Lightweight Directory Access Protocol (LDAP) Procedures and Schema Definitions for the Storage of X.660 Registration Information

draft-coretta-x660-ldap-07.txt

Abstract

This specification defines models, procedures and schema definitions meant to facilitate the storage of [X.660] registration data within a Lightweight Directory Access Protocol Directory Information Tree.

Status of This Memo

This Internet-Draft will expire on January 24, 2022.

Copyright Notice

Table of Contents

1. Introduction

This specification describes a means for storing [X.660] registration and OPTIONAL registrant data within an LDAP [RFC4510] implementation.

Some additional concepts and strategies are introduced that are not explicitly defined in either [X.660] or [X.680]. These concepts are meant to facilitate ease-of-implementation and sensible consumption.

1.1. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Acronyms Used

This specification makes reference to several acronyms, each of which are defined below.

DN	Distinguished Name
RA	Registration Authority
DIT	Directory Information Tree
DSA	Directory System Agent
DUA	Directory User Agent (an LDAP client)
GUI	Graphical User Interface
IRI	Internationalized Resource Identifier
OID	ASN.1 Object Identifier
PEN	IANA Private Enterprise Number
RDN	Relative Distinguished Name
TUI	Textual User Interface
URI	Uniform Resource Identifier
LDAP	Lightweight Directory Access Protocol
ASN.1	Abstract Syntax Notation one

1.3. Intended Audience

This specification is intended for use by any entity or individual in need of a means for storing and serving [X.660] data, in whole or in part. The most likely candidates for use are RAs, whether internal to an organization or public.

1.4. OIDs Allocated

This specification provides a dedicated registered OID branch for all LDAP schema elements as defined in Section 2, as well as OIDs for two (2) directory models defined in Section 3.

- 1.3.6.1.4.1.56521 (author root)
- 1.3.6.1.4.1.56521.101 (this specification)
- 1.3.6.1.4.1.56521.101.2 (schema)
- 1.3.6.1.4.1.56521.101.2.1 (attribute types)
- 1.3.6.1.4.1.56521.101.2.2 (object classes)
- 1.3.6.1.4.1.56521.101.3 (models)
- 1.3.6.1.4.1.56521.101.3.2 (twoDimensional)
- 1.3.6.1.4.1.56521.101.3.3 (threeDimensional)

1.5. Well-Known OIDs

This specification makes use of well-known OIDs defined by other parties or institutions. These OIDs are mentioned for example purposes and schema configuration only.

- 1.3 (Identified-Organization, per Section A.4.2 of [X.660])

- 1.3.6 (dod, per Section 3.1 of [RFC1155])

- 1.3.6.1 (Internet OID, per Section 3.1 of [RFC1155])

- 1.3.6.1.4.1.1466.115.121.1.12 (Distinguished Name syntax and matching rule, per Section 4.2.15 of [RFC4517])

- 1.3.6.1.4.1.1466.115.121.1.24 (Generalized Time syntax, per Section 3.3.13 of [RFC4517])

- 1.3.6.1.4.1.1466.115.121.1.27 (Integer syntax, per Section 3.3.16 of [RFC4517])

- 1.3.6.1.4.1.1466.115.121.1.38 (OID syntax, per Section 3.3.26 of [RFC4517])

- 1.3.6.1.4.1.1466.115.121.1.40 (Octet String syntax, per Section 3.3.25 of [RFC4517])

2. Schema Definitions

This section discusses the particulars of the LDAP schema definitions made available through this specification.

These schema definitions described in this section are provided using LDAP description formats [RFC4512]. These elements are line-wrapped and indented for readability.

2.1. Attribute Types

The following subsections detail LDAP attribute types created for use within implementations of this specification.

Please note that a great many of these attribute type definitions are sub types of attribute types defined in the following Standards Documents, and as such are dependencies:

- [RFC2079] for URI support
- [RFC4519] for so-called "core" schema elements
- [RFC4524] for Cosine schema elements

If the nature of a particular directory implementation precludes the use of sub-typed attributes, this specification may not be practical for adoption.

If a directory architect opts to extend certain definitions in this Section for the purpose of Collective Attribute [RFC3671] support, they are advised to adhere to the prescribed attribute type naming prefix ("c-"), per Section 3 of [RFC3671].

Very few attribute types are actually REQUIRED for registrations and not all types will necessarily apply to every situation. Directory architects are advised to carefully identify what types are needed, and make interactive procedures and "best practices" available and well-documented for all relevant consuming entities.

2.1.1. 'n'

The 'n' attribute type allows the assignment of an unsigned integer, meant to represent the primary identifier, or NumberForm [X.680], of the entry.

This attribute type plays a crucial role with regards to DN syntax used in the three-dimensional directory model described in Section 3.3.

```
( 1.3.6.1.4.1.56521.101.2.1.1
    NAME ( 'n' 'numberForm' )
    DESC 'A single unsigned integer value assigned to
       a registration to represent its primary integer
       identifier'
    EQUALITY integerMatch
    SINGLE-VALUE
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )
```

Examples: "56521", "0"

2.1.2. 'dotNotation'

The 'dotNotation' attribute type allows the assignment of an OID value [X.680] in dot-delimited form to a registration entry.

```
( 1.3.6.1.4.1.56521.101.2.1.2
    NAME 'dotNotation'
    DESC 'Dotted ASN.1 Object Identifier for a
       registration'
    EQUALITY objectIdentifierMatch
    SINGLE-VALUE
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
```

Examples: "1.3.6.1", "2.999"

2.1.3. 'iRI'

The 'iRI' attribute type allows the assignment of one or more IRI values to an registration entry.

```
    ( 1.3.6.1.4.1.56521.101.2.1.3
        NAME 'iRI'
        DESC 'Internationalized Resource Identifiers
           for a registration'
        EQUALITY octetStringMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 )
```

Examples: "/ITU-T", "/ISO/Identified-Organization", "/ASN.1"

2.1.4. 'asn1Notation'

The 'asn1Notation' attribute type allows the assignment of an OID in ASN.1 notation to a registration entry.

```
    ( 1.3.6.1.4.1.56521.101.2.1.4
        NAME 'asn1Notation'
        DESC 'An ordered sequence of NameAndNumberForm
           or NumberForm values, enclosed within curly
           braces, that identify an OID'
        SINGLE-VALUE
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

Examples: "{itu-t(0)}", "{iso(1) identified-organization(3)}"

2.1.5. 'unicodeValue'

The 'unicodeValue' attribute type allows the assignment of one or more Unicode-based primary identifiers (non-numeric) [X.660] to a registration entry.

Although multi-value support is positive, cases in which any given registration has multiple 'unicodeValue' values is extremely rare.

```
    ( 1.3.6.1.4.1.56521.101.2.1.5
        NAME 'unicodeValue'
        DESC 'Primary non-numeric Unicode identifiers
           for a registration'
        EQUALITY octetStringMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 )
```

Examples: "ITU-T", "Identified-Organization"

2.1.6. 'identifier'

The 'identifier' attribute type allows the assignment of a single non-Unicode, non-numeric identifier [X.660], or NameForm [X.680], to a registration entry.

The first character MUST be a lowercase letter and MUST NOT include any white space.

The same value can be applied to the 'nameAndNumberForm' attribute as a partial value. See Section 2.1.17 for more information on this attribute type.

The attribute type 'name', as defined in Section 2.18 of [RFC4519], is a super type of this attribute type.

```
  ( 1.3.6.1.4.1.56521.101.2.1.6
      NAME ( 'identifier' 'nameForm' )
      DESC 'The non-Unicode secondary identifier
         for a registration'
      SINGLE-VALUE
      SUP name )
```

Examples: "itu-t", "iso"

2.1.7. 'additionalIdentifier'

The 'additionalIdentifier' attribute type allows the assignment of of one or more additional identifiers [X.660] in a registration.

The attribute type 'name', as defined in Section 2.18 of [RFC4519], is a super type of this attribute type.

```
  ( 1.3.6.1.4.1.56521.101.2.1.7
      NAME 'additionalIdentifier'
      DESC 'The non-Unicode additional identifiers
         or nameForms for a registration'
      SUP name )
```

Examples: "enterprises", "ccitt"

2.1.8. 'registrationInformation'

The 'registrationInformation' attribute type allows the OPTIONAL assignment of octet based values intended for extended information relating to the registration in question.

```
  ( 1.3.6.1.4.1.56521.101.2.1.8
      NAME 'registrationInformation'
      DESC 'Extended octet-based data for
         a registration'
      EQUALITY octetStringMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.40{4096} )
```

2.1.9. 'registrationURI'

The 'registrationURI' attribute type allows for the assignment of one or more URI values, with optional labels, to a registration.

The attribute type 'labeledURI', as defined in [RFC2079], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.9
   NAME 'registrationURI'
   DESC 'URI, with an optional label, leading to
      further related subject matter information'
   SUP labeledURI )
```

Examples: "http://example.com Example", "http://example.com"

2.1.10. 'registrationCreated'

The 'registrationCreated' attribute type allows for the assignment of a generalized timestamp indicating the date and time at which a registration was, or will be, created or officiated.

```
( 1.3.6.1.4.1.56521.101.2.1.10
   NAME 'registrationCreated'
   DESC 'Generalized timestamp for
      registration creation'
   SINGLE-VALUE
   EQUALITY generalizedTimeMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )
```

Examples: "19951231115959Z", "20130109033116Z"

2.1.11. 'registrationModified'

The 'registrationModified' attribute type allows for the assignment of one or more generalized timestamp values indicating the dates and times of all applied updates to a registration.

Whether multiple dates, or only most recent date, are stored is entirely up to the directory architect(s) involved.

```
( 1.3.6.1.4.1.56521.101.2.1.11
   NAME 'registrationModified'
   DESC 'Generalized timestamps for
      registration modifications'
   EQUALITY generalizedTimeMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )
```

Examples: "19951231115959Z", "20130109033116Z"

2.1.12. 'registrationRange'

The 'registrationRange' attribute type allows for the expression of an OID allocation range, such as "100" to indicate 'up to 100', as well as "-1" to indicate 'to infinity'.

The implied start of a range is the 'n' value of the registration entry that bears this attribute type, and would not need to be set explicitly.

The value of this attribute MUST always be greater-than and NOT equal to the value of 'n' EXCEPT when "-1" is used.

For example, if a 'rangeTermintor' attribute value of '999' were set for the OID '2.999.44', it MUST be interpreted as an entire range of OIDs starting at '2.999.44' up to and including '2.999.999'.

Similarly, keeping with the same example above, if a value of "-1" were used instead, this MUST be interpreted as an all-encompassing OID range starting at '2.999.44' with absolutely no upper limit.

DUAs used to manage and allocate registration entries MUST always perform preemptive searches for occurrences of this attribute type prior to allocating any sibling registrations, so as to avoid illegal overlaps.

```
( 1.3.6.1.4.1.56521.101.2.1.12
   NAME 'registrationRange'
   DESC 'Sibling registration range expression'
   EQUALITY integerMatch
   SINGLE-VALUE
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )
```

Examples: "-1", "1999", "1000000"

2.1.13. 'registrationStatus'

The 'registrationStatus' attribute type allows for the assignment of status information, indicative of the current state of the registration. Multiple values can be provided, however not all combinations are meaningful.

In most cases, this is used to mark a registration as "obsolete", "reserved", "private" or "deallocated", however values of "active" or "in-force" are also applicable in some cases.

Absence of this attribute type within a given entry SHOULD be viewed as an implied declaration of "active" or "in-force".

A value of "private" MAY be used a component in any access control mechanics defined by the directory architect(s), but at the risk of possible performance costs depending on implementation. In addition, it is possible to leverage the 'discloseTo' attribute type defined in Section 2.1.26. See Section 5 for more information.

Other possible values not listed here MAY be used, however they would be wholly proprietary in nature.

```
     ( 1.3.6.1.4.1.56521.101.2.1.13
        NAME 'registrationStatus'
        DESC 'Current status of a registration'
        SINGLE-VALUE
        SUP description )
```

Examples: "obsolete", "deallocated", "reserved"

2.1.14. 'isLeafNode'

The 'isLeafNode' attribute type allows for the assignment of a single Boolean value indicative of whether a registration can be a parent to any subordinate registrations.

