INTERNET-DRAFT
Document: draft-hall-ldap-idn-00.txt
Expires: January, 2004
Category: Standards-Track

Eric A. Hall
June 2003

LDAP Schema Extensions for Internationalized Domain Names

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC 2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Copyright Notice

Abstract

This document defines schema and behavioral rules which are necessary to fully accommodate the use of internationalized domain names as UTF-8 sequences in LDAP. Specifically, this document defines an internationalized domain component attribute, email address attribute, URI syntax, and several related object classes, and also discusses their usage considerations.

Table of Contents

1. Background and Overview

Although the LDAPv3 [RFC3377] service is explicitly capable of transferring characters from the Universal Character Set (UCS) [ISO10646] which have been encoded into eight-bit sequences with UTF-8 [RFC2279], the core LDAP schema and service elements which explicitly carry domain names as structured data are syntactically restricted to a subset of seven-bit character codes.

This restriction is due to historical limitations with the domain names themselves, although recent standards-track activity towards defining internationalized domain names and their usage has made these universal restrictions somewhat inappropriate and excessive, particularly in those cases where LDAP systems are required to use a UTF-8 form of the internationalized domain names directly.

Although LDAP applications are free to define whichever attributes and syntaxes they may need for their own internal use (including any private attributes related to domain names), there is also a

need for internationalized versions of the common domain-related schema and service elements to be made available for those applications. As such, this document defines internationalized versions of those common elements for the benefit of those applications, and also discusses their usage considerations.

Specifically, this document defines internationalized forms of the domainComponent attribute and its associated object classes, the mail attribute and a related object class, and the LDAP URL. Through the judicious use of these elements, LDAP applications can take full advantage of the "native" representation of internationalized domain names, while also using the legacy attributes to ensure backwards-compatibility with applications that still require the legacy form of those domain names.

2. Prerequisites and Terminology

Readers of this specification are expected to be familiar with LDAPv3 [RFC3377], IDNA [RFC3490], and UTF-8 [RFC2279].

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

3. Validation and Conversion

Internationalized domain names are expected to be validated before they are used, wherever this is possible. This section discusses where that validation should occur, and also defines the validation algorithm that should be used at those junctures.

3.1. Processing Junctures

The specifications which currently govern internationalized domain names do not define specific syntax rules for those domain names, but instead define functions for encoding and decoding the domain names into ASCII [ASCII] compatible sequences. This effectively means that the contents of an internationalized domain name element cannot be validated with syntax rules, but instead must be validated through the use of local function calls.

Since existing systems have historically relied upon syntax rules to determine validity, those systems cannot be readily expected to perform these function calls. In order to accommodate these systems and avoid problems which may otherwise occur, this specification does not imposes any formal syntax rules on the

elements themselves, but instead requires that internationalized domain names be validated before the data is stored or transferred, if the opportunity presents itself. Essentially, this approach makes the internationalized domain name elements transparent, capable of carrying any eight-bit data, while requiring that the party which generates the data perform the necessary function calls.

Specifically, internationalized domain names are expected to be validated at the following points:

* Operators of LDAP servers SHOULD validate the data before it is added to the server, if possible. For example, if the operator of a server creates a new naming context that uses the inetDomainComponent attribute, then that operator SHOULD ensure that the internationalized domain name labels contained therein are syntactically valid prior to creating the naming context. Similarly, if an application generates LDIF output which contains internationalized domain names, the script which generates that output SHOULD ensure that those domain names are syntactically valid.

* Client applications SHOULD validate the data whenever that data is originally created, if possible. For example, if an application allows the creation of inetMail attribute values with foreknowledge of that attribute's usage, the application SHOULD validate the attribute value prior to submitting the data to the server for storage.

* Client applications SHOULD validate any data they extract from LDAP prior to passing that data to an external application. For example, if an user-facing application will pass an inetMail attribute value to an email client, the LDAP application SHOULD validate the attribute value prior to sending it on, if possible.

* The client SHOULD use the most appropriate form of the data when passing that data to another application, either by using the most-appropriate of the attributes, or by down-converting the internationalized form if necessary. In those cases where the client has no a priori knowledge of the recipient application's capabilities, the ASCII compatible form MUST be used.

