Network Working Group
Request for Comments: 4521
BCP: 118
Category: Best Current Practice

K. Zeilenga
OpenLDAP Foundation

June 2006

Considerations for Lightweight Directory Access Protocol (LDAP) Extensions

Status of This Memo

This document specifies an Internet Best Current Practices for the Internet Community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

Copyright Notice

Abstract

The Lightweight Directory Access Protocol (LDAP) is extensible. It provides mechanisms for adding new operations, extending existing operations, and expanding user and system schemas. This document discusses considerations for designers of LDAP extensions.

Table of Contents

1. Introduction

The Lightweight Directory Access Protocol (LDAP) [RFC4510] is an extensible protocol.

LDAP allows for new operations to be added and for existing operations to be enhanced [RFC4511].

LDAP allows additional schema to be defined [RFC4512][RFC4517]. This can include additional object classes, attribute types, matching rules, additional syntaxes, and other elements of schema. LDAP provides an ability to extend attribute types with options [RFC4512].

LDAP supports a Simple Authentication and Security Layer (SASL) authentication method [RFC4511][RFC4513]. SASL [RFC4422] is extensible. LDAP may be extended to support additional authentication methods [RFC4511].

LDAP supports establishment of Transport Layer Security (TLS) [RFC4511][RFC4513]. TLS [RFC4346] is extensible.

LDAP has an extensible Uniform Resource Locator (URL) format [RFC4516].

Lastly, LDAP allows for certain extensions to the protocol's Abstract Syntax Notation - One (ASN.1) [X.680] definition to be made. This facilitates a wide range of protocol enhancements, for example, new result codes needed to support extensions to be added through extension of the protocol's ASN.1 definition.

This document describes practices that engineers should consider when designing extensions to LDAP.

1.1. Terminology

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 BCP 14 [RFC2119]. In this document, "the specification", as used by BCP 14, RFC 2119, refers to the engineering of LDAP extensions.

The term "Request Control" refers to a control attached to a client-generated message sent to a server. The term "Response Control" refers to a control attached to a server-generated message sent to a client.

DIT stands for Directory Information Tree.
DSA stands for Directory System Agent, a server.
DSE stands for DSA-Specific Entry.
DUA stands for Directory User Agent, a client.
DN stands for Distinguished Name.

2. General Considerations

2.1. Scope of Extension

Mutually agreeing peers may, within the confines of an extension, agree to significant changes in protocol semantics. However, designers MUST consider the impact of an extension upon protocol peers that have not agreed to implement or otherwise recognize and support the extension. Extensions MUST be "truly optional" [RFC2119].

2.2. Interaction between extensions

Designers SHOULD consider how extensions they engineer interact with other extensions.

Designers SHOULD consider the extensibility of extensions they specify. Extensions to LDAP SHOULD themselves be extensible.

Except where it is stated otherwise, extensibility is implied.

2.3. Discovery Mechanism

Extensions SHOULD provide adequate discovery mechanisms.

As LDAP design is based upon the client-request/server-response paradigm, the general discovery approach is for the client to discover the capabilities of the server before utilizing a particular extension. Commonly, this discovery involves querying the root DSE and/or other DSEs for operational information associated with the extension. LDAP provides no mechanism for a server to discover the capabilities of a client.

The 'supportedControl' attribute [RFC4512] is used to advertise supported controls. The 'supportedExtension' attribute [RFC4512] is used to advertise supported extended operations. The 'supportedFeatures' attribute [RFC4512] is used to advertise features. Other root DSE attributes MAY be defined to advertise other capabilities.

2.4. Internationalization Considerations

LDAP is designed to support the full Unicode [Unicode] repertory of characters. Extensions SHOULD avoid unnecessarily restricting applications to subsets of Unicode (e.g., Basic Multilingual Plane, ISO 8859-1, ASCII, Printable String).

LDAP Language Tag options [RFC3866] provide a mechanism for tagging text (and other) values with language information. Extensions that define attribute types SHOULD allow use of language tags with these attributes.