Absence of this attribute type SHOULD be interpreted as an implicit FALSE value.

A value of FALSE indicates there are no restrictions regarding the allocation or enumeration of any child entries.

A value of TRUE forbids the enumeration of all existing subordinate registrations, as well as the creation of new registrations.

When TRUE, this attribute type implies a TRUE value for 'isFrozen', per Section 2.1.15. See Section 3.6.1.2.2.1 for considerations on various combinations of these attribute types.

This attribute type is NOT meant to serve in a security capacity. If it is desirable to limit the enumeration of subordinate registrations with privacy or security in mind, the attribute types 'discloseTo' (Section 2.1.26) and/or 'registrationStatus' (Section 2.1.13) would be more appropriate.

```
     ( 1.3.6.1.4.1.56521.101.2.1.14
         NAME 'isLeafNode'
         DESC 'Whether a registration may allocate,
            or allow the enumeration of, subordinate
            registrations'
         EQUALITY booleanMatch
         SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )
```

Example: "TRUE", "FALSE", or UNDEFINED (implies "FALSE")

2.1.15. 'isFrozen'

The 'isFrozen' attribute type allows for the assignment of a single Boolean value indicative of whether a registration can be a parent to any further subordinate registrations beyond those that already exist at present.

A value of TRUE indicates that the given registration MUST NOT have any further children allocated, but that any entries that are already allocated SHALL be enumerated.

A value of FALSE indicates there are no restrictions regarding the allocation of any subsequent child entries, UNLESS a value of TRUE is defined for the 'isLeafNode' attribute type, per Section 2.1.14. Such a value supersedes any state of this attribute type. Please see Section 3.6.1.2.2.1 for considerations on various combinations of these attribute types.

```
  ( 1.3.6.1.4.1.56521.101.2.1.15
      NAME 'isFrozen'
      DESC 'Whether a registration may allocate
         any additional subordinate registrations'
      EQUALITY booleanMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )
```

Example: "TRUE", "FALSE", or UNDEFINED (implies "FALSE")

2.1.16. 'stdNameForm'

The 'stdNameForm' attribute type allows for the assignment of one or more Standardized NameForm values, per [X.660] and [X.680], to a registration.

```
   ( 1.3.6.1.4.1.56521.101.2.1.16
       NAME 'stdNameForm'
       DESC 'Standardized NameForm per X.680'
       EQUALITY caseExactMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

Example: "{itu-t}", "{0 0 d}"

2.1.17. 'nameAndNumberForm'

The 'nameAndNumberForm' attribute type allows for the assignment of an [X.680] NameAndNumberForm value to a registration.

The value assigned to this attribute MUST manifest as a combination of values of 'identifier' and/or (at a minimum) 'n'.

```
   ( 1.3.6.1.4.1.56521.101.2.1.17
       NAME 'nameAndNumberForm'
       DESC 'NameAndNumberForm value, per X.680'
       EQUALITY caseExactMatch
       SINGLE-VALUE
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

   Example: "private(4)", "itu-t(0)", "56521"

2.1.18. 'longArc'

The 'longArc' attribute type allows the assignment of one or more so-called "Long Arc" well-known identifiers to a registration.

Per [X.660], entries that bear values of this attribute type MUST reside below the root joint-iso-itu-t(2) registration.

```
( 1.3.6.1.4.1.56521.101.2.1.18
   NAME 'longArc'
   DESC 'The well-known Long Arc names associated with,
      and registered to, a Joint-ISO-ITU-T subordinate
      registration'
   EQUALITY octetStringMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 )
```

Examples: "/Example", "/Ejemplo"

2.1.19. 'supArc'

The 'supArc' attribute type allows for the assignment of an LDAP DN value to a registration, thereby identifying the DN of the immediate superior (parent) registration entry.

For example, the OID registration for 2.1.4 (ecn) would possess a DN value for this attribute type identifying its superior registration entry as 2.1 (asn1).

This OPTIONAL attribute type is among the seven (7) spatial types discussed in more detail in Section 3.6.

The attribute type 'distinguishedName', as defined in Section 2.7 of [RFC4519], is a super type of this attribute type. This sub type is MULTI-VALUED so as to support Collective Attribute extensibility [RFC3671], but is intended only for the storage of a single DN value per registration.

```
 ( 1.3.6.1.4.1.56521.101.2.1.19
    NAME 'supArc'
    DESC 'LDAP Distinguished Name of the logically
       superior immediate registration'
    SUP distinguishedName )
```

Example: "n=1,n=2,ou=OID,ou=X660,dc=example,dc=com"

2.1.20. 'topArc'

The 'topArc' attribute type allows for the assignment of an LDAP DN value to a registration identifying the superior root registration.

For example, the OID registration for 2.1.4 (ecn) would possess a DN value for this attribute type identifying its root registration entry as 2 (Joint-ISO-ITU-T).

This OPTIONAL attribute type is among the seven (7) spatial types discussed in more detail in Section 3.6.

The attribute type 'distinguishedName', as defined in Section 2.7 of [RFC4519], is a super type of this attribute type. This sub type is MULTI-VALUED so as to support Collective Attribute extensibility [RFC3671], but is intended only for the storage of a single DN value per registration.

```
  ( 1.3.6.1.4.1.56521.101.2.1.20
      NAME 'topArc'
      DESC 'LDAP Distinguished Name of the absolute
         superior root registration'
      SUP distinguishedName )
```

Example: "n=2,ou=OID,ou=X660,dc=example,dc=com"

2.1.21. 'subArc'

The 'subArc' attribute type allows for the assignment of one or more LDAP DN values to a registration as a manifest of subordinate registrations residing exactly one (1) logical level below.

This OPTIONAL attribute type is among the seven (7) spatial types discussed in more detail in Section 3.6.

The attribute type 'distinguishedName', as defined in Section 2.7 of [RFC4519], is a super type of this attribute type.

```
  ( 1.3.6.1.4.1.56521.101.2.1.21
      NAME 'subArc'
      DESC 'LDAP Distinguished Names of immediate
         subordinate registrations'
      SUP distinguishedName )
```

Example: "n=1,n=6,n=3,n=1,ou=OID,ou=X660,dc=example,dc=com"

2.1.22. 'leftArc'

The 'leftArc' attribute type allows for the assignment of an LDAP DN value to a registration. The value SHOULD reference the nearest lexically-antecedent (left-hand) sibling registration.

As an example, given a registration for the OID 2.6 (mhs), the DN referencing the OID 2.5 (ds) is the lexically-nearest antecedent sibling registration.

This OPTIONAL attribute type is among the seven (7) spatial types discussed in more detail in Section 3.6.

The attribute type 'distinguishedName', as defined in Section 2.7 of [RFC4519], is a super type of this attribute type.

```
     ( 1.3.6.1.4.1.56521.101.2.1.22
         NAME 'leftArc'
         DESC 'LDAP Distinguished Name of the nearest
            lexically antecedent sibling registration'
         SINGLE-VALUE
         SUP distinguishedName )
```

Example: "n=5,n=2,ou=OID,ou=X660,dc=example,dc=com"

2.1.23. 'firstArc'

The 'firstArc' attribute type allows for the assignment of an LDAP DN value to a registration. The value SHOULD reference the absolute farthest antecedent sibling registration.

As an example, given a registration for the OID 2.6 (mhs), while the DN referencing 2.0 (presentation) is the lexically-farthest antecedent sibling registration.

This OPTIONAL attribute type is among the seven (7) spatial types discussed in more detail in Section 3.6.

The attribute type 'distinguishedName', as defined in Section 2.7 of [RFC4519], is a super type of this attribute type. This sub type is MULTI-VALUED so as to support Collective Attribute extensibility [RFC3671], but is intended only for the storage of a single DN value per registration.

```
     ( 1.3.6.1.4.1.56521.101.2.1.23
         NAME 'firstArc'
         DESC 'LDAP Distinguished Name of the farthest
            lexically antecedent sibling registration'
         SUP distinguishedName )
```

Example: "n=0,n=2,ou=OID,ou=X660,dc=example,dc=com"

2.1.24. 'rightArc'

The 'rightArc' attribute type allows for the assignment of an LDAP DN value to a registration. The value SHOULD reference the nearest subsequent sibling registration.

As an example, given a registration for the OID 2.1 (asn1), the DN referencing OID 2.2 (association-control) is the lexically-nearest subsequent sibling registration.

This OPTIONAL attribute type is among the seven (7) spatial types discussed in more detail in Section 3.6.

The attribute type 'distinguishedName', as defined in Section 2.7 of [RFC4519], is a super type of this attribute type.

```
    ( 1.3.6.1.4.1.56521.101.2.1.24
        NAME 'rightArc'
        DESC 'LDAP Distinguished Name of the nearest
           lexically subsequent sibling registration'
        SINGLE-VALUE
        SUP distinguishedName )
```

Example: "n=2,n=2,ou=OID,ou=X660,dc=example,dc=com"

2.1.25. 'finalArc'

The 'finalArc' attribute type allows for the assignment of an LDAP DN value to a registration. The value SHOULD reference the absolute farthest subsequent sibling registration.

As an example, given a registration for the OID 2.1 (asn1), the DN referencing OID 2.999 (example) is lexically-farthest subsequent sibling registration.

This OPTIONAL attribute type is among the seven (7) spatial types discussed in more detail in Section 3.6.

The attribute type 'distinguishedName', as defined in Section 2.7 of [RFC4519], is a super type of this attribute type. This sub type is MULTI-VALUED so as to support Collective Attribute extensibility [RFC3671], but is intended only for the storage of a single DN value per registration.

```
    ( 1.3.6.1.4.1.56521.101.2.1.25
        NAME 'finalArc'
        DESC 'LDAP Distinguished Name of the farthest
           lexically subsequent sibling registration'
        SUP distinguishedName )
```

Example: "n=999,n=2,ou=OID,ou=X660,dc=example,dc=com"

2.1.26. 'discloseTo'

The 'discloseTo' attribute type allows for the assignment of one or more LDAP DN values to a registration, each of which reference an identity that is authorized to access the entry's information in read-only fashion. This MAY cover any depth of subordinate entry disclosures as seen fit by the directory architect(s).

Identities referenced through use of this attribute type MAY be single user entries, a 'groupOfNames'-based entry, per Section 3.5 of [RFC4519] or a current authority or sponsorship registrant.

Write-based access SHOULD NOT be governed by this attribute type, as that is an intended function of the current registration authority.

The attribute type 'distinguishedName', as defined in Section 2.7 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.26
    NAME 'discloseTo'
    DESC 'LDAP Distinguished Names of entries which
       are granted access to a given registration
       and its immediate children'
    SUP distinguishedName )
```

Example: "cn=ClearanceLevel4,ou=Groups,dc=example,dc=com"

2.1.27. 'registrantID'

The 'registrantID' attribute type is intended to allow for singular assignment of a UUID, GUID or some other auto-generated value to a registrant entry. When used, this value would act as an absolute identifier for registration entries that may change in the future.

In larger, more complete implementations of this specification, it is RECOMMENDED that this attribute type be the primary identifier (or, RDN) for registrant entries. This allows for an absolute and unambiguous reference to any registration entry by DN in terms of authority and/or sponsorship.

```
( 1.3.6.1.4.1.56521.101.2.1.27
   NAME 'registrantID'
   DESC 'Random identifier assigned
      to a registrant'
   SINGLE-VALUE
   EQUALITY octetStringMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 )
```

Examples: "rfc4519", "69118e61-cc02-4c50-bde7-5bdaf4e973e4"

2.1.28. 'currentAuthority'

The 'currentAuthority' attribute type allows for the assignment of one or more DN values to a registration.

The value(s) of this attribute type are meant to refer to distinct entries that contain current registrant authority information for the registration to which it is linked.

This attribute type is only required if registrant information is not stored within a given registration directly.

The attribute type 'distinguishedName', as defined in Section 2.7 of [RFC4519], is a super type of this attribute type.

```
 ( 1.3.6.1.4.1.56521.101.2.1.28
     NAME 'currentAuthority'
     DESC 'LDAP Distinguished Name of an entry
        bearing current registration authority
        information'
     SUP distinguishedName )
```

Example: "registrantID=XYZ,ou=Registrants,ou=X660,dc=example,dc=com"

2.1.29. 'currentAuthorityStartTimestamp'

The 'currentAuthorityStartTimestamp' attribute type allows for the assignment of a generalized timestamp value to a current registration authority, indicative of the date and time at which the current registration authority was, or will be, officiated.