3.2. Processing Algorithm

Wherever an internationalized domain name is to be validated, the input domain name MUST use the conversion algorithm defined in this section. These rules apply to any domain name which is stored in any internationalized domain name element, regardless of the contents of the input. For example, these rules apply to domain names which only contain printable ASCII characters, domain names which appear to already be encoded into IDNA, and domain names which contain UCS character codes.

In general terms, the validation process requires that every domain name which is to be stored in an internationalized domain name element undergo a two-part conversion, with the input first being reduced to its canonical IDNA-encoded form, and then being expanded into its UTF-8 encoded UCS form. This process ensures that the domain name has been validated as a semantically correct IDNA sequence, and that the resulting internationalized domain name has been properly normalized into its canonical form.

The full process is as follows:

a. Unless otherwise explicitly defined, disable the UseSTD3ASCIIRules IDNA flag and enable the AllowUnassigned IDNA flag, thereby permitting the broadest range of character codes to be used.

b. If the input domain name terminates with a Full-Stop character (0x2E) but does not consist of a single Full-Stop character alone, remove the trailing Full-Stop character from the input.

c. Perform the "ToASCII" conversion operation specified in [RFC3490]. This step will reduce the input domain name to the canonical IDNA-compatible form, thus ensuring that the input data can be properly normalized when it is reconstructed, and also ensuring that any subsequent conversions back into the ASCII-compatible form will result in predictable and legitimate domain names.

d. Perform the "ToUnicode" conversion operation specified in [RFC3490] against the output from step 3.2.a above. This step will convert the ASCII-compatible sequence into a sequence of UCS code-point values.

e. Encode the output from step 3.2.d into UTF-8.

Note that the UseSTD3ASCIIRules and AllowUnassigned IDNA flags MUST be set to their most liberal settings by default, and are not to be used unless the underlying application-specific usage of a domain name is known to require usage to the contrary.

By following these rules, the internationalized domain names which appear in the available elements will always be valid, and will always be usable by applications which specifically make use of the elements, while those systems which do not make explicit use of these elements but which may inadvertently pass the internationalized domain names to other applications will not be exposed to any potential risks which may have been otherwise caused from malformed data.

Also note that these requirements are significantly more stringent than the requirements for validating legacy domain names in the legacy elements, and also apply to legacy-compatible domain names which are stored in the internationalized elements. For example, the existing domainComponent and mail attributes do not require data to be validated against the known syntax rules for domain names and email addresses, but instead simply limit the range of character codes to a relatively small subset, while the rules defined above will result in the same canonical input having a stricter actual syntax.

3.3. Escape Syntax

Certain applications allow for the use of "unusual" characters or octet values which are not typically associated with traditional domain names, but which must be preserved in order for the associated applications to function properly. For example, an application-specific domain name may contain an Underscore character (0x5F) or a Space character (0x20), or may contain a "raw" octet value such as 0xC0 and which cannot be treated as a UCS character code during normalization routines (otherwise the corresponding UCS character code value would be interpreted and lowercased, thus destroying the actual octet value).

In order to ensure that these kinds of values are properly preserved, a formal escape syntax is defined for their use. In general terms, this syntax requires problematic eight-bit values to be replaced with a Reverse-Solidus character (0x5C, "\"), followed by a three-digit decimal value (in the range of "000" through "255") that corresponds to the canonical octet value.

This escape syntax MUST be applied to any octet value which does not explicitly represent a printable character (0x00 through 0x20, 0x7F through 0x9F, and 0xA0, inclusive), or which represents an embedded Reverse-Solidus character (0x5C, "\"). In those cases where a valid escape sequence already exists, that sequence (including its leading Reverse-Solidus character) MUST NOT be escaped again.

This escape syntax MAY be applied to any other character code or octet value, although the unnecessary usage of this mechanism is strongly DISCOURAGED. Furthermore, the availability of this mechanism MUST NOT be interpreted to mean that this mechanism can be used with any domain name; instead, it is only to be used with application-specific domain names which explicitly allow the presence of these problematic characters.

For example, if an application-specific domain name contains "weird name.example.com", the "weird name" portion of that domain name MUST be escaped as "weird\032name". Meanwhile, if an application-specific domain name contains "local\046postmaster", this sequence would be unmodified since the Reverse-Solidus character is already part of a valid escape sequence.