2.5. Use of the Basic Encoding Rules

Numerous elements of LDAP are described using ASN.1 [X.680] and are encoded using a particular subset [Protocol, Section 5.2] of the Basic Encoding Rules (BER) [X.690]. To allow reuse of parsers/generators used in implementing the LDAP "core" technical specification [RFC4510], it is RECOMMENDED that extension elements (e.g., extension specific contents of controlValue, requestValue, responseValue fields) described by ASN.1 and encoded using BER be subjected to the restrictions of [Protocol, Section 5.2].

2.6. Use of Formal Languages

Formal languages SHOULD be used in specifications in accordance with IESG guidelines [FORMAL].

2.7. Examples

Example DN strings SHOULD conform to the syntax defined in [RFC4518]. Example LDAP filter strings SHOULD conform to the syntax defined in [RFC4515]. Example LDAP URLs SHOULD conform to the syntax defined in [RFC4516]. Entries SHOULD be represented using LDIF [RFC2849].

2.8. Registration of Protocol Values

Designers SHALL register protocol values of their LDAP extensions in accordance with BCP 64, RFC 4520 [RFC4520]. Specifications that create new extensible protocol elements SHALL extend existing registries or establish new registries for values of these elements in accordance with BCP 64, RFC 4520 [RFC4520] and BCP 26, RFC 2434 [RFC2434].

3. LDAP Operation Extensions

Extensions SHOULD use controls in defining extensions that complement existing operations. Where the extension to be defined does not complement an existing operation, designers SHOULD consider defining an extended operation instead.

For example, a subtree delete operation could be designed as either an extension of the delete operation or as a new operation. As the feature complements the existing delete operation, use of the control mechanism to extend the delete operation is likely more appropriate.

As a counter (and contrived) example, a locate services operation (an operation that would return for a DN a set of LDAP URLs to services that may hold the entry named by this DN) could be designed as either a search operation or a new operation. As the feature doesn't complement the search operation (e.g., the operation is not contrived to search for entries held in the Directory Information Tree), it is likely more appropriate to define a new operation using the extended operation mechanism.

3.1. Controls

Controls [Protocol, Section 4.1.11] are the RECOMMENDED mechanism for extending existing operations. The existing operation can be a base operation defined in [RFC4511] (e.g., search, modify) , an extended operation (e.g., Start TLS [RFC4511], Password Modify [RFC3062]), or an operation defined as an extension to a base or extended operation.

Extensions SHOULD NOT return Response controls unless the server has specific knowledge that the client can make use of the control. Generally, the client requests the return of a particular response control by providing a related request control.

An existing operation MAY be extended to return IntermediateResponse messages [Protocol, Section 4.13].

Specifications of controls SHALL NOT attach additional semantics to the criticality of controls beyond those defined in [Protocol, Section 4.1.11]. A specification MAY mandate the criticality take on a particular value (e.g., TRUE or FALSE), where appropriate.

3.1.1. Extending Bind Operation with Controls

Controls attached to the request and response messages of a Bind Operation [RFC4511] are not protected by any security layers established by that Bind operation.

Specifications detailing controls extending the Bind operation SHALL detail that the Bind negotiated security layers do not protect the information contained in these controls and SHALL detail how the information in these controls is protected or why the information does not need protection.

It is RECOMMENDED that designers consider alternative mechanisms for providing the function. For example, an extended operation issued subsequent to the Bind operation (hence, protected by the security layers negotiated by the Bind operation) might be used to provide the desired function.

Additionally, designers of Bind control extensions MUST also consider how the controls' semantics interact with individual steps of a multi-step Bind operation. Note that some steps are optional and thus may require special attention in the design.

3.1.2. Extending the Start TLS Operation with Controls

Controls attached to the request and response messages of a Start TLS Operation [RFC4511] are not protected by the security layers established by the Start TLS operation.

Specifications detailing controls extending the Start TLS operation SHALL detail that the Start TLS negotiated security layers do not protect the information contained in these controls and SHALL detail how the information in these controls is protected or why the information does not need protection.

It is RECOMMENDED that designers consider alternative mechanisms for providing the function. For example, an extended operation issued subsequent to the Start TLS operation (hence, protected by the security layers negotiated by the Start TLS operation) might be used to provided the desired function.