```
 ( 1.3.6.1.4.1.56521.101.2.1.29
    NAME 'currentAuthorityStartTimestamp'
    DESC 'Generalized time stamp indicating the date
       and time at which current authority commenced'
    EQUALITY generalizedTimeMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )
```

Example: "19951231115959Z"

2.1.30. 'currentAuthorityCommonName'

The 'currentAuthorityCommonName' attribute type allows for the assignment of a common name to a current authority entry.

The attribute type 'cn', as defined in Section 2.3 of [RFC4519], is a super type of this attribute type.

```
 ( 1.3.6.1.4.1.56521.101.2.1.30
    NAME 'currentAuthorityCommonName'
    DESC 'Common Name assigned to a current
       registration authority entry'
    SINGLE-VALUE
    SUP cn )
```

Example: "Jesse Coretta", "Jane Smith"

2.1.31. 'currentAuthorityCountryCode'

The 'currentAuthorityCountryCode' attribute type allows for the assignment of a country code to a current authority entry.

The attribute type 'c', as defined in Section 2.2 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.31
   NAME 'currentAuthorityCountryCode'
   DESC 'Country Code assigned to a current
      registration authority entry'
   SINGLE-VALUE
   SUP c )
```

Examples: "US", "CA"

2.1.32. 'currentAuthorityCountryName'

The 'currentAuthorityCountryName' attribute type allows for the assignment of a country name to a current authority entry.

The attribute type 'co', as defined in Section 2.4 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.32
   NAME 'currentAuthorityCountryName'
   DESC 'Country name assigned to a current
      registration authority entry'
   SINGLE-VALUE
   SUP co )
```

Examples: "United States", "Canada"

2.1.33. 'currentAuthorityEmail'

The 'currentAuthorityEmail' attribute type allows for the assignment of an email address to the current registration authority entry.

The attribute type 'mail', as defined in Section 2.16 of [RFC4524], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.33
   NAME 'currentAuthorityEmail'
   DESC 'Email address assigned to a current
      registration authority entry'
   SINGLE-VALUE
   SUP mail )
```

Example: "jesse.coretta@icloud.com"

2.1.34. 'currentAuthorityFax'

The 'currentAuthorityFax' attribute type allows for the assignment of a facsimile telephone number to a current authority entry.

The attribute type 'facsimileTelephoneNumber', as defined in Section 2.10 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.34
   NAME 'currentAuthorityFax'
   DESC 'Facsimile telephone number assigned to
      a current registration authority entry'
   SINGLE-VALUE
   SUP facsimileTelephoneNumber )
```

Example: "+11234567890"

2.1.35. 'currentAuthorityLocality'

The 'currentAuthorityLocality' attribute type allows for the assignment of a locality name to a current authority entry.

The attribute type 'l', as defined in Section 2.16 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.35
   NAME 'currentAuthorityLocality'
   DESC 'Locality name assigned to a current
      registration authority entry'
   SINGLE-VALUE
   SUP l )
```

Example: "Palm Springs", "Anna Maria Island"

2.1.36. 'currentAuthorityMobile'

The 'currentAuthorityMobile' attribute type allows for the assignment of a mobile telephone number to a current registration authoritative entry.

The attribute type 'mobile', as defined in Section 2.18 of [RFC4524], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.36
   NAME 'currentAuthorityMobile'
   DESC 'Mobile telephone number assigned to a
      current registration authority entry'
   SINGLE-VALUE
   SUP mobile )
```

Example: "+11234567890"

2.1.37. 'currentAuthorityOrg'

The 'currentAuthorityOrg' attribute type allows for the assignment of an organization name to a current authority entry.

The attribute type 'o', as defined in Section 2.19 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.37
   NAME 'currentAuthorityOrg'
   DESC 'Organization name assigned to a current
      registration authority entry'
   SINGLE-VALUE
   SUP o )
```

Example: "Acme, Co."

2.1.38. 'currentAuthorityPOBox'

The 'currentAuthorityPOBox' attribute type allows for the assignment of a post office box number to a current registration authority entry.

The attribute type 'postOfficeBox', as defined in Section 2.25 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.38
   NAME 'currentAuthorityPOBox'
   DESC 'Post office box number assigned to a
      current registration authority entry'
   SINGLE-VALUE
   SUP postOfficeBox )
```

Examples: "555", "475"

2.1.39. 'currentAuthorityPostalAddress'

The 'currentAuthorityPostalAddress' attribute type allows for the assignment of a complete postal address to a current registration authority entry. This single attribute may be used instead of other individual address component attribute types, but will require field parsing on the client side.

The attribute type 'postalAddress', as defined in Section 2.23 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.39
   NAME 'currentAuthorityPostalAddress'
   DESC 'Full postal address assigned to a current
      registration authority entry'
   SINGLE-VALUE
   SUP postalAddress )
```

Example: "1 Fake St$Anytown$CA$12345$US"

2.1.40. 'currentAuthorityPostalCode'

The 'currentAuthorityPostalCode' attribute type allows for the assignment of a postal code to a current authority entry.

The attribute type 'postalCode', as defined in Section 2.23 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.40
   NAME 'currentAuthorityPostalCode'
   DESC 'Postal code assigned to a current
      registration authority entry'
   SINGLE-VALUE
   SUP postalCode )

   Examples: "92262", "34216"
```

2.1.41. 'currentAuthorityState'

The 'currentAuthorityState' attribute type allows for the assignment of a state or province name to a current registration authority entry.

The attribute type 'st', as defined in Section 2.33 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.41
   NAME 'currentAuthorityState'
   DESC 'State or province name assigned to a current
      registration authority entry'
   SINGLE-VALUE
   SUP st )
```

Examples: "California", "North Dakota"

2.1.42. 'currentAuthorityStreet'

The 'currentAuthorityStreet' attribute type allows for the assignment of a street name and number to a current registration authority entry.

The attribute type 'street', as defined in Section 2.34 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.42
   NAME 'currentAuthorityStreet'
   DESC 'Street name and number assigned to a current
      registration authority entry'
   SINGLE-VALUE
```

```
        SUP street )
```

Example: "1 Fake Street"

2.1.43. 'currentAuthorityTelephone'

The 'currentAuthorityTelephone' attribute type allows for the assignment of a telephone number to a current authority entry.

The attribute type 'telephoneNumber', as defined in Section 2.35 of [RFC4519], is a super type of this attribute type.

```
  ( 1.3.6.1.4.1.56521.101.2.1.43
     NAME 'currentAuthorityTelephone'
     DESC 'Telephone number assigned to a current
        registration authority entry'
     SINGLE-VALUE
     SUP telephoneNumber )
```

Example: "+11234567890"

2.1.44. 'currentAuthorityTitle'

The 'currentAuthorityTitle' attribute type allows for the assignment of an official or professional title to a current authority entry.

The attribute type 'title', as defined in Section 2.38 of [RFC4519], is a super type of this attribute type.

```
  ( 1.3.6.1.4.1.56521.101.2.1.44
     NAME 'currentAuthorityTitle'
     DESC 'Title assigned to a current
        registration authority entry'
     SINGLE-VALUE
     SUP title )
```

Example: "Chief Engineer"

2.1.45. 'currentAuthorityURI'

The 'currentAuthorityURI' attribute type allows for the assignment of one or more URI values, with optional labels, to a current authority entry.

The attribute type 'labeledURI', as defined in [RFC2079], is a super type of this attribute type.

```
  ( 1.3.6.1.4.1.56521.101.2.1.45
     NAME 'currentAuthorityURI'
     DESC 'URI, with an optional label, leading
        to a resource related to a current
        registration authority'
        SUP labeledURI )
```

Example: "http://example.com Example", "http://example.com"

2.1.46. 'firstAuthority'

The 'firstAuthority' attribute type allows for the assignment of one or more DN values to an registration entry.

The value(s) of this attribute type are meant to refer to distinct entries that contain previous authority information.

This attribute type is only required if registrant information is not stored within a given registration directly.

The attribute type 'distinguishedName', as defined in Section 2.7 of [RFC4519], is a super type of this attribute type.

```
  ( 1.3.6.1.4.1.56521.101.2.1.46
      NAME 'firstAuthority'
      DESC 'LDAP Distinguished Name of an entry
         bearing previous registration authority
         information'
      SUP distinguishedName )
```

Example: "registrantID=XYZ,ou=Registrants,ou=X660,dc=example,dc=com"

2.1.47. 'firstAuthorityStartTimestamp'

The 'firstAuthorityStartTimestamp' attribute type allows for the assignment of a generalized timestamp value to a previous registration authority, indicative of the date and time at which the previous registration authority was, or will be, officiated.

```
  ( 1.3.6.1.4.1.56521.101.2.1.47
     NAME 'firstAuthorityStartTimestamp'
     DESC 'Generalized timestamp indicating the date and
        time at which a previous registration authority
        commenced'
     EQUALITY generalizedTimeMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )
```

Example: "20130105135904Z"

2.1.48. 'firstAuthorityEndTimestamp'

The 'firstAuthorityEndTimestamp' attribute type allows for the assignment of a generalized timestamp value to a previous registration authority, indicative of the date and time at which an entity's authoritative role was, or will be, terminated.

```
     ( 1.3.6.1.4.1.56521.101.2.1.48
        NAME 'firstAuthorityEndTimestamp'
        DESC 'Generalized timestamp indicating the date and
           time at which a previous registration authority
           terminated'
        EQUALITY generalizedTimeMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )
```

Example: "20170528110555Z"

2.1.49. 'firstAuthorityCommonName'

The 'firstAuthorityCommonName' attribute type allows for the assignment of a common name to a previous registration authority entry.

The attribute type 'cn', as defined in Section 2.3 of [RFC4519], is a super type of this attribute type.

```
     ( 1.3.6.1.4.1.56521.101.2.1.49
        NAME 'firstAuthorityCommonName'
        DESC 'Common Name assigned to a previous
           registration authority entry'
        SINGLE-VALUE
        SUP cn )
```

Examples: "Jesse Coretta", "Jane Smith"

2.1.50. 'firstAuthorityCountryCode'

The 'firstAuthorityCountryCode' attribute type allows for the assignment of a country code to a previous registration authority entry.

The attribute type 'c', as defined in Section 2.2 of [RFC4519], is a super type of this attribute type.

```
     ( 1.3.6.1.4.1.56521.101.2.1.50
        NAME 'firstAuthorityCountryCode'
        DESC 'Country Code assigned to a previous
           registration authority entry'
        SINGLE-VALUE
        SUP c )
```

Examples: "US", "CA"

2.1.51. 'firstAuthorityCountryName'

The 'firstAuthorityCountryName' attribute type allows for the assignment of a country name to a previous registration authority entry.

The attribute type 'co', as defined in Section 2.4 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.51
   NAME 'firstAuthorityCountryName'
   DESC 'Country name assigned to a previous
      registration authority entry'
   SINGLE-VALUE
   SUP co )
```

Examples: "United States", "Canada"

2.1.52. 'firstAuthorityEmail'

The 'firstAuthorityEmail' attribute type allows for the assignment of an email address to a previous registration authority entry.

The attribute type 'mail', as defined in Section 2.16 of [RFC4524], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.52
   NAME 'firstAuthorityEmail'
   DESC 'Email address assigned to a previous
      registration authority entry'
   SINGLE-VALUE
   SUP mail )
```

Example: "jesse.coretta@icloud.com"

2.1.53. 'firstAuthorityFax'

The 'firstAuthorityFax' attribute type allows for the assignment of a facsimile telephone number to a previous registration authority entry.

The attribute type 'facsimileTelephoneNumber', as defined in Section 2.10 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.53
   NAME 'firstAuthorityFax'
   DESC 'Facsimile telephone number assigned to
      a previous registration authority entry'
   SINGLE-VALUE
   SUP facsimileTelephoneNumber )
```

Example: "+11234567890"

2.1.54. 'firstAuthorityLocality'

The 'firstAuthorityLocality' attribute type allows for the assignment of a locality name to a previous registration authority entry.