This escape syntax MUST be applied to an input domain name before that domain name undergoes the conversion process described in section 3.2. Furthermore, the leaf-node applications which generate and use these domain names SHOULD escape the data before it is passed to an LDAP agent, since those agents cannot be expected to know all of the application-specific usages of a domain name. For example, an application which uses a domain name with an embedded Full-Stop character (0x2E, ".") SHOULD escape that character before storing or passing the domain name to an LDAP agent, thus eliminating the possibility of having that agent interpret the embedded Full-Stop character as a label separator.

Note that any Reverse Solidus characters in the resulting domain name will be further escaped when these sequences are transferred in LDAP messages. For example, "weird\032name" will be further escaped as "weird\\032name" when it is passed in an LDAP message (this secondary escape will be stripped upon receipt, leaving the escaped domain name in its original form).

Also note that Reverse-Solidus characters are frequently illegal as data in URIs, and these characters will probably end up being percent-escaped whenever they are provided in a URI as data.

3.4. Client Transformation Guidelines

This specification defines data elements for storing the rich representations of internationalized domain names, with the expectation that those domain names will be viewed and manipulated in their native form, and that ASCII compatible encodings of those domain names will be provided in the legacy elements.

Therefore, in the general case, clients and servers SHOULD display and use internationalized domain name elements in the native form as offered by the elements in use, and SHOULD NOT transform the data into another representation for display purposes unless the user has specifically requested the client to provide an alternative representation of those elements.

Specifically, IDNA encoded elements SHOULD NOT be transformed into UTF-8, and vice versa, unless the user explicitly requests this action to take place.

4. Internationalized Domain Components

This section defines the idc (inetDomainComponent) attribute, the inetIdcObject object class, and the inetIdcDomain object class, as internationalized versions of the schema defined in [RFC2247].

In the typical usage scenario, the inetDomainComponent attribute and related object classes will be used to define the naming context of a directory partition. For new installations which are limited to this simple usage, these elements can be deployed without any special considerations.

In those cases which uses the domainComponent attribute already exists, but where applications do not perform any kind of active mapping between domain names and the domainComponent attributes, it may be feasible to simply migrate the existing data from the existing naming context, and to either update the clients to use the newer partition or use referrals to redirect the clients automatically.

In those cases where agents perform some kind of active mapping between domain names and the legacy domainComponent attributes, the use of referrals may be sufficient to redirect LDAP clients away from the domainComponent naming context and towards the inetDomainComponent naming context.

In those cases where referrals to another naming context are not feasible, organizations can continue to use the domainComponent naming, and simply apply the inetIdcObject auxiliary object class to the existing entries. Although this approach will not modernize the partition structure, it will allow both attribute sets to coexist simultaneously, and will also allow searches for the internationalized domain names to successfully match.

4.1. The idc (inetDomainComponent) Attribute

The idc attribute is for internationalized domain names which are to be stored in LDAP as sequences of relative distinguished names, with each attribute being mapped to discrete labels from the associated DNS domain name.

The idc attribute type is defined as follows:

```
( OID.TBD
  NAME ( 'idc' 'inetDomainComponent' )
  EQUALITY caseIgnoreMatch
  ORDERING caseIgnoreOrderingMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
```

Servers MAY map the idc attribute name to a synonym of inetDomainComponent. Systems MUST NOT use inetDomainComponent as the attribute name in any messages they generate.

The value of this attribute is a directory string holding one label component of an internationalized domain name, preferably as normalized and validated according to the process defined in section 3.

In those cases where the input domain name specifically and solely refers to the root domain, the root domain MUST be represented as a single idc attribute containing a single Full-Stop character ("idc="), and in this specific scenario, the Full-Stop character MUST NOT be escaped. If the input domain name contains any other sequence of labels, the root domain MUST NOT be specified in the resulting idc attribute sequence.

idc attributes are typically mapped to zone names, and as such, the corresponding domain names are usually restricted to hostname rules. However, this restriction is not formally implied or mandated, and any domain name (containing any octet value) is

explicitly permitted. As such, the UseSTD3ASCIIRules and AllowUnassigned IDNA flags MUST be set to their most liberal settings during conversion. However, subsequent application-specific uses of the idc attribute MAY apply stricter syntax checks if needed (this may be necessary with application-specific usages such as digital certificates, for example). In this regard, the ability to represent a domain name in an idc attribute does not guarantee that every application will be able to use the corresponding attribute value.

