType
                                    values SET OF AttributeValue
                                    -- at least one value is required --}

AttributeType               ::= OBJECT IDENTIFIER

AttributeValue              ::= ANY

AttributeValueAssertion     ::= SEQUENCE {AttributeType, AttributeValue}

-- naming data types --

Name            ::=     CHOICE {-- only one possibility for now --
                        RDNSequence}

RDNSequence                     ::=     SEQUENCE OF
                                            RelativeDistinguishedName

DistinguishedName               ::=     RDNSequence

RelativeDistinguishedName       ::=     SET OF AttributeValueAssertion

-- macros --

OBJECT-CLASS MACRO      ::=
BEGIN
    TYPENOTATION        ::=     SubclassOf MandatoryAttributes
                                    OptionalAttributes

    VALUENOTATION       ::=     value (VALUE OBJECT IDENTIFIER)
    SubclassOf          ::=     "SUBCLASS OF" Subclasses | empty
    Subclasses          ::=     Subclass | Subclass "," Subclasses
```

```
    Subclass                ::=   value (OBJECT-CLASS)
    MandatoryAttributes     ::=   "MUST CONTAIN {"Attributes"}" | empty
    OptionalAttributes      ::=   "MAY CONTAIN {"Attributes"}" | empty
    Attributes              ::=   AttributeTerm | AttributeTerm "," Attributes
    AttributeTerm           ::=   Attribute | AttributeSet
    Attribute               ::=   value(ATTRIBUTE)
    AttributeSet            ::=   value(ATTRIBUTE-SET)
END
ATTRIBUTE-SET-MACRO     ::=
    BEGIN
    TYPE NOTATION           ::=   "CONTAINS" "{Attributes"}" | empty
    VALUE NOTATION          ::=   value(VALUEOBJECTIDENTIFIER)
    Attributes              ::=   AttributeTerm | AttributeTerm "," Attributes
    AttributeTerm           ::=   Attribute | AttributeSet
    Attribute               ::=   value(ATTRIBUTE)
    AttributeSet            ::=   value(ATTRIBUTE-SET)
    END
ATTRIBUTE MACRO         ::=
BEGIN
TYPENOTATION            ::=   AttributeSyntax Multivalued | empty
VALUENOTATION           ::=   value(VALUE OBJECT IDENTIFIER)
AttributeSyntax         ::=   "WITH ATTRIBUTE-SYNTAX"  SyntaxChoice
Multivalued             ::=   "SINGLE VALUE" | "MULTI VALUE" | empty
SyntaxChoice            ::=   value(ATTRIBUTE-SYNTAX)
                              Constraint | type Match Types
Constraint              ::=   "("ConstraintAlternative")" | empty
ConstraintAlternative   ::=   StringConstraint | IntegerConstraint
StringConstraint        ::=   "SIZE" "("SizeConstraint")"
SizeConstraint          ::=   SingleValue | Range
SingleValue             ::=   value(INTEGER)
Range                   ::=   value(INTEGER) ".." value(INTEGER)
IntegerConstraint       ::=   Range
MatchTypes              ::=   "MATCHES FOR" Matches | empty
Matches                 ::=   Match Matches | Match
Match                   ::=   "EQUALITY" | "SUBSTRINGS" | "ORDERING"
END
ATTRIBUTE-SYNTAX MACRO ::=
BEGIN
    TYPENOTATION        ::=   Syntax MatchTypes | empty
    VALUENOTATION       ::=   value(VALUE OBJECT IDENTIFIER)
    Syntax              ::=   type
    MatchTypes          ::=   "MATCHES FOR "Matches | empty
    Matches             ::=   Match Matches | Match
    Match               ::=   "EQUALITY" | "SUBSTRINGS" | "ORDERING"
END
-- object classes --
Top     ::=     OBJECT-CLASS
```