The attribute type 'l', as defined in Section 2.16 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.54
   NAME 'firstAuthorityLocality'
   DESC 'Locality name assigned to a previous
      registration authority entry'
   SINGLE-VALUE
   SUP l )
```

Examples: "Palm Springs", "Anna Maria Island"

2.1.55. 'firstAuthorityMobile'

The 'firstAuthorityMobile' attribute type allows for the assignment of a mobile telephone number to a previous registration authoritative entry.

The attribute type 'mobile', as defined in Section 2.18 of [RFC4524], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.55
   NAME 'firstAuthorityMobile'
   DESC 'Mobile telephone number assigned to a
      previous registration authority entry'
   SINGLE-VALUE
   SUP mobile )
```

Example: "+11234567890"

2.1.56. 'firstAuthorityOrg'

The 'firstAuthorityOrg' attribute type allows for the assignment of an organization name to a previous registration authority entry.

The attribute type 'o', as defined in Section 2.19 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.56
   NAME 'firstAuthorityOrg'
   DESC 'Organization name assigned to a previous
      registration authority entry'
   SINGLE-VALUE
   SUP o )
```

Example: "Acme, Co."

2.1.57. 'firstAuthorityPOBox'

The 'firstAuthorityPOBox' attribute type allows for the assignment of a post office box number to a previous registration authority entry.

The attribute type 'postOfficeBox', as defined in Section 2.25 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.57
   NAME 'firstAuthorityPOBox'
   DESC 'Post office box number assigned to a
      previous registration authority entry'
   SINGLE-VALUE
   SUP postOfficeBox )
```

Examples: "555", "475"

2.1.58. 'firstAuthorityPostalAddress'

The 'firstAuthorityPostalAddress' attribute type allows for the assignment of a complete postal address to a previous registration authority entry. This single attribute may be used instead of other individual address component attribute types, but will require field parsing on the client side.

The attribute type 'postalAddress', as defined in Section 2.23 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.58
   NAME 'firstAuthorityPostalAddress'
   DESC 'Full postal address assigned to a previous
      registration authority entry'
   SINGLE-VALUE
   SUP postalAddress )
```

Example: "1 Fake St$Anytown$CA$12345$US"

2.1.59. 'firstAuthorityPostalCode'

The 'firstAuthorityPostalCode' attribute type allows for the assignment of a postal code to a previous registration authority entry.

The attribute type 'postalCode', as defined in Section 2.23 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.59
   NAME 'firstAuthorityPostalCode'
   DESC 'Postal code assigned to a previous
      registration authority entry'
   SINGLE-VALUE
   SUP postalCode )
```

Examples: "92262", "34216"

2.1.60. 'firstAuthorityState'

The 'firstAuthorityState' attribute type allows for the assignment of a state or province name to a previous registration authority entry.

The attribute type 'st', as defined in Section 2.33 of [RFC4519], is a super type of this attribute type.

```
  ( 1.3.6.1.4.1.56521.101.2.1.60
     NAME 'firstAuthorityState'
     DESC 'State or province name assigned to a previous
        registration authority entry'
     SINGLE-VALUE
     SUP st )
```

Examples: "California", "North Dakota"

2.1.61. 'firstAuthorityStreet'

The 'firstAuthorityStreet' attribute type allows for the assignment of a street name and number to a previous registration authority entry.

The attribute type 'street', as defined in Section 2.34 of [RFC4519], is a super type of this attribute type.

```
  ( 1.3.6.1.4.1.56521.101.2.1.61
     NAME 'firstAuthorityStreet'
     DESC 'Street name and number assigned to a previous
        registration authority entry'
     SINGLE-VALUE
     SUP street )
```

Example: "1 Fake Street"

2.1.62. 'firstAuthorityTelephone'

The 'firstAuthorityTelephone' attribute type allows for the assignment of a telephone number to a previous registration authority entry.

The attribute type 'telephoneNumber', as defined in Section 2.35 of [RFC4519], is a super type of this attribute type.

```
  ( 1.3.6.1.4.1.56521.101.2.1.62
     NAME 'firstAuthorityTelephone'
     DESC 'Telephone number assigned to a previous
        registration authority entry'
```

```
         SINGLE-VALUE
         SUP telephoneNumber )
```

Example: "+11234567890"

2.1.63. 'firstAuthorityTitle'

The 'firstAuthorityTitle' attribute type allows for the assignment of an official or professional title to a previous registration authority entry.

The attribute type 'title', as defined in Section 2.38 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.63
   NAME 'firstAuthorityTitle'
   DESC 'Title assigned to a previous
      registration authority entry'
   SINGLE-VALUE
   SUP title )
```

Example: "Chief Engineer"

2.1.64. 'firstAuthorityURI'

The 'firstAuthorityURI' attribute type allows for the assignment of one or more URI values, with optional labels, to a previous registration authority entry.

The attribute type 'labeledURI', as defined in [RFC2079], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.64
   NAME 'firstAuthorityURI'
   DESC 'URI, with an optional label, leading
      to a resource related to a previous
      registration authority'
   SUP labeledURI )
```

Examples: "http://example.com Example", "http://example.com"

2.1.65. 'sponsor'

The 'sponsor' attribute type allows for the assignment of one or more DN values to a registration.

The value(s) of this attribute type are meant to refer to distinct entries that contains sponsorship-related information for a given registration.

This attribute type is only required if such information is not stored within a given registration directly.

The attribute type 'distinguishedName', as defined in Section 2.7 of [RFC4519], is a super type of this attribute type.

```
     ( 1.3.6.1.4.1.56521.101.2.1.65
         NAME 'sponsor'
         DESC 'LDAP Distinguished Name of an entry
            bearing sponsorship information'
         SUP distinguishedName )
```

Example: "registrantID=XYZ,ou=Registrants,ou=X660,dc=example,dc=com"

2.1.66. 'sponsorStartTimestamp'

The 'sponsorStartTimestamp' attribute type allows for the assignment of a generalized timestamp value to a sponsor entry, indicative of the date and time at which sponsorship was, or will be, officiated.

```
     ( 1.3.6.1.4.1.56521.101.2.1.66
        NAME 'sponsorStartTimestamp'
        DESC 'Generalized timestamp indicating the date
           and time sponsorship commenced'
        EQUALITY generalizedTimeMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )
```

Example: "20130105135904Z"

2.1.67. 'sponsorEndTimestamp'

The 'sponsorEndTimestamp' attribute type allows for the assignment of a generalized timestamp value to a sponsor entry, indicative of the date and time at which sponsorship was, or will be, terminated.

```
     ( 1.3.6.1.4.1.56521.101.2.1.67
        NAME 'sponsorEndTimestamp'
        DESC 'Generalized timestamp indicating the date
           and time sponsorship terminated'
        EQUALITY generalizedTimeMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )
```

Example: "20170528110555Z"

2.1.68. 'sponsorCommonName'

The 'sponsorCommonName' attribute type allows for the assignment of a common name to a sponsor entry.

The attribute type 'cn', as defined in Section 2.3 of [RFC4519], is a super type of this attribute type.

```
     ( 1.3.6.1.4.1.56521.101.2.1.68
        NAME 'sponsorCommonName'
        DESC 'Common Name of a sponsor entry'
        SINGLE-VALUE
        SUP cn )
```

Example: "Jane Sponsor"

2.1.69. 'sponsorCountryCode'

The 'sponsorCountryCode' attribute type allows for the assignment of a two-letter country code to a sponsor entry.

The attribute type 'c', as defined in Section 2.2 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.69
   NAME 'sponsorCountryCode'
   DESC 'Country code for a sponsor entry'
   SINGLE-VALUE
   SUP c )
```

Examples: "US", "CA"

2.1.70. 'sponsorCountryName'

The 'sponsorCountryName' attribute type allows the assignment of a country name to a sponsor entry.

The attribute type 'co', as defined in Section 2.4 of [RFC4524], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.70
   NAME 'sponsorCountryName'
   DESC 'Country name for a sponsor entry'
   SINGLE-VALUE
   SUP co )
```

Examples: "United States", "Canada"

2.1.71. 'sponsorEmail'

The 'sponsorEmail' attribute type allows for the assignment of an email address to a sponsor entry.

The attribute type 'mail', as defined in Section 2.16 of [RFC4524], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.71
   NAME 'sponsorEmail'
   DESC 'Email address for a sponsor entry'
   SINGLE-VALUE
   SUP mail )
```

Example: "sponsor@example.com"

2.1.72. 'sponsorFax'

The 'sponsorFax' attribute type allows for the assignment of a facsimile telephone number to a sponsor entry.

The attribute type 'facsimileTelephoneNumber', as defined in Section 2.10 of [RFC4519], is a super type of this attribute type.

```
  ( 1.3.6.1.4.1.56521.101.2.1.72
     NAME 'sponsorFax'
     DESC 'Facsimile telephone number for a sponsor entry'
     SINGLE-VALUE
     SUP facsimileTelephoneNumber )
```

Example: "+11234567890"

2.1.73. 'sponsorLocality'

The 'sponsorLocality' attribute type allows for the assignment of a locality name to a sponsor entry.

The attribute type 'l', as defined in Section 2.16 of [RFC4519], is a super type of this attribute type.

```
  ( 1.3.6.1.4.1.56521.101.2.1.73
     NAME 'sponsorLocality'
     DESC 'Locality name for a sponsor entry'
     SINGLE-VALUE
     SUP l )
```

Examples: "Palm Springs", "Anna Maria Island"

2.1.74. 'sponsorMobile'

The 'sponsorMobile' attribute type allows for the assignment of a mobile telephone number to a sponsor entry.

The attribute type 'mobile', as defined in Section 2.18 of [RFC4524], is a super type of this attribute type.

```
  ( 1.3.6.1.4.1.56521.101.2.1.74
     NAME 'sponsorMobile'
     DESC 'Mobile telephone number for a sponsor entry'
     SINGLE-VALUE
     SUP mobile )
```

Example: "+11234567890"

2.1.75. 'sponsorOrg'

The 'sponsorOrg' attribute type allows for the assignment of an organization name to a sponsor entry.

The attribute type 'o', as defined in Section 2.19 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.75
   NAME 'sponsorOrg'
   DESC 'Organization name for a sponsor entry'
   SINGLE-VALUE
   SUP o )
```

Example: "Sponsor, Co."

2.1.76. 'sponsorPOBox'

The 'sponsorPOBox' attribute type allows for the assignment of a post office box number to a sponsor entry.

The attribute type 'postOfficeBox', as defined in Section 2.25 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.76
   NAME 'sponsorPOBox'
   DESC 'Post office box number for a sponsor entry'
   SINGLE-VALUE
   SUP postOfficeBox )
```

Examples: "555", "475"

2.1.77. 'sponsorPostalAddress'

The 'sponsorPostalAddress' attribute type allows for the assignment of a complete postal address sponsor entry. This single attribute may be used instead of other individual address component attribute types, but will require field parsing on the client side.

The attribute type 'postalAddress', as defined in Section 2.23 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.77
   NAME 'sponsorPostalAddress'
   DESC 'Full postal address for a sponsor entry'
   SINGLE-VALUE
   SUP postalAddress )
```

Example: "1 Fake St$Anytown$CA$12345$US"

2.1.78. 'sponsorPostalCode'

The 'sponsorPostalCode' attribute type allows for a postal code to be assigned to a sponsor entry.

The attribute type 'postalCode', as defined in Section 2.23 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.78
   NAME 'sponsorPostalCode'
   DESC 'Postal code for a sponsor entry'
   SINGLE-VALUE
   SUP postalCode )
```

Example: "92262", "34216"

2.1.79. 'sponsorState'

The 'sponsorState' attribute type allows for the assignment of a state or province name to a sponsor entry.

The attribute type 'st', as defined in Section 2.33 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.79
   NAME 'sponsorState'
   DESC 'State or province name for a sponsor entry'
   SINGLE-VALUE
   SUP st )
```

Examples: "California", "North Dakota"

2.1.80. 'sponsorStreet'

The 'sponsorStreet' attribute type allows for the assignment of a street name and number to a sponsor entry.

The attribute type 'street', as defined in Section 2.34 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.80
   NAME 'sponsorStreet'
   DESC 'Street name and number for a sponsor entry'
   SINGLE-VALUE
   SUP street )
```

Example: "1 Fake Street"

2.1.81. 'sponsorTelephone'

The 'sponsorTelephone' attribute type allows for the assignment of a telephone number to a sponsor entry.

The attribute type 'telephoneNumber', as defined in Section 2.35 of [RFC4519], is a super type of this attribute type.