Note that many characters in an internationalized domain name will be converted to lowercase as part of the normalization process. However, case MUST be ignored during comparison operations since the existence of legacy systems will mean that not all domain names will have been properly normalized.

4.2. The inetIdcDomain Object Class

The inetIdcDomain object class is a structural object class, which uses idc as the mandatory naming attribute, and which provides several additional optional attributes that may be used to describe a particular organizational entity.

The inetIdcDomain object class is defined as follows:

```
( OID.TBD
  NAME 'inetIdcDomain'
  SUP top
  STRUCTURAL
  MUST idc
  MAY ( userPassword $ searchGuide $ seeAlso $
  businessCategory $ x121Address $ registeredAddress $
  destinationIndicator $ preferredDeliveryMethod $
  telexNumber $ teletexTerminalIdentifier $ telephoneNumber $
  internationaliSDNNumber $ facsimileTelephoneNumber $ street
  $ postOfficeBox $ postalCode $ postalAddress $
  physicalDeliveryOfficeName $ st $ l $ description $ o $
  associatedName ) )
```

The optional attributes of the inetIdcDomain object class are used for describing the object represented by this domain name, and may also be useful for searches.

Note that the dcObject object class defined in [RFC2247] may be used to supplement the inetIdcDomain object class, allowing the

legacy domainComponent attribute to be bound to an entry which uses the inetIdcDomain object class.

4.3. The inetIdcObject Object Class

The inetIdcObject object class is an auxiliary object class which provides idc as an optional attribute. The inetIdcObject object class is intended to be used in conjunction with a structural object class such as organization, organizationalUnit, or locality, none of which provide the idc attribute.

The inetIdcObject object class is defined as follows:

```
( OID.TBD
  NAME 'inetIdcObject'
  SUP top
  AUXILIARY
  MUST idc )
```

Note that the structural object class in use with the entry will define the naming attribute for that entry. As such, the idc attribute will only be one of potentially many child attributes, with no relevance to the name of the entry.

5. Internationalized Email Addresses

This section defines the inetEmail attribute and the auxiliary inetEmailObject object class, as internationalized versions of the mail attribute defined in RFC 1274 [RFC1274].

In the typical usage scenario, the inetEmail attribute and related object class will be used to define an internationalized Internet email address attribute as additional data for a contact-related object class such as inetOrgPerson. The inetEmail attribute is not expected to be used as a naming attribute, and imposes no namespace considerations.

The inetEmail and mail attributes can coexist in an entry without difficulty, with the mail attribute providing the mail domain name in the IDNA encoded form.

5.1. The inetEmail Attribute

The inetEmail attribute is for internationalized Internet email addresses which are to be stored in LDAP as whole sequences.

The inetEmail attribute type is defined as follows:

```
( OID.TBD
  NAME 'inetEmail'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

The inetEmail attribute is technically unstructured, although it ostensibly uses a three-part syntax consisting of a localpart element, a Commercial-At character (0x40, "@"), and an internationalized domain name.

The localpart element is currently unspecified, pending ongoing effort to internationalize this element. In the meantime, agents SHOULD NOT prohibit any possible uses of this element. Note that subsequent versions of this specification may define specific rules for this element.

The domain name portion of the email address SHOULD be treated as an internationalized domain name, and SHOULD be validated according to the procedure defined in section 3.2 wherever it is feasible and beneficial to do so.

Due to the way that email routing uses domain names, the domain element of an inetEmail attributes has to be mapped to a domain name which satisfies strict naming requirements, and as such, the corresponding domain names have to be restricted to hostname rules. Since the more liberal forms cannot be used, the UseSTD3ASCIIRules and AllowUnassigned IDNA flags MUST be set to their strictest settings when the domain name element is validated and normalized.

5.2. The inetEmailObject Object Class

The inetEmailObject object class is an auxiliary object class which provides inetEmail as an optional attribute. The inetEmailObject object class is intended to be used in conjunction with a structural object class such as person, inetOrgPerson, or organizationalPerson, none of which provide the inetEmail attribute.

The inetEmailObject object class is defined as follows:

