Oracle® Fusion Middleware Glossary for Oracle Unified Directory

11*g* Release 1 (11.1.1)

Part No: 821–1276
June 2011

110711@25097

Contents

CHAPTER 1

Glossary

This glossary defines the vocabulary used to describe LDAP and directory servers and includes terms that are specific to the Oracle Unified Directory proxy.

A

abandon operation

The LDAP abandon operation can be used to request that the server stop processing on an outstanding request. The abandon request protocol op is as follows:

```
AbandonRequest ::= [APPLICATION 16] MessageID
```

The message ID provided in the request is the message ID of the operation to abandon.

The abandon operation does not have a response, so there is no way for clients to know whether the abandon operation was successful. Similarly, if an operation was abandoned, then no response will be provided for it, so the client may wait indefinitely for a response that will never be sent. Both of these issues are addressed by the cancel extended operation.

Bind, unbind, abandon, and StartTLS extended operations cannot be abandoned.

abstract object class

An abstract object class is one that cannot be used directly in an entry but must be subclassed by either a structural object class or auxiliary object class. The subclasses will inherit any required and/or optional attribute type defined by the abstract class.

One of the most notable abstract object classes defined in LDAP is