```
     ( 1.3.6.1.4.1.56521.101.2.1.81
        NAME 'sponsorTelephone'
        DESC 'Telephone number for a sponsor entry'
        SINGLE-VALUE
        SUP telephoneNumber )
```

Example: "+11234567890"

2.1.82. 'sponsorTitle'

The 'sponsorTitle' attribute type allows for the assignment of an official or professional title to a sponsor entry.

The attribute type 'title', as defined in Section 2.38 of [RFC4519], is a super type of this attribute type.

```
     ( 1.3.6.1.4.1.56521.101.2.1.82
        NAME 'sponsorTitle'
        DESC 'Title for a sponsor entry'
        SINGLE-VALUE
        SUP title )
```

Example: "Executive Sponsor"

2.1.83. 'sponsorURI'

The 'sponsorURI' attribute type allows for the assignment of one or more URI values, each with an optional label, to a sponsor entry.

The attribute type 'labeledURI', as defined in [RFC2079], is a super type of this attribute type.

```
     ( 1.3.6.1.4.1.56521.101.2.1.83
        NAME 'sponsorURI'
        DESC 'URI, with an optional label, for a
           sponsor entry'
        SUP labeledURI )
```

Examples: "http://example.com Example", "http://example.com"

2.1.84. 'rARegistrationBase'

The 'rARegistrationBase' attribute type allows for the storage of an LDAP DN meant to store the location of registration entries within the DIT. Clients SHOULD expect entries of types x660RootArc or x660SubArc exactly one (1) level below this DN.

This is primarily used in scenarios where it is desirable for an application to self-configure for maximum efficiency in terms of registration location, management and retrieval.

An ideal location for this attribute type is within the DSA's RootDSE.

The attribute type 'distinguishedName', as defined in Section 2.7 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.84
    NAME 'rARegistrationBase'
    DESC 'LDAP Distinguished Name of the root X.660
       registration entry storage location in a DIT'
    SUP distinguishedName )
```

Example: "ou=OID,ou=X660,dc=example,dc=com"

2.1.85. 'rARegistrantBase'

The 'rARegistrantBase' attribute type allows for the storage of an LDAP DN referencing the location of registrant (contact) entries within the DIT. Clients SHOULD expect entries bearing the object class of 'x660Registrant' exactly one (1) level below this DN.

This is primarily used in scenarios where it is desirable for an application to self-configure for maximum efficiency in terms of registrant location, management and retrieval.

An ideal location for this attribute type is within the DSA's RootDSE.

The attribute type 'distinguishedName', as defined in Section 2.7 of [RFC4519], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.85
    NAME 'rARegistrantBase'
    DESC 'LDAP Distinguished Name of the root X.660
       registrant entry storage location in a DIT'
    SUP distinguishedName )
```

Example: "ou=Registrants,ou=X660,dc=example,dc=com"

2.1.86. 'rADirectoryModel'

The 'rADirectoryModel' attribute type allows for the storage of of an object identifier meant to declare the structural design of the DIT content pertaining to this specification.

This is primarily used in scenarios where it is desirable for an application to self-configure for maximum efficiency in terms of registration location and retrieval.

The supported values for this attribute type are as follows and correspond to Sections 3.2 and 3.3 of this specification:

- 1.3.6.1.4.1.56521.101.3.2 (twoDimensional)
- 1.3.6.1.4.1.56521.101.3.3 (threeDimensional)

An ideal location for this attribute type is within the DSA's RootDSE.

```
( 1.3.6.1.4.1.56521.101.2.1.86
    NAME 'rADirectoryModel'
    DESC 'Object Identifier meant to advertise
       the directory model governing the storage
       of X.660 LDAP entries within the DIT'
    EQUALITY objectIdentifierMatch
    SINGLE-VALUE
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
```

Example: "1.3.6.1.4.1.56521.101.3.3"

2.1.87. 'rAServiceMail'

The 'rAServiceMail' attribute type allows for the OPTIONAL assignment of one or more email addresses to an 'x660DUAConfig' entry for the purpose of email-based allocation request handling, error reporting or general support for end-users.

The attribute type 'mail', as defined in Section 2.16 of [RFC4524], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.87
   NAME 'rAServiceMail'
   DESC 'Email address used for RA-level
      registration requests, inquiries or
      problem reports'
   SUP mail )
```

Example: "ra@example.com"

2.1.88. 'rAServiceURI'

The 'rAServiceURI' attribute type allows for the OPTIONAL assignment of one or more URI values to an 'x660DUAConfig' entry for the purpose of directing users to appropriate RA endpoint for request handling.

The attribute type 'labeledURI', as defined in [RFC2079], is a super type of this attribute type.

```
( 1.3.6.1.4.1.56521.101.2.1.88
   NAME 'rAServiceURI'
   DESC 'URI, with an optional label, defining
      an appropriate RA endpoint'
```

```
        SUP labeledURI )
```

Example: "http://example.com/ra.html Registrations"

2.2. Object Classes

The following subsections describes LDAP object classes made available by this specification.

2.2.1. 'x660RootArc'

The 'x660RootArc' class is meant to define a maximum of three (3) root registrations within a DIT, per Rec. ITU-T X.660 (ISO/IEC 9834-1).

```
  ( 1.3.6.1.4.1.56521.101.2.2.1
      NAME 'x660RootArc'
      DESC 'Top-level class for entries meant to represent
         ITU-T, ISO or Joint-ISO-ITU-T root arcs as defined
         in Section A.2 of the X.660 specification'
      SUP top STRUCTURAL
      MUST ( n $ unicodeValue $ identifier )
      MAY ( registrationInformation $ additionalIdentifier $
            asn1Notation $ iRI $ currentAuthority $ subArc $
            stdNameForm $ description $ registrationURI $
            registrationCreated $ registrationModified $
            leftArc $ rightArc $ nameAndNumberForm $
            firstAuthority ) )
```

2.2.2. 'x660SubArc'

The 'x660SubArc' object class makes a collection of attribute types available for use when crafting subordinate registrations within a DIT.

```
  ( 1.3.6.1.4.1.56521.101.2.2.2
      NAME 'x660SubArc'
      DESC 'A generalized class meant to represent sub arcs
         beneath any root, as defined in X.660 Sections A.3-A.5'
      SUP top STRUCTURAL
      MUST ( n )
      MAY ( nameAndNumberForm $ firstAuthority $ finalArc $
            currentAuthority $ supArc $ subArc $ firstArc $
            topArc $ description $ registrationURI $ iRI $
            isFrozen $ discloseTo $ sponsor $ isLeafNode $
            leftArc $ rightArc $ registrationInformation $
            additionalIdentifier $ registrationCreated $
            unicodeValue $ longArc $ registrationRange $
            registrationModified $ registrationStatus $
            identifier $ stdNameForm $ asn1Notation $
            dotNotation ) )
```

2.2.3. 'x660Registrant'

The 'x660Registrant' object class allows for current, previous (first) and/or sponsorship data to be stored within an entry.

```
  ( 1.3.6.1.4.1.56521.101.2.2.3
      NAME 'x660Registrant'
      DESC 'A generalized auxiliary class for
         registrant contact information'
      SUP top AUXILIARY
      MAY ( currentAuthorityURI $ sponsorCommonName $ sponsorOrg $
            firstAuthorityPOBox $ sponsorPOBox $ sponsorLocality $
            sponsorCountryCode $ firstAuthorityURI $ description $
            firstAuthorityStartTimestamp $ sponsorStartTimestamp $
            firstAuthorityFax $ sponsorCountryName $ sponsorURI $
            sponsorTelephone $ currentAuthorityOrg $ sponsorFax $
            currentAuthorityCommonName $ firstAuthorityMobile $
            currentAuthorityTitle $ firstAuthorityCountryCode $
            currentAuthorityPostalAddress $ firstAuthorityOrg $
            currentAuthorityFax $ firstAuthorityPostalAddress $
            currentAuthorityCountryCode $ firstAuthorityState $
            currentAuthorityCountryName $ firstAuthorityEmail $
            currentAuthorityMobile $ firstAuthorityCommonName $
            firstAuthorityCountryName $ currentAuthorityState $
            currentAuthorityPostalCode $ firstAuthorityStreet $
            currentAuthorityTelephone $ currentAuthorityEmail $
            currentAuthorityLocality $ firstAuthorityLocality $
            currentAuthorityStreet $ firstAuthorityPostalCode $
            firstAuthorityTitle $ registrantID $ sponsorEmail $
            firstAuthorityEndTimestamp $ sponsorEndTimestamp $
            firstAuthorityTelephone $ currentAuthorityPOBox $
            currentAuthorityStartTimestamp $ sponsorTitle $
            sponsorStreet $ sponsorMobile $ sponsorState $
            sponsorPostalCode $ sponsorPostalAddress ) )
```

2.2.4. 'x660DUAConfig'

The 'x660DUAConfig' object class allows for the storage of so-called auto-configuration attribute types meant to guide various DUAs in their attempt to access registration and/or registrant information contained within the DIT hosted by the RA DSA. See Section 3.5 for details.

```
  ( 1.3.6.1.4.1.56521.101.2.2.4
      NAME 'x660DUAConfig'
      DESC 'Entry class to facilitate advertisement
         of optimal X.660 DUA configuration values'
      SUP top AUXILIARY
```

```
         MAY ( rADirectoryModel $ rARegistrantBase $
               rAServiceMail $ rAServiceURI $
               rARegistrationBase ) )
```

3. Directory Models and Procedures

This section offers two (2) distinct models, and some procedures, by which both directory architects and application developers SHOULD be guided in efforts to implement this specification.

Note that in various examples shown, some DNs are particularly long and are line-wrapped and indented for readability.

3.1. Naming Context and Organization Entries

In these examples, a naming context of "ou=X660, dc=example, dc=com" is used as the "suffix". Within this suffix are two (2) entries:

- "ou=OID, ou=X660, dc=example, dc=com" - Storage of all registration entries (e.g.: 'rARegistrationBase')

- "ou=Registrants, ou=X660, dc=example, dc=com" - Storage of all registrant (authority/sponsorship) entries (e.g.: 'rARegistrantBase')

Directory architects MAY choose to use models of their own design, so long as noted requirements in the following sections are satisfied.

3.2. Two-Dimensional Model

This model suggests that registrations reside as siblings within an LDAP DIT in singular, non-hierarchical locations.

This model is RECOMMENDED for small and/or sparse implementations. The three-dimensional model (See Section 3.3) may be more appropriate for larger, more robust implementations.

Use of this model is entirely at the discretion of the directory architect(s) involved. It should be noted that if users will be managing OID data directly through use of standard LDAP TUI or GUI applications, this model would seem to be more convenient as opposed to the three-dimensional model.

A DSA can advertise the use of this structural model using the following attribute type and value in a location that is easily discovered by clients (e.g.: the RootDSE [RFC4512]):

```
rADirectoryModel: 1.3.6.1.4.1.56521.101.3.2
```

3.2.1. Requirements

One requirement of this model is strict use of the 'dotNotation' attribute type, covered in Section 2.1.2. This attribute MUST be used on all non-root registrations.

Root registrations SHALL NOT bear an 'dotNotation' value, as the syntax for OIDs (see Section 3.3.26 of [RFC4517]) requires at least two (2) arcs in a given value.

Uniqueness of 'dotNotation' values within a directory structure MUST always be enforced to ensure unambiguous results. The simplest way to meet this requirement would be to adopt a DN structure based on this attribute type, as shown in Section 3.2.2.

3.2.2. Distinguished Name Convention

Because all LDAP Search Requests [RFC4511] can be conducted using a scope of singleLevel (per Section 4.5.1.2 of [RFC 4511]) below the necessary directory branch, a DN structure of hierarchical nature is wholly unnecessary. While the three-dimensional model (as shown in Section 3.3) uses the integer-based 'n' attribute type (per Section 2.1.1) to form the effective LDAP RDN of an entry, this would not be practical in this model.