3.1.3. Extending the Search Operation with Controls

The Search operation processing has two distinct phases:

- finding the base object; and
- searching for objects at or under that base object.

Specifications of controls extending the Search Operation should clearly state in which phase(s) the control's semantics apply. Semantics of controls that are not specific to the Search Operation SHOULD apply in the finding phase.

3.1.4. Extending the Update Operations with Controls

Update operations have properties of atomicity, consistency, isolation, and durability ([ACID]).

- atomicity: All or none of the DIT changes requested are made.
- consistency: The resulting DIT state must be conform to schema and other constraints.
- isolation: Intermediate states are not exposed.
- durability: The resulting DIT state is preserved until subsequently updated.

When defining a control that requests additional (or other) DIT changes be made to the DIT, these additional changes SHOULD NOT be treated as part of a separate transaction. The specification MUST be clear as to whether the additional DIT changes are part of the same or a separate transaction as the DIT changes expressed in the request of the base operation.

When defining a control that requests additional (or other) DIT changes be made to the DIT, the specification MUST be clear as to the order in which these and the base changes are to be applied to the DIT.

3.1.5. Extending the Responseless Operations with Controls

The Abandon and Unbind operations do not include a response message. For this reason, specifications for controls designed to be attached to Abandon and Unbind requests SHOULD mandate that the control's criticality be FALSE.

3.2. Extended Operations

Extended Operations [Protocol, Section 4.12] are the RECOMMENDED mechanism for defining new operations. An extended operation consists of an ExtendedRequest message, zero or more IntermediateResponse messages, and an ExtendedResponse message.

3.3. Intermediate Responses

Extensions SHALL use IntermediateResponse messages instead of ExtendedResponse messages to return intermediate results.

3.4. Unsolicited Notifications

Unsolicited notifications [Protocol, Section 4.4] offer a capability for the server to notify the client of events not associated with the operation currently being processed.

Extensions SHOULD be designed such that unsolicited notifications are not returned unless the server has specific knowledge that the client can make use of the notification. Generally, the client requests the return of a particular unsolicited notification by performing a related extended operation.

For example, a time hack extension could be designed to return unsolicited notifications at regular intervals that were enabled by an extended operation (which possibly specified the desired interval).

4. Extending the LDAP ASN.1 Definition

LDAP allows limited extension [Protocol, Section 4] of the LDAP ASN.1 definition [Protocol, Appendix B] to be made.

4.1. Result Codes

Extensions that specify new operations or enhance existing operations often need to define new result codes. The extension SHOULD be designed such that a client has a reasonably clear indication of the nature of the successful or non-successful result.

Extensions SHOULD use existing result codes to indicate conditions that are consistent with the intended meaning [RFC4511][X.511] of these codes. Extensions MAY introduce new result codes [RFC4520] where no existing result code provides an adequate indication of the nature of the result.

Extensions SHALL NOT disallow or otherwise restrict the return of general service result codes, especially those reporting a protocol, service, or security problem, or indicating that the server is unable or unwilling to complete the operation.

4.2. LDAP Message Types

While extensions can specify new types of LDAP messages by extending the protocolOp CHOICE of the LDAPMessage SEQUENCE, this is generally unnecessary and inappropriate. Existing operation extension mechanisms (e.g., extended operations, unsolicited notifications, and intermediate responses) SHOULD be used instead. However, there may be cases where an extension does not fit well into these mechanisms.

In such cases, a new extension mechanism SHOULD be defined that can be used by multiple extensions that have similar needs.

4.3. Authentication Methods

The Bind operation currently supports two authentication methods, simple and SASL. SASL [RFC4422] is an extensible authentication framework used by multiple application-level protocols (e.g., BEEP, IMAP, SMTP). It is RECOMMENDED that new authentication processes be defined as SASL mechanisms. New LDAP authentication methods MAY be added to support new authentication frameworks.

The Bind operation's primary function is to establish the LDAP association [RFC4513]. No other operation SHALL be defined (or extended) to establish the LDAP association. However, other operations MAY be defined to establish other security associations (e.g., IPsec).