```
                    MUST CONTAIN {objectClass}
Alias     ::=   OBJECT-CLASS
                    SUBCLASS OF top
                    MUST CONTAIN {aliasedObjectName}
-- attribute types --
ObjectClass     ::= ATTRIBUTE
                                WITH ATTRIBUTE-SYNTAX objectIdentifierSyntax
AliasedObjectName ::= ATTRIBUTE
                                WITH ATTRIBUTE-SYNTAX distinguishedNameSyntax
                                SINGLE VALUE
-- attribute syntaxes --
ObjectIdentifierSyntax ::=
    ATTRIBUTE-SYNTAX
    OBJECT IDENTIFIER
    MATCHES FOR EQUALITY
DistinguishedNameSyntax ::=
    ATTRIBUTE-SYNTAX
    DistinguishedName
    MATCHES FOR EQUALITY
END
```

ANNEX D

(to Recommendation X.501)

ALPHABETICAL INDEX OF DEFINITIONS

This Annex is not part of the standard.

This Annex alphabetically lists all of the terms defined in this Recommendation together with a cross reference to the in which they are defined.

A	access point	5
	Administration Directory Management Domain	5
	alias	8
	alias entry	6
	attribute	7
	attribute type	7
	attribute value	7
	attribute value assertion	7
D	the Directory	5
	Directory entry	6
	Directory Information Base (DIB)	6
	Directory Information Tree (DIT)	6
	Directory Management Domain (DMD)	5
	Directory name	8
	Directory schema	9
	Directory System Agent (DSA)	5
	Directory User Agent (DUA)	5
	distinguished name	8
	DIT Structure Rule	9
E	entry	6
I	immediate(ly) subordinate	6
	immediate(ly) superior	6
N	name	8
	naming authority	8
O	object (of interest)	6
	object class	6
	object entry	6
P	Private Directory Management Domain	5
	purported name	8
R	relative distinguished name	8
S	subordinate	6
	superior	6

ANNEX E

(to Recommendation X.501)

NAME DESIGN CRITERIA

This Annex is not part of the standard.

The information framework is very general, and allows for arbitrary variety of entries and attributes within the DIT. Since, as defined there, names are closely related to paths through the DIT, this means that arbitrary variety in names is possible. This section suggests criteria to be considered in the design of names. The appropriate criteria have been used in the design of the recommended name forms which are to be found in Recommendation X.521. It is suggested that the criteria also be used, where appropriate, in designing the names for objects to which the recommended name forms do not apply.

Presently, only one criterion is addressed: that of user-friendliness.

Note - Not all names need to be user-friendly.

E.1 User-friendliness

Names with which human beings must deal directly should be user-friendly. A user-friendly name is one that takes the human user's point of view, not the computer's. It is one that is easy for people to deduce, remember, and understand, rather than one that is easy for computers to interpret.

The goal of user-friendliness can be stated somewhat more precisely in terms of the following two principles:

- A human being usually should be able to correctly guess an object's user-friendly name on the basis of information about the object that he naturally possesses. For example, one should be able to guess a business person's name given only the information about her casually acquired through normal business association.
- When an object's name is ambiguously specified, the Directory should recognize the fact rather than conclude that the name identifies one particular object. For example, where two people have the same last name, the last name alone should be considered inadequate identification of either party.

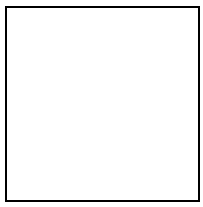
The following subgoals follow from the goal of user-friendliness:

a) Names should not artificially remove natural ambiguities. For example, if two people share the last name "Jones", neither should be required to answer to "WJones" or "Jones2". Instead, the naming convention should provide a user-friendly means of discriminating between the entities. For example, it might require first name and middle initial in addition to last name.

b) Names should admit common abbreviations and common variations in spelling. For example, if one is employed by the Conway Steel Corporation and the name of one's employer figures in one's name, any of the names "Conway Steel Corporation", "Conway Steel Corp.", "Conway Steel", and "CSC" should suffice to identify the organization in question.

c) In certain cases, alias names can be used to direct the search for a particular entry, in order to be more user-friendly, or to reduce the scope of a search. The following example demonstrates the use of an alias name for such a purpose: as shown in Figure E-1/X.501, the branch office in Osaka can also be identified with the name {C = Japan, L = Osaka, O = ABC, OU = Osaka-branch}.

FIGURE E-1/X.501 - T0704370-88



d) If names are multi-part, both the number of mandatory parts and the number of optional parts should be relatively small and thus easy to remember.

e) If names are multi-part, the precise order in which those parts appear should generally be immaterial.

f) User-friendly names should not involve computer addresses.

ANNEX F

(to Recommendation X.501)

ACCESS CONTROL

This Annex is not part of the standard.

F.1 Introduction

Directory users are granted access to the information in the DIB on the basis of their access control rights in accordance with the access control policy in force protecting that information.

Access Control is left as a local matter in this series of Recommendations. However, it is recognized that implementations will need to introduce means of controlling access and that future versions of this series of Recommendations are likely to define standardized means of creating, maintaining and applying access control information. This Annex describes the principles underlying access control, and outlines two possible approaches to access control.

F.2 Principles

The two principles that will guide the establishment of procedures for managing access control are:

a) there must be means of protecting information in the Directory from unauthorized detection, examination, and modification, including protecting the DIT from unauthorized modification;

b) the information required to determine a user's rights to perform a given operation must be available to the DSA(s) involved in performing the operation in order to avoid further remote operations solely to determine these rights.

F.3 Protected items

These levels of protection are presently identified:

a) protection of an entire subtree of the DIT;

b) protection of an individual entry;

c) protection of an entire attribute within an entry;

d) protection of selected instances of attribute values.

F.4 Access categories

A need for at least five categories of access is envisaged. If access is not granted to a protected item in any category, then the directory in so far as is possible responds as though their protected item did not exist at all.

The categories of access are shown in Table F-1/X.501. The items column denotes whether the item that can be so protected is an entry (E), an attribute (A) or both (EA).

TABLE F-1/X.501

Access categories

Category	Items	Description
detect	A	Allows the protected item to be detected.
compare	A	Allows a presented value to be compared to the protected item.
read	A	Allows the protected item to be read.
modify	A	Allows the protected item to be updated.
add/delete	EA	Allows the creation and deletion of new components (attributes or attribute values) within the protected item.
naming	E	Allows the modification of the Relative Distinguished Name of, and the creation and deletion of, entries which are immediately subordinate to the protected entry.

F.5 Determination of access rights

One scheme for managing access control associates with every protected item, either explicitely or implicitely, a list of access rights. Each item in such a list pairs a set of users with a set of access categories.

Determining if a user is in one (or more) of the noted sets must be possible from the information supplied with the request - either from the authenticated identity and credentials of the user as supplied in BIND, or from information carried in the operation argument.

There at least two possibilities:

a) The sets are described in terms of the distinguished names of the users they identify either the distinguished name of the user or the distinguished name of a superior with a flag specifying that the entire subtree is included.

b) The sets give only a capability, and implicitly include all users having that capability. This scheme requires that such users' capability be available locally or else carried in the BIND or operation argument. The latter may require an extension to the currently defined protocols.