The most sensible convention for DN involves use of the attribute type 'dotNotation' as shown:

```
dn: dotNotation=1.3,ou=OID,ou=X660,dc=example,dc=com
objectClass: top
objectClass: x660SubArc
n: 3
unicodeValue: Identified-Organization
dotNotation: 1.3
```

Subsequent entries, regardless of their true [X.660] hierarchical placement, manifest as sibling directory entries. For example, the addition of "deeper" arcs would be procedurally identical:

```
dn: dotNotation=1.3.6.1,ou=OID,ou=X660,dc=example,dc=com
objectClass: top
objectClass: x660SubArc
n: 1
identifier: internet
dotNotation: 1.3.6.1
```

3.2.3. Root Arc Entries

A maximum of three (3) root arcs MAY exist within the directory landscape. If one or more are created, they SHOULD be identifiable as follows:

- ITU-T (0)
- ISO (1)
- Joint-ISO-ITU-T (2)

As sibling entries, these root arcs MUST use the 'x660RootArc' class, as shown in Section 2.2.1:

```
dn: n=0,ou=OID,ou=X660,dc=example,dc=com
objectClass: top
objectClass: x660RootArc
n: 0
unicodeValue: ITU-T

dn: n=1,ou=OID,ou=X660,dc=example,dc=com
objectClass: top
objectClass: x660RootArc
n: 1
unicodeValue: ISO

dn: n=2,ou=OID,ou=X660,dc=example,dc=com
objectClass: top
objectClass: x660RootArc
n: 2
unicodeValue: Joint-ISO-ITU-T
```

Using root registrations is only useful in the two-dimensional model if the administrator wishes to organize lists of OIDs beneath their respective roots. This is likely unnecessary in implementations that are small and sparse. In larger implementations, however, this model may be convenient in situations where DIT content segmentation is in effect.

3.3. Three-Dimensional Model

This model is hierarchical by nature, providing a means for storing registrations in nested fashion, thereby reflecting the hierarchical logic of the [X.660] specification itself.

This model is RECOMMENDED for thorough or complete implementations, or implementations in which custom solutions (applications) have been tailored for this purpose. This model may be prohibitively tedious in sparse and/or small implementations.

Use of this model is entirely at the discretion of the directory architect(s) involved. It should be noted that end-users that will directly access or manage this data through standard LDAP TUI or GUI applications alone may find this model tedious, and may prefer the two-dimensional model as described in Section 3.2.

A DSA can advertise the use of this structural model using the following attribute type and value in a location that is easily discovered by clients (e.g.: the RootDSE [RFC4512]):

```
rADirectoryModel: 1.3.6.1.4.1.56521.101.3.3
```

3.3.1. Requirements

In this model, interim arcs MUST exist even if they are otherwise unnecessary.

For example, in order to add the well-known "internet" OID (1.3.6.1), directory administrators MUST ensure the following registrations exist beforehand:

```
  dn: n=1,ou=OID,ou=X660,dc=example,dc=com
  objectClass: top
  objectClass: x660RootArc
  n: 1
  identifier: iso
  unicodeValue: ISO

  dn: n=3,n=1,ou=OID,ou=X660,dc=example,dc=com
  objectClass: top
  objectClass: x660SubArc
  n: 3
  identifier: identified-organization
  unicodeValue: Identified-Organization

  dn: n=6,n=3,n=1,ou=OID,ou=X660,dc=example,dc=com
  objectClass: top
  objectClass: x660SubArc
  n: 6
  identifier: dod
```

Only once this requirement is satisfied would the administrators be able to create the desired registration, such as a registration entry for the "internet" OID, as shown in [RFC1155]:

```
  dn: n=1,n=6,n=3,n=1,ou=OID,ou=X660,dc=example,dc=com
  objectClass: top
  objectClass: x660SubArc
  n: 1
  identifier: internet
```

3.3.2. Distinguished Name Convention

Under a strict interpretation of this model, its implementation will provide a means for bidirectional resolution of registered OIDs. LDAP DNs can be deduced from OIDs, and vice versa.

This is achieved by using the 'n' attribute type (as defined in Section 2.1.1) as components in the effective LDAP DN, but in reverse order to reflect the directory hierarchy.

For example: the "internet" OID would exist as an entry with a DN as depicted below:

```
            1.3.6.1
        ----------------
```

```
           |    |    |    |
     dn: n=1, n=6, n=3, n=1, ou=OID, ou=X660, dc=example, dc=com
```

As a result, use of the 'dotNotation' attribute type becomes unnecessary unless users wish to search for an OID using an LDAP search filter.

3.3.3. Root Arc Entries

A maximum of three (3) root arcs SHOULD exist within the directory landscape. If one or more are created, they MUST be identifiable as follows:

- ITU-T (0)
- ISO (1)
- Joint-ISO-ITU-T (2)

As sibling entries, these root arcs MUST use the 'x660RootArc' class, as shown in Section 2.2.1:

```
dn: n=0,ou=OID,ou=X660,dc=example,dc=com
objectClass: top
objectClass: x660RootArc
n: 0
unicodeValue: ITU-T

dn: n=1,ou=OID,ou=X660,dc=example,dc=com
objectClass: top
objectClass: x660RootArc
n: 1
unicodeValue: ISO

dn: n=2,ou=OID,ou=X660,dc=example,dc=com
objectClass: top
objectClass: x660RootArc
n: 2
unicodeValue: Joint-ISO-ITU-T
```

Depending on the breadth and scope of an implementation, creation and use of root registrations is RECOMMENDED, but not required for every situation.

3.3.3.1. Lack of Root Arc Entries

In situations where a three-dimensional model is used, but only contains a subset of subordinate registrations, root registrations may not be necessary.

Instead, directories architects MAY choose to create a subordinate registration as a false root, and store the relevant contents there.

For example, if a directory architect only wanted to store IANA

PEN registrations that are allocated below the OID 1.3.6.1.4.1, the relevant DIT entries could manifest as follows:

```
     dn: dotNotation=1.3.6.1.4.1,ou=OID,ou=X660,dc=example,
      dc=com
     objectClass: top
     objectClass: x660SubArc
     identifier: enterprise
     additionalIdentifier: enterprises
     registrationURI: https://www.iana.org/assignments/enterprise
      -numbers

     dn: n=56521,dotNotation=1.3.6.1.4.1,ou=OID,ou=X660,
      dc=example,dc=com
     objectClass: top
     objectClass: x660SubArc

     ... other sibling PEN entries ...
```

Alternatively, if the above non-standard DN syntax is undesirable, directory architects MAY choose to honor the three-dimensional model DN syntax, but limit creation of new entries to those that are direct ancestors of the intended subset. In this situation use of a root registration is necessary, but only relevant entries would need to be created.

Keeping with the above scenario, the relevant DIT entries could manifest as follows:

```
     dn: n=1,ou=OID,ou=X660,dc=example,dc=com
     objectClass: top
     objectClass: x660RootArc
     n: 1
     unicodeValue: ISO

     dn: n=3,n=1,ou=OID,ou=X660,dc=example,dc=com
     objectClass: top
     objectClass: x660SubArc
     n: 3
     unicodeValue: Identified-Organization

     dn: n=6,n=3,n=1,ou=OID,ou=X660,dc=example,dc=com
     objectClass: top
     objectClass: x660SubArc
     n: 6
     identifier: dod

     dn: n=1,n=6,n=3,n=1,ou=OID,ou=X660,dc=example,dc=com
     objectClass: top
     objectClass: x660SubArc
     n: 1
     identifier: internet
```

```
     dn: n=4,n=1,n=6,n=3,n=1,ou=OID,ou=X660,dc=example,dc=com
     objectClass: top
     objectClass: x660SubArc
     n: 4
     identifier: private

     dn: n=1,n=4,n=1,n=6,n=3,n=1,ou=OID,ou=X660,dc=example,dc=com
     objectClass: top
     objectClass: x660SubArc
     n: 1
     identifier: enterprise
     additionalIdentifier: enterprises
     registrationURI: https://www.iana.org/assignments/enterprise
      -numbers

     dn: n=56521,n=1,n=4,n=1,n=6,n=3,n=1,ou=OID,ou=X660,dc=example,
      dc=com
     n: 56521
     objectClass: top
     objectClass: x660SubArc

     ... other sibling PEN entries ...
```

3.4. Registrant Information

Directory architects MAY choose to store registrant information in one of two main ways:

- Store sponsor or authority registrant information within registration entries themselves, or ...

- Store sponsor and/or authority registrant information within dedicated entries, and reference the DNs of these entries via the 'currentAuthority', 'sponsor' and/or 'firstAuthority' attribute types assigned to registrations

The nature of these contacts will typically encompass three (3) kinds of entities:

- An individual
- An organization, institution or working group
- A document (e.g.: a standard or draft)

3.4.1. Use of Collective or Virtual Attributes

In situations involving particularly large and/or distributed LDAP DIT contexts, directory architects MAY choose to extend certain attribute types within this specification to allow for Collective Attributes [RFC3671] support, or some other proprietary feature for the facilitation of so-called "virtual attributes" to allow for a broad-scale and low-cost association of attribute values across entire pools of entries.

While these concepts are out of scope for this specification, a few attribute types defined in Section 2 MAY be used in situations that involve one or more specific registrants being assigned to a very large number of registrations, or an entire subtree in perpetuity:

- 'currentAuthority', defined in Section 2.1.28
- 'firstAuthority', defined in Section 2.1.46
- 'sponsor', defined in Section 2.1.65

3.4.2. Examples

3.4.2.1. Combined Registration and Registrant Entries

This is a basic two-dimensional example entry comprised of both registration and registrant attribute types.

```
dn: dotNotation=1.3.6.1.4.1.56521,n=1,ou=OID,ou=X660,
 dc=example,dc=com
objectClass: x660SubArc
objectClass: x660Registrant
objectClass: top
currentAuthorityPostalAddress: 1 Fake St$Anywhere$CA$92262
currentAuthorityCommonName: Jesse Coretta
currentAuthorityEmail: jesse.coretta@example.com
currentAuthorityMobile: +11234567890
dotNotation: 1.3.6.1.4.1.56521
n: 56521
```

This is a basic three-dimensional example entry of the same design.

```
dn: n=56521,n=1,n=4,n=1,n=6,n=3,n=1,ou=OID,ou=X660,
 dc=example,dc=com
objectClass: x660SubArc
objectClass: x660Registrant
objectClass: top
currentAuthorityPostalAddress: 1 Fake St$Anywhere$CA$92262
currentAuthorityCommonName: Jesse Coretta
currentAuthorityEmail: jesse.coretta@example.com
currentAuthorityMobile: +11234567890
n: 56521
```

3.4.2.2. Dedicated Registrant Entries

This is a basic example of a single authority-based contact entry.

Please note that use of the 'organizationalRole' object class (per Section 3.10 of [RFC4519]) is purely incidental here. Directory architects MAY opt for another STRUCTURAL object class.

```
   dn: registrantID=draft-coretta-x660-ldap,ou=Registrants,
    ou=X660,dc=example,dc=com
   registrantID: draft-coretta-x660-ldap
   cn: draft-coretta-x660-ldap
   objectClass: organizationalRole
   objectClass: x660Registrant
   objectClass: top
   currentAuthorityPostalAddress: 1 Fake St$Palm Springs$
    CA$92262
   currentAuthorityCommonName: Jesse Coretta
   currentAuthorityEmail: jesse.coretta@icloud.com
   currentAuthorityMobile: +11234567890
   currentAuthorityStartTimestamp: 20200229134901Z
```

In cases where multiple distinct individuals or addresses are used, they can all be combined into a single entry:

```
   dn: registrantID=draft-coretta-x660-ldap,ou=Registrants,
    ou=X660,dc=example,dc=com
   registrantID: draft-coretta-x660-ldap
   cn: draft-coretta-x660-ldap
   objectClass: organizationalRole
   objectClass: x660Registrant
   objectClass: top
   currentAuthorityPostalAddress: 1 Fake St$Palm Springs$
    CA$92262
   currentAuthorityCommonName: Jesse Coretta
   currentAuthorityEmail: jesse.coretta@icloud.com
   currentAuthorityMobile: +11234567890
   currentAuthorityStartTimestamp: 20200229134901Z
   sponsorOrg: Sponsor, Co.
   sponsorEmail: sponsor@example.com
   sponsorMobile: +11234560987
   sponsorStartTimestamp: 20010104120144Z
   sponsorPostalAddress: 456 Fugazzi Ln$Anywhere$CA$92262
   firstAuthorityPostalAddress: 1 Fake St$Palm Springs$
    CA$92262
   firstAuthorityCommonName: Jesse Coretta
   firstAuthorityEmail: jesse.coretta@icloud.com
   firstAuthorityMobile: +11234567890
   firstAuthorityStartTimestamp: 20200229134901Z
```

Keeping with the example registration described in Section 3.4.2.2, the three-dimensional registration would manifest as follows:

```
   dn: n=56521,n=1,n=4,n=1,n=6,n=3,n=1,ou=OID,ou=X660,
    dc=example,dc=com
   objectClass: x660SubArc
```

```
      objectClass: top
      currentAuthority: registrantID=draft-coretta-x660-ldap,
       ou=Registrants,ou=X660,dc=example,dc=com
      n: 56521
```

3.5. DUA Configuration

Two (2) distinct methods for client-side configuration are shown in this section. Users and directory architects alike MAY opt for one over the other, depending on the situation.