4.4. General ASN.1 Extensibility

Section 4 of [RFC4511] states the following:

> In order to support future extensions to this protocol, extensibility is implied where it is allowed per ASN.1 (i.e., sequence, set, choice, and enumerated types are extensible). In addition, ellipses (...) have been supplied in ASN.1 types that are explicitly extensible as discussed in [RFC4520]. Because of the implied extensibility, clients and servers MUST (unless otherwise specified) ignore trailing SEQUENCE components whose tags they do not recognize.

Designers SHOULD avoid introducing extensions that rely on unsuspecting implementations to ignore trailing components of SEQUENCE whose tags they do not recognize.

5. Schema Extensions

Extensions defining LDAP schema elements SHALL provide schema definitions conforming with syntaxes defined in [Models, Section 4.1]. While provided definitions MAY be reformatted (line wrapped) for readability, this SHALL be noted in the specification.

For definitions that allow a NAME field, new schema elements SHOULD provide one and only one name. The name SHOULD be short.

Each schema definition allows a DESC field. The DESC field, if provided, SHOULD contain a short descriptive phrase. The DESC field MUST be regarded as informational. That is, the specification MUST

be written such that its interpretation is the same with and without the provided DESC fields.

The extension SHALL NOT mandate that implementations provide the same DESC field in the schema they publish. Implementors MAY replace or remove the DESC field.

Published schema elements SHALL NOT be redefined. Replacement schema elements (new OIDs, new NAMEs) SHOULD be defined as needed.

Schema designers SHOULD reuse existing schema elements, where appropriate. However, any reuse MUST not alter the semantics of the element.

5.1. LDAP Syntaxes

Each LDAP syntax [RFC4517] is defined in terms of ASN.1 [X.680]. Each extension detailing an LDAP syntax MUST specify the ASN.1 data definition associated with the syntax. A distinct LDAP syntax SHOULD be created for each distinct ASN.1 data definition (including constraints).

Each LDAP syntax SHOULD have a string encoding defined for it. It is RECOMMENDED that this string encoding be restricted to UTF-8 [RFC3629] encoded Unicode [Unicode] characters. Use of Generic String Encoding Rules (GSER) [RFC3641][RFC3642] or other generic string encoding rules to provide string encodings for complex ASN.1 data definitions is RECOMMENDED. Otherwise, it is RECOMMENDED that the string encoding be described using a formal language (e.g., ABNF [RFC4234]). Formal languages SHOULD be used in specifications in accordance with IESG guidelines [FORMAL].

If no string encoding is defined, the extension SHALL specify how the transfer encoding is to be indicated. Generally, the extension SHOULD mandate use of binary or other transfer encoding option.

5.2. Matching Rules

Three basic kinds of matching rules (e.g., EQUALITY, ORDERING, and SUBSTRING) may be associated with an attribute type. In addition, LDAP provides an extensible matching rule mechanism.

The matching rule specification SHOULD detail which kind of matching rule it is and SHOULD describe which kinds of values it can be used with.

In addition to requirements stated in the LDAP technical specification, equality matching rules SHOULD be commutative.

5.3. Attribute Types

Designers SHOULD carefully consider how the structure of values is to be restricted. Designers SHOULD consider that servers will only enforce constraints of the attribute's syntax. That is, an attribute intended to hold URIs, but that has directoryString syntax, is not restricted to values that are URIs.

Designers SHOULD carefully consider which matching rules, if any, are appropriate for the attribute type. Matching rules specified for an attribute type MUST be compatible with the attribute type's syntax.

Extensions specifying operational attributes MUST detail how servers are to maintain and/or utilize values of each operational attribute.

5.4. Object Classes

Designers SHOULD carefully consider whether each attribute of an object class is required ("MUST") or allowed ("MAY").

Extensions specifying object classes that allow (or require) operational attributes MUST specify how servers are to maintain and/or utilize entries belonging to these object classes.

6. Other Extension Mechanisms

6.1. Attribute Description Options

Each option is identified by a string of letters, numbers, and hyphens. This string SHOULD be short.

6.2. Authorization Identities

Extensions interacting with authorization identities SHALL support the LDAP authzId format [RFC4513]. The authzId format is extensible.

