Network Working Group
Request for Comments: 2596
Category: Standards Track

M. Wahl
Innosoft International, Inc.
T. Howes
Netscape Communications Corp.

May 1999

Use of Language Codes in LDAP

Status of this Memo

Copyright Notice

1. Abstract

The Lightweight Directory Access Protocol [1] provides a means for clients to interrogate and modify information stored in a distributed directory system. The information in the directory is maintained as attributes [2] of entries. Most of these attributes have syntaxes which are human-readable strings, and it is desirable to be able to indicate the natural language associated with attribute values.

This document describes how language codes [3] are carried in LDAP and are to be interpreted by LDAP servers. All implementations MUST be prepared to accept language codes in the LDAP protocols. Servers may or may not be capable of storing attributes with language codes in the directory. This document does not specify how to determine whether particular attributes can or cannot have language codes.

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

2. Language Codes

Section 2 of RFC 1766 [3] describes the language code format which is used in LDAP. Briefly, it is a string of ASCII alphabetic characters and hyphens. Examples include "fr", "en-US" and "ja-JP".

Language codes are case insensitive. For example, the language code "en-us" is the same as "EN-US" and "en-US".

Implementations MUST NOT otherwise interpret the structure of the code when comparing two codes, and MUST treat them as simply strings of characters. Client and server implementations MUST allow any arbitrary string which follows the patterns given in RFC 1766 to be used as a language code.

3. Use of Language Codes in LDAP

This section describes how LDAP implementations MUST interpret language codes in performing operations.

In general, an attribute with a language code is to be treated as a subtype of the attribute without a language code. If a server does not support storing language codes with attribute values in the DIT, then it MUST always treat an attribute with a language code as an unrecognized attribute.

3.1. Attribute Description

An attribute consists of a type, a list of options for that type, and a set of one or more values. In LDAP, the type and the options are combined into the AttributeDescription, defined in section 4.1.5 of [1]. This is represented as an attribute type name and a possibly-empty list of options. One of these options associates a natural language with values for that attribute.

```
language-option = "lang-" lang-code

lang-code = printable-ascii ; a code as defined in RFC 1766
```

Multiple language options may be present on a particular value.

The language code has no effect on the character set encoding for string representations of DirectoryString syntax values; the UTF-8 representation of UniversalString (ISO 10646) is always used.

Examples of valid AttributeDescription:

```
givenName;lang-en-US
CN;lang-ja
```

In LDAP and in examples in this document, a directory attribute is represented as an AttributeDescription with a list of values. Note that the data could be stored in the LDAP server in a different representation.

3.2. Distinguished Names and Relative Distinguished Names

No attribute description options are permitted in Distinguished Names or Relative Distinguished Names. Thus language codes MUST NOT be used in forming DNs.

3.3. Search Filter

If a language code is present in an AttributeDescription in a search filter, then only attribute values in the directory which match the base attribute type or its subtype, the language code and the assertion value match this filter.

Thus for example a filter of an equality match of type "name;lang-en-US" and assertion value "Billy Ray", against the following directory entry

objectclass: top	DOES NOT MATCH (wrong type)
objectclass: person	DOES NOT MATCH (wrong type)
name;lang-EN-US: Billy Ray	MATCHES
name;lang-EN-US: Billy Bob	DOES NOT MATCH (wrong value)
CN;lang-en-us: Billy Ray	MATCHES
CN;lang-EN-US;dynamic: Billy Ray	MATCHES
CN;lang-en;dynamic: Billy Ray	DOES NOT MATCH (differing lang-)
name: Billy Ray	DOES NOT MATCH (no lang-)
SN: Ray	DOES NOT MATCH (wrong value)

(Note that "CN" and "SN" are subtypes of "name".)

Client implementors should however note that providing a language code in a search filter AttributeDescription will often filter out desirable values where the language code does not match exactly. For example, the filter (name;lang-en=Billy Ray) does NOT match the attribute "name;lang-en-US: Billy Ray".

If the server does not support storing language codes with attribute values in the DIT, then any filter which includes a language code will always fail to match, as it is an unrecognized attribute type. No error would be returned because of this; a presence filter would evaluate to FALSE and all other forms to Undefined.

If no language code is specified in the search filter, then only the base attribute type and the assertion value need match the value in the directory.

Thus for example a filter of an equality match of type "name" and assertion value "Billy Ray", against the following directory entry

objectclass: top	DOES NOT MATCH (wrong type)
objectclass: person	DOES NOT MATCH (wrong type)
name;lang-EN-US: Billy Ray	MATCHES
name;lang-EN-US: Billy Bob	DOES NOT MATCH (wrong value)
CN;lang-EN-US;dynamic: Billy Ray	MATCHES
CN;lang-en;dynamic: Billy Ray	MATCHES
name: Billy Ray	MATCHES
SN: Ray	DOES NOT MATCH (wrong value)

Thus in general, clients SHOULD NOT use the language code option in AttributeDescription fields in search filters.

3.4. Compare

A language code can be present in an AttributeDescription used in a compare request AttributeValueAssertion. This is to be treated by servers the same as the use of language codes in a search filter with an equality match, as described in the previous section. If there is no attribute in the entry with the same subtype and language code, the noSuchAttributeType error will be returned.