Regardless of the style of configuration, DUA observance of the following constants is strongly RECOMMENDED when leveraging this specification in the manner intended:

- An effective registration search base MUST be made available
- The condition of a missing or undefined registrant search base MUST indicate that NO authority or sponsorship information is stored or advertised
- The condition of a registration search base that is equal to the registrant search base MUST indicate the use of so-called Combined Registration and Registrant entries
- The condition of a registration search base that is not equal to the registrant search base MUST indicate the use of so-called Dedicated Registrant entries

3.5.1. Manual Configuration

Manual configuration MAY be used if:

- Advertisement of configuration information, as shown in Section 3.5.2, is unavailable, or ...
- The information advertised is somehow unreliable, or ...
- The DUA is not optimized for this specification

Assuming one of the above applies, the following conceptual runtime settings are sufficient for basic interaction with the DSA acting as an RA:

- Directory model object identifier (see Sections 3.2 and 3.3)
- Distinguished name(s) used for registration and registrant LDAP search bases

3.5.2. Automatic Configuration

Any DUA optimized or designed with this specification in mind SHOULD support the automatic retrieval of the following attribute types:

- 'rARegistrationBase', as defined in Section 2.1.84
- 'rARegistrantBase', as defined in Section 2.1.85
- 'rADirectoryModel', as defined in Section 2.1.86
- 'rAServiceMail', as defined in Section 2.1.87
- 'rAServiceURI', as defined in Section 2.1.88

Limitations pertaining to the artificial addition of attributes to the RootDSE are implementation-specific, and thus the DUA SHOULD allow manual configuration, as shown in Section 3.5.1, as a fallback measure.

There are two techniques a DUA MAY use to locate the above attribute types:

- Check the RootDSE advertised by the DSA(s)
- If a RootDSE check fails, fallback to a broad-level search for a single LDAP entry bearing the 'x660DUAConfig' object class, as shown in Section 2.2.4

3.5.2.1. Examples

This subsection shows examples for storing values assigned to the attribute types mentioned in Section 3.5.2. Note these may not be complete and legal LDAP entries, as their full contents may be omitted for brevity.

RootDSE alterations may manifest as follows for an RA storing a three-dimensional [X.660] DIT structure with dedicated registrant entries:

```
dn:
objectClass: x660DUAConfig
rADirectoryModel: 1.3.6.1.4.1.56521.101.3.3
rARegistrationBase: ou=OID,ou=X660,dc=example,dc=com
rARegistrantBase: ou=Registrants,ou=X660,dc=example,dc=com
```

Alternatively, for an RA storing a two-dimensional [X.660] DIT structure with combined registration and registrant entries:

```
dn:
objectClass: x660DUAConfig
rADirectoryModel: 1.3.6.1.4.1.56521.101.3.2
rARegistrationBase: ou=OID,ou=X660,dc=example,dc=com
rARegistrantBase: ou=OID,ou=X660,dc=example,dc=com
```

If use of the RootDSE is not feasible for storing this information, but automatic configuration is still desired, directory architects MAY opt to store entries similar to those above within another DIT, however the drawback to this approach is that the DUA (or its user) would require foreknowledge of the relevant suffix.

```
dn: dc=example,dc=com
objectClass: x660DUAConfig
rADirectoryModel: 1.3.6.1.4.1.56521.101.3.3
rARegistrationBase: ou=OID,ou=X660,dc=example,dc=com
```

```
   rARegistrantBase: ou=Registrants,ou=X660,dc=example,dc=com
```

Depending on the nature of the RA service indicated, the creation or modification of registrations MAY require approval before such changes can be committed to the directory. As such, a directory architect MAY opt to advertise an appropriate email address and/or URI as shown below.

DUAs or services optimized for this specification may be required to alter their behavior as it relates to write operations against the RA DSA if the 'rAServiceMail' and/or 'rAServiceURI' attribute types are present:

```
dn:
objectClass: x660DUAConfig
rADirectoryModel: 1.3.6.1.4.1.56521.101.3.3
rAServiceMail: ra@example.com
rAServiceURI: https://www.example.com/ra.html
rARegistrationBase: ou=OID,ou=X660,dc=example,dc=com
rARegistrantBase: ou=Registrants,ou=X660,dc=example,dc=com
```

3.6. Spatial Orientation and Navigation

Some of the more complete RA services, whether public or private, may offer a simple interface to facilitate intuitive contiguous movement between logically adjacent registrations in terms of "up", "down", "left" and "right"

Depending on the needs of the intended audience, as well as the manner in which this specification is adopted, this can be an exceptionally difficult feature to implement.

The main concerns on this topic are summarized as follows:

- DSA resource utilization
- DIT content model, complexity and footprint

The functionality discussed in this subsection is NOT officially defined in [X.660], but is present in this specification so as to mitigate or overcome certain challenges associated with the use of directory services in this context.

3.6.1. Client Capabilities

In terms of this specification, a client will be most effective when its implementer, or developer, takes care to understand the nature of the DSA(s) it may use, as well as the DIT(s) in question.

Assuming a sound configuration is in place, per Section 3.5, a given client needs to determine a means for the following actions by OID alone:

- Locating individual registrations
- Traversing surrounding registrations (spatially)

3.6.1.1. Locating Individual Registrations

In sufficiently thorough three-dimensional implementations of this specification, it is not uncommon to see an absence of literal 'dotNotation' values for registrations. The reason for this, as shown in Section 3.3.2, is that the DN syntax alone is sufficient for the "storage" of an OID. The drawback, however, is that the client is now required to perform the following abstract actions to locate (or extrapolate) an OID to an entry:

- Reverse OID (1.3.6 becomes 6.3.1)
- Transmute reversed OID to a partial RDN sequence using the 'n' attribute type (n=6,n=3,n=1)
- Combine the appropriate registration suffix DN and RDNs (n=6,n=3,n=1,ou=OID,ou=X660,dc=example,dc=com)

Procedurally, this is a simple operation and is beneficial when an application is dealing with a particularly busy DSA, as a baseObject LDAP Search Request (per Section 4.5.1.2 of [RFC4511]) will always return either one (1) or zero (0) results. This would result in lower DSA utilization overall when compared to broad search request for a 'dotNotation' value, but because the composed DN is "inferred", subsequent requests come with the risk of a 'noSuchObject (32)' error being returned (per Appendix A, Section 2 of [RFC4511]) should the entry not exist at said specified DN.

However, in two-dimensional implementations of this specification, as shown in Section 3.2.2, a literal 'dotNotation' value exists which the client can query one of two ways:

- If 'dotNotation' is used as a component in the relevant DN syntax, call the extrapolated DN, or ...
- Perform broad filter-based searches for the desired OID as a literal 'dotNotation' (e.g.: '(dotNotation=<oid>)')

The former option requires extra effort on part of the client, while the latter option will require more DSA resources, and would require the attribute type be populated for the given registration.

3.6.1.2. Traversing Adjacent or Ancestral Registrations

OIDs exist in lexically hierarchical relation to one another, and with no absolutely no limits imposed on overall OID "depth" nor on the magnitude of any single NumberForm [X.680] component (so long as the magnitude is zero (0) or more).

An effective client intended for use in situations involving this specification SHOULD have the capability to locate all applicable adjacent entries without extraordinary effort. The subsections that follow will cover attribute types and procedures to that end.

3.6.1.2.1. Use of the 'supArc' and 'topArc' Attribute Types

The 'supArc' and 'topArc' attribute types are used to describe the vertical relationships between registrations in an ancestral sense. One need only perform a baseObject LDAP Search Request (per Section 4.5.1.2 of [RFC4511]) upon the referenced DN to obtain the superior entry in question.

If a registration possesses a DN value for the 'supArc' attribute type, the DN MUST belong to the registration that is the immediate superior (parent).

Similarly, if a registration possesses a DN value for the 'topArc' attribute type, this DN MUST belong to the absolute root registration entry within the directory. In this case, the registration entry MUST ONLY be one (1) of itu-t(0), iso(1) or joint-iso-itu-t(2).

A directory architect MAY choose to leverage either of these types virtually, depending on the situation. For example, if a certain registration possesses tens of thousands of subordinate registration entries, it may be prudent to assign the 'topArc' and/or 'supArc' attribute types to those registrations through the use of Collective Attribute [RFC3671] support, or some other proprietary means for the assignment of virtual values to a vast number of entries. Both of these concepts are out of scope for this document.

If a registration does not possess a value for either of the above attribute types, this is indicative of one of these conditions:

- Registration is an 'x660RootArc'-based entry, and thus no such superior registrations could ever possibly exist, or ...
- The DSA in question is intended only to host a subset of the OID spectrum, and any superior references would violate such constraints, or ...
- Client is expected to extrapolate the appropriate superior registration using its own means

Assuming the client is expected to extrapolate the logically superior registration, this can be handled in one of two ways:

- If a three-dimensional directory model is indicated, and the DN syntax described in Section 3.3.2 is in use, the client should attempt to truncate the leaf-node RDN to achieve its superior (parent) DN, or ...
- If a two-dimensional directory model is indicated, and the DN syntax described in Section 3.2.2 is in use, the client should attempt to read the literal 'dotNotation' value from the entry DN, truncate the value's leaf-node (right-hand) OID component, and utilize the remaining contents for the basis of subsequent

entry location

In the former (three-dimensional) case, consider the following example, in which the leaf-node (n=6) is truncated to obtain the DN of the superior registration:

- n=6,n=3,n=1,ou=OID,ou=X660,dc=example,dc=com
- n=3,n=1,ou=OID,ou=X660,dc=example,dc=com (parent)

In the latter (two-dimensional) case, consider the following example, in which the 'dotNotation' RDN value is truncated to extrapolate the DN of the superior registration:

- dotNotation=1.3.6,ou=OID,ou=X660,dc=example,dc=com
- dotNotation=1.3,ou=OID,ou=X660,dc=example,dc=com

3.6.1.2.2. Use of the 'subArc' Type

The 'subArc' attribute type is intended to hold one (1) or more LDAP DNs for the purpose of enumerating all subordinate registrations that reside beneath a given registration, or to reference the (lexically) first subordinate registration only.

If a registration does not possess a value for 'subArc', this is indicative of one of the following conditions:

- No subordinate registration manifest exists, and management of such elements is wholly up to the client, or ...
- No subordinate registrations exist at this time, or ...
- Subordinate registrations exist, but are not visible to the indicated identity (e.g.: access control)

If only a single 'subArc' value exists for the registration, this can mean one of the following:

- There are a very large number of subordinate registrations, and only the first registration is referenced, or ...
- There is only one (1) subordinate registration

```
dn: n=999,n=2,ou=OID,ou=X660,dc=example,dc=com
objectClass: x660SubArc
identifier: example
subArc: n=0,n=999,n=2,ou=OID,ou=X660,dc=example,dc=com
```

If two (2) or more 'subArc' values exist for the registration, this SHOULD be interpreted as a complete manifest of subordinate entries. As such, costly enumeration of registrations (through a singleLevel LDAP Search Request, per Section 4.5.1.2 of [RFC4511]) should not be necessary.

```
     dn: n=999,n=2,ou=OID,ou=X660,dc=example,dc=com
     objectClass: x660SubArc
     identifier: example
     subArc: n=0,n=999,n=2,ou=OID,ou=X660,dc=example,dc=com
     subArc: n=1,n=999,n=2,ou=OID,ou=X660,dc=example,dc=com
     ... many DNs omitted for brevity ...
     subArc: n=200,n=999,n=2,ou=OID,ou=X660,dc=example,dc=com
```

If the given 'subArc' manifest is perceived to be exceedingly large, and DSA utilization and/or performance are concerns, it is strongly RECOMMENDED the client application support the use of a local cache to prevent repeated and wasteful calls of certain large entries. This concept is out of scope for this document.