```
( OID.TBD
  NAME 'inetEmailObject'
  SUP top
  AUXILIARY
  MUST inetEmail )
```

Note that the structural object class in use with the entry will define the naming attribute for that entry. As such, the inetEmail attribute will only be one of potentially many child attributes, with no relevance to the name of the entry.

6. The iLDAP URI Scheme

This section defines the iLDAP URI scheme as an internationalized version of the LDAP URL scheme defined in RFC 2255 [RFC2255], and clarifies its use with the labeledURI attribute defined in RFC 2079 [RFC2079].

The iLDAP URI scheme is designed as a UTF-8 URI scheme, capable of containing a fully internationalized sequence of elements. Specifically, the iLDAP URI scheme is syntactically identical to the LDAP URL scheme defined in [RFC2255], except that all of the elements in the iLDAP URI (including the hostname element) carry UTF-8 data, in an unencoded and unescaped form.

The LDAP attributes designed to carry URIs are multi-valued, and an iLDAP URI can freely coexist with the legacy LDAP URL scheme as an equal sibling. In these kinds of scenarios, LDAP agents can freely choose among the URI schemes offered in an multi-valued array, and ignore those URI schemes that it does not support. As such, the presence of the iLDAP URI scheme in these arrays is not usually harmful.

In order for that strategy to work properly, traditional LDAP URIs MUST be provided as an equal sibling wherever an iLDAP URI is also being provided. Otherwise, it is possible for a legacy agent to be presented with a URI that it does not support.

Furthermore, in order to limit the potential for damage in secondary applications, LDAP agents which support the iLDAP URI syntax MUST ensure that they will perform any necessary encoding of the URI if that data is subsequently passed across a seven-bit medium. For example, if an LDAP agent expects to pass an iLDAP URI

in a seven-bit email message, the contents of the URI must either be converted into a seven-bit compatible form, or the message body must be encoded as Base64 or Quoted-Printable, or some other steps must be taken to ensure that the iLDAP URI does not become corrupted by the seven-bit medium.

In those cases where an iLDAP URI has to be converted into a seven-bit compatible form, the hostname in the "hostport" element MUST be IDNA encoded, and the remainder of the elements MUST be percent-escaped as described in [RFC2255].

Note that the labeledURI attribute defined in [RFC2079] uses the directoryString syntax, although [RFC2079] explicitly states that the use of non-ASCII characters is discouraged. This specification overrides that recommendation when the iLDAP URI type is used.

7. Open Issues

Should a structured domain name sequence be formally defined, with the inetDomainComponent and inetEmail attributes formally incorporating that sequence?

Should the inetEmail attribute be formally defined as structured, with the appropriate ASN.1?

Should there be extensible matching filters that accept non-normalized input, normalize it, and then search for matching elements and/or attributes?

8. Security Considerations

TBD

9. IANA Considerations

TBD

10. Author's Addresses

Eric A. Hall
ehall@ehsco.com

11. Normative References

[ASCII] Cerf, V. "ASCII format for Network Interchange", RFC 20, October 1969.

[ISO10646] "International Standard -- Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane", ISO/IEC 10646-1:2000.

[RFC1274] Barker, P., and S. Kille, "The COSINE and Internet X.500 Schema", RFC 1274, November 1991.

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

[RFC2247] Kille, S., Wahl, M., Grimstad, A., Huber, R., and Sataluri, S. "Using Domains in LDAP/X.500 DNs", RFC 2247, January 1998.

[RFC2251] Wahl, M., Howes, T., and Kille, S. "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.

[RFC2252] Wahl, M., Coulbeck, A., Howes, T., and Kille, S. "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions", RFC 2252, December 1997.

[RFC2254] Howes, T. "The String Representation of LDAP Search Filters", RFC 2254, December 1997.

[RFC2255] Howes, T., and M. Smith, "The LDAP URL Format", RFC 2255, December 1997.

[RFC2279] Yergeau, F. "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[RFC3377] Hodges, J., and Morgan, R. "Lightweight Directory Access Protocol (v3): Technical Specification", RFC 3377, September 2002.

[RFC3490] Faltstrom, P., Hoffman, P., and Costello, A. "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

12. Acknowledgments

Funding for the RFC editor function is currently provided by the Internet Society.

This document incorporates several elements from Internet Drafts written by Kurt Zeilenga, who also played an important part in the development of the concepts included herein.

13. Full Copyright Statement