Thus for example a compare request of type "name" and assertion value "Johann", against an entry with all the following directory entry

```
objectclass: top
objectclass: person
givenName;lang-de-DE: Johann
CN: Johann Sibelius
SN: Sibelius
```

will cause the server to return compareTrue.

However, if the client issued a compare request of type "name;lang-de" and assertion value "Johann" against the above entry, the request would fail with the noSuchAttributeType error.

If the server does not support storing language codes with attribute values in the DIT, then any comparison which includes a language code will always fail to locate an attribute type, and noSuchAttributeType will be returned.

Thus in general, clients SHOULD NOT use the language code option in AttributeDescription fields in the compare request.

3.5. Requested Attributes in Search

Clients MAY provide language codes in AttributeDescription in the requested attribute list in a search request.

If a language code is provided in an attribute description, then only attribute values in a directory entry which have the same language code as that provided are to be returned. Thus if a client requests an attribute "description;lang-en", the server MUST NOT return values of an attribute "description" or "description;lang-fr".

Clients MAY provide in the attribute list multiple AttributeDescription which have the same base attribute type but different options. For example a client MAY provide both "name;lang-en" and "name;lang-fr", and this would permit an attribute with either language code to be returned. Note there would be no need to provide both "name" and "name;lang-en" since all subtypes of name would match "name".

If a server does not support storing language codes with attribute values in the DIT, then any attribute descriptions in the list which include language codes are to be ignored, just as if they were unknown attribute types.

If a request is made specifying all attributes or an attribute is requested without providing a language code, then all attribute values regardless of their language code are returned.

For example, if the client requests a "description" attribute, and a matching entry contains

```
objectclass: top
objectclass: organization
O: Software GmbH
description: software
description;lang-en: software products
description;lang-de: Softwareprodukte
postalAddress: Berlin 8001 Germany
postalAddress;lang-de: Berlin 8001 Deutschland
```

The server will return:

```
description: software
description;lang-en: software products
description;lang-de: Softwareprodukte
```

3.6. Add Operation

Clients MAY provide language codes in AttributeDescription in attributes of a new entry to be created, subject to the limitation that the client MUST NOT use language codes in the attribute value or values which form the RDN of the entry.

A client MAY provide multiple attributes with the same attribute type and value, so long as each attribute has a different language code, and at most one attribute does not have a language code option.

Servers which support storing language codes in the DIT MUST allow any attribute it recognizes that has the Directory String syntax to have a language option associated with it. Servers SHOULD allow language options to be associated with other attributes.

For example, the following is a legal request.

```
objectclass: top
objectclass: person
objectclass: residentialPerson
name: John Smith
CN: John Smith
CN;lang-en: John Smith
SN: Smith
streetAddress: 1 University Street
streetAddress;lang-en: 1 University Street
streetAddress;lang-fr: 1 rue Universite
houseIdentifier;lang-fr: 9e etage
```

If a server does not support storing language codes with attribute values in the DIT, then it MUST treat an AttributeDescription with a language code as an unrecognized attribute. If the server forbids the addition of unrecognized attributes then it MUST fail the add request with the appropriate result code.

3.7. Modify Operation

A client MAY provide a language code in an AttributeDescription as part of a modification element in the modify operation.

Attribute types and language codes MUST match exactly against values stored in the directory. For example, if the modification is a "delete", then if the stored values to be deleted have a language code, the language code MUST be provided in the modify operation, and if the stored values to be deleted do not have a language code, then no language code is to be provided.

If the server does not support storing language codes with attribute values in the DIT, then it MUST treat an AttributeDescription with a language code as an unrecognized attribute, and MUST fail the request with an appropriate result code.

3.8. Diagnostic Messages

Servers SHOULD use only printable ASCII characters in the errorMessage field, as not all clients will be able to display the full range of Unicode.

4. Differences from X.500(1997)

X.500(1997) defines a different mechanism, contexts, as the means of representing language tags. This section summarizes the major differences in approach.

a) An X.500 operation which has specified a language code on a value matches a value in the directory without a language code.
b) LDAP references RFC 1766, which allows for IANA registration of new tags.
c) LDAP does not allow language codes in distinguished names.
d) X.500 describes subschema administration procedures to allow language codes to be associated with particular attributes types.

5. Security Considerations

There are no known security considerations for this document. See the security considerations sections of [1] and [2] for security considerations of LDAP in general.

6. Acknowledgements

This document is a product of the IETF ASID and LDAPEXT working groups. Martin Duerst provided many valuable comments on an earlier version of this document.

7. Bibliography

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

[2] Wahl, M., Coulbeck, A., Howes, T. and S. Kille, "Lightweight X.500 Directory Access Protocol Attribute Syntax Definitions", RFC 2252, December 1997.

[3] Alvestrand, H.,"Tags for the Identification of Languages", RFC 1766, March 1995.

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

8. Authors' Addresses

Mark Wahl
Innosoft International, Inc.
8911 Capital of Texas Hwy Suite 4140
Austin, TX 78759 USA

EMail: M.Wahl@innosoft.com

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

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

Full Copyright Statement

Acknowledgement

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