3.6.1.2.2.1. Awareness of Subordinate Registration Constraints

Any DUA or service interacting with RAs based upon this specification for the purpose of enumerating and/or allocating registrations MUST always check for the presence of definitions of both the 'isLeafNode' attribute type (per Section 2.1.14) and the 'isFrozen' attribute type (per Section 2.1.15) before any further subordinate traversals or allocations may take place beneath a given registration.

If an 'isLeafNode' value of "TRUE" is present for a registration, the requesting entity SHALL NOT enumerate any subordinate entries for any reason, nor will any new subordinate allocations take place.

If an 'isFrozen' value of "TRUE" is present for a given registration, but 'isLeafNode' is "FALSE" or otherwise UNDEFINED, the requesting entity SHOULD enumerate any subordinate entries already present, but SHALL NOT create any new registrations.

A value of "TRUE" for 'isLeafNode' attribute type SHALL ALWAYS imply an equal value for 'isFrozen'.

Neither of these attribute types are meant for use in access control contexts. Instead, see the 'registrationStatus' attribute type (per Section 2.1.13), as well as the associated remarks in Section 5, for information on security-specific restrictions relating to subordinate registration access and privileges.

3.6.1.2.3. Use of Sibling Adjacency Attribute Types

Within the confines of this specification, sibling registration adjacency can be depicted as follows:

```
                         current registration
                             (2.999.140)
                                  |
      << antecedent/left          V         subsequent/right >>
               +---+--------+---+---+---+--------+---+
               | 0 |........|139|140|141|........|200|
               +---+--------+---+---+---+--------+---+
                 |            \       /            |
                 farthest      nearest      farthest
```

In order to effectively express such horizontal relationships in a logical hierarchical data structure, and to foster a low-cost means for ascertaining these relationships, four (4) DN-based attribute types are made available:

- 'leftArc', as defined in Section 2.1.22
- 'firstArc', as defined in Section 2.1.23
- 'rightArc', as defined in Section 2.1.24
- 'finalArc', as defined in Section 2.1.25

The 'leftArc' and 'rightArc' attribute types are assigned to entries in a manual fashion, as the relationships are globally unique in that no single registration will share the same "left" and "right" sibling combination as any other registration. As such, the nature of these two (2) attribute types precludes use of virtual value assignments.

The 'firstArc' and 'finalArc' attribute types MAY be assigned in one (1) of the following three (3) ways:

- In literal fashion (manually), or ...
- Through the use of Collective Attribute [RFC3671] support, or ...
- Through the use of an implementation-specific attribute value virtualization feature

Any given sibling registration pool, regardless of its size, will always share the same 'firstArc' and 'finalArc' attribute values for every registration contained therein. As such, Directory architects MAY opt to leverage or otherwise extend these attribute types for virtualization purposes, a topic out of scope for this document.

If these four (4) attribute types are not defined for a registration, use of an ordered 'subArc' manifest derived from the registration's parent, OR a (costly) singleLevel search (per Section 4.5.1.2 of [RFC4511]) below the superior (parent) registration, are the only other feasible ways to ascertain sibling adjacency. For more details on this topic, see Section 3.6.1.2.2.

Another practical use for these attribute types is in case a finite 'registrationRange' value exists in a range of non-contiguous sibling entries. Use of these attribute types may negate the need for costly range "size checks". See Section 2.1.12 for more details.

In cases where a collection of sibling entries is prone to continued growth, directory architects are advised to maintain the effective 'finalArc' value(s) regularly, if defined at all. Such growth will continue in right-handed (subsequent) fashion, and thus the so-called "final registration" will shift after a time, rendering a stale value inaccurate in nature.

3.7. DSA Resource Utilization and Administrative Costs

Much of the previous subsection advocated the use of low-cost client driven methods for finding and circumnavigating registrations using only baseObject-scoped LDAP Search Requests (per Section 4.5.1.2 of [RFC4511]). This section offers some considerations relating to the DSA's relation to these concepts.

Directory architects who choose to adopt some or all of the precepts defined in this specification will inevitably encounter a situation in which they're forced to consider one or more of the following:

- Overall size of DIT is a concern (in terms of bytes and/or entries)
- Effort required to maintain DIT content for all adopted attribute types is not feasible
- Client-driven resource utilization is problematic

The following subsections covers these concerns from a variety of standpoints.

3.7.1. Literal vs. Composite Values

Consider the following scenario:

A directory architect adopts certain elements from this specification in minimalistic three-dimensional fashion. In this case, only the 'n', 'unicodeValue' and 'identifier' attribute types are maintained for any given registration. This is due to the prohibitively costly and/or time-consuming nature of maintaining any other attributes.

A user consuming this information wants to obtain 'iRI' as well as 'asn1Notation' values for any given registration. Because these attribute types are not maintained in literal fashion, this forces the user to traverse the directory through multiple individual LDAP baseObject Search Requests (per Section 4.5.1.2 of [RFC4511]), so as to derive the needed component values.

The drawbacks of this strategy are as follows:

- Tedious for the user
- Increased number of requests to DSA

- Extrapolation or prediction of composite values is not always possible

The merits of this strategy are as follows:

- Smaller data footprint
- Lower overall data maintenance costs

There are several attributes that, in some form or another, contain data that MAY reside in another attribute assigned to the same entry.

- 'identifier' (a.k.a.: 'nameForm') and 'n' (a.k.a.: 'numberForm') are the sole components of 'nameAndNumberForm'
- 'nameAndNumberForm' is (often) the iterative component of the 'asn1Notation' type, with outer "curly brace encapsulation" (e.g.: "{...}")
- 'unicodeValue' and 'n' are (often) the components of 'iRI', using a solidus (/) prefix and delimitation scheme
- 'n' is the iterative component of 'dotNotation', using a dot (.) delimitation scheme comprised of two (2) or more elements
- 'additionalIdentifier' is a multi-valued attribute type populated solely by one or more non-primary 'identifier' (or 'nameForm') values

With such cases in mind, directory architects are advised to plan attribute support carefully, as well as manage expectations of the end-user.

Directory architects SHOULD populate some of the above attribute types in cases where their true value may not be possible to predict or extrapolate. One example of this is the 'iRI' value for the OID assigned to 'asn1' (2.1). While its logical 'unicodeValue'-based path might produce an effective value of "/Joint-ISO-ITU-T/ASN.1", which is not its true 'iRI' - the correct value is "/ASN.1".

3.7.2. Subtree Search Operations

Use of wholeSubtree, or Subtree, LDAP Search Operations (as defined in Section 4.5.1.2 of [RFC4511]) MAY be required in cases where a registration lookup is needed, but only a single abstract piece of information is known by the user. Frequently sought-after attribute types of this nature include:

- 'nameAndNumberForm'
- 'identifier'
- 'additionalIdentifier'
- 'iRI'
- 'longArc'
- 'stdNameForm'
- 'unicodeValue'

Given an effective LDAP Search baseObject (per Section 4.5.1.1 of [RFC4511]), and depending on the nature of the information requested, such operations may be especially expensive. Should these operations be deemed necessary, directory administrators are STRONGLY advised to take appropriate performance-level actions to ensure fast request responses. This may be achieved through a variety of DSA-level optimizations -- concepts that are out of scope for this document.

It may also be prudent for directory administrators to place limits on the number of entries that can be returned as a result of a given request. This is especially true if the RA DSA supports substring search filters (e.g.: '(identifier=a*)') that would almost certainly return many results.

4. IANA Considerations

There are no requests to IANA in this document.

5. Security Considerations

This document focuses on providing flexible directory models and LDAP schema elements in order to serve OID-related entries, and to allow for an LDAP-based means for OID resolution. No assumptions are made with regards to confidentiality of entries.

If some or all of the [X.660] data in a given directory is sensitive in nature, directory architects MUST take appropriate steps to secure this information. Although such concepts are out of scope for this document, specific attribute types were made available in Section 2 to aid directory architects in this regard.

- 'discloseTo', defined in Section 2.1.26, allows the declaration of one or more LDAP DNs authorized to read the necessary content
- 'registrationStatus', defined in Section 2.1.13, facilitates access control evaluations of subtrees with a top-level parent that is assigned a value of "private"
- 'currentAuthority' and/or 'sponsor', defined in Sections 2.1.28 and 2.1.65 respectively, MAY be used to refer to identities that are authorized to allocate new registrations, as well to make certain changes to existing ones

Directory architects SHOULD employ custom attribute types and/or an enhanced DSA configuration for access control purposes if the above attribute types are unsuitable in some way.

5.1. Modification Identity Obfuscation

If the operational attributes 'creatorsName' and/or' 'modifiersName' (as defined in Sections 3.4.1 and 3.4.3 of [RFC4512] respectively) are exposed, directory architects MAY opt to use Proxy Authorization Controls, per [RFC4370], to allow an asserted (proxied) identity to effect the necessary registration changes without exposing the true authentication identity.

An appropriate proxy identity could be the distinguished name of the registrant referenced by the 'currentAuthority' or 'sponsor' types, or even a given registration entry itself. In either case, this will require [RFC4370] functionality, a topic that is well out of scope for this document.

5.2. Registrant Privacy

Some public RA services are known to offer privacy options that are meant to allow redaction of contact-related attributes such as for telephone numbers and/or email and postal addresses from public view.

In such cases, the directory architect(s) in question SHOULD take the time to set appropriate security and access control mechanisms so as to protect the data from undesirable entities, such as spammers. This MAY include methodologies such as:

- Requiring account registration and authentication to access such personal data via the RA DSA
- Limiting disclosure of personal data to administrative personnel and the relevant registrant only
- Requiring deliberate baseObject LDAP Search requests only (per Section 4.5.1.2 of [RFC4511]), which would result in registrant entries only being reachable by singular references through an OID registration

Overall, the public hosting of any contact information, regardless of its nature, is an enormous responsibility that SHOULD NEVER be taken lightly.

6. References

6.1. Normative References

[RFC2079] Smith, M., "Definition of an X.500 Attribute Type and an Object Class to Hold Uniform Resource Identifiers (URIs)", RFC 2079, January 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4512] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Directory Information Models", RFC 4512, June 2006.

[RFC4517] Legg, Ed., S., "Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules", RFC 4517, June 2006.

[RFC4519] Sciberras, Ed., A., "Lightweight Directory Access Protocol (LDAP): Schema for User Applications", RFC 4519, June 2006.

[RFC4524] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): COSINE LDAP/X.500 Schema", RFC 4524, June 2006.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", RFC 8174, May 2017.

[X.660] International Telecommunication Union - Telecommunication Standardization Sector, "General procedures and top arcs of the international object identifier tree", ITU-T X.660, July 2011.

[X.680] International Telecommunication Union - Telecommunication Standardization Sector, "Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T X.680, July 2002.

6.2. Informative References

[RFC1155] Rose, M., "Structure and Identification of Management Information for TCP/IP-based Internets", RFC 1155, May 1990.

[RFC3296] Zeilenga, K., "Named Subordinate References in Lightweight Directory Access Protocol (LDAP) Directories", RFC 3296, July 2002.

[RFC3671] Zeilenga, K., "Collective Attributes in the Lightweight Directory Access Protocol (LDAP)", RFC 3671, December 2003.

[RFC4370] Weltman, R., "Lightweight Directory Access Protocol (LDAP): Proxy Authorization Control", RFC 4370, February 2006.

[RFC4510] Zeilenga, K., Ed., "Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map", RFC 4510, June 2006.

[X.501] International Telecommunication Union - Telecommunication Standardization Sector, "The Directory: Models", ITU-T X.501, October 2016.

[PRIVATE] IANA, "Private Enterprise Numbers", https://www.iana.org/assignments/enterprise-numbers.

Author's Address

Jesse Coretta
Palm Springs, CA 92262
United States

Email: jesse.coretta@icloud.com