6.3. LDAP URL Extensions

LDAP URL extensions are identified by a short string, a descriptor. Like other descriptors, the string SHOULD be short.

7. Security Considerations

LDAP does not place undue restrictions on the kinds of extensions that can be implemented. While this document attempts to outline some specific issues that designers need to consider, it is not (and

cannot be) all encompassing. Designers MUST do their own evaluations of the security considerations applicable to their extensions.

Designers MUST NOT assume that the LDAP "core" technical specification [RFC4510] adequately addresses the specific concerns surrounding their extensions or assume that their extensions have no specific concerns.

Extension specifications, however, SHOULD note whether security considerations specific to the feature they are extending, as well as general LDAP security considerations, apply to the extension.

8. Acknowledgements

The author thanks the IETF LDAP community for their thoughtful comments.

This work builds upon "LDAP Extension Style Guide" [GUIDE] by Bruce Greenblatt.

9. References

9.1. Normative References

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

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC2849] Good, G., "The LDAP Data Interchange Format (LDIF) - Technical Specification", RFC 2849, June 2000.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC3641] Legg, S., "Generic String Encoding Rules (GSER) for ASN.1 Types", RFC 3641, October 2003.

[RFC3642] Legg, S., "Common Elements of Generic String Encoding Rules (GSER) Encodings", RFC 3642, October 2003.

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

[RFC3866] Zeilenga, K., Ed., "Language Tags and Ranges in the Lightweight Directory Access Protocol (LDAP)", RFC 3866, July 2004.

[RFC4234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

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

[RFC4511] Sermersheim, J., Ed., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, June 2006.

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

[RFC4513] Harrison, R., Ed., "Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms", RFC 4513, June 2006.

[RFC4515] Smith, M., Ed. and T. Howes, "Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters", RFC 4515, June 2006.

[RFC4516] Smith, M., Ed. and T. Howes, "Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator", RFC 4516, June 2006.

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

[RFC4518] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names", RFC 4518, June 2006.

[RFC4520] Zeilenga, K., "Internet Assigned Numbers Authority (IANA) Considerations for the Lightweight Directory Access Protocol (LDAP)", BCP 64, RFC 4520, June 2006.

[RFC4422] Melnikov, A., Ed. and K. Zeilenga, Ed., "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.

[Unicode] The Unicode Consortium, "The Unicode Standard, Version 3.2.0" is defined by "The Unicode Standard, Version 3.0" (Reading, MA, Addison-Wesley, 2000. ISBN 0-201-61633-5), as amended by the "Unicode Standard Annex #27: Unicode 3.1" (http://www.unicode.org/reports/tr27/) and by the "Unicode Standard Annex #28: Unicode 3.2" (http://www.unicode.org/reports/tr28/).

[FORMAL] IESG, "Guidelines for the use of formal languages in IETF specifications", <http://www.ietf.org/IESG/STATEMENTS/pseudo-code-in-specs.txt>, 2001.

[X.511] International Telecommunication Union - Telecommunication Standardization Sector, "The Directory: Abstract Service Definition", X.511(1993) (also ISO/IEC 9594-3:1993).

[X.680] International Telecommunication Union - Telecommunication Standardization Sector, "Abstract Syntax Notation One (ASN.1) - Specification of Basic Notation", X.680(2002) (also ISO/IEC 8824-1:2002).

[X.690] International Telecommunication Union - Telecommunication Standardization Sector, "Specification of ASN.1 encoding rules: Basic Encoding Rules (BER), Canonical Encoding Rules (CER), and Distinguished Encoding Rules (DER)", X.690(2002) (also ISO/IEC 8825-1:2002).

9.2. Informative References

[ACID] Section 4 of ISO/IEC 10026-1:1992.

[GUIDE] Greenblatt, B., "LDAP Extension Style Guide", Work in Progress.

[RFC3062] Zeilenga, K., "LDAP Password Modify Extended Operation", RFC 3062, February 2001.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

Author's Address

Kurt D. Zeilenga
OpenLDAP Foundation

EMail: Kurt@OpenLDAP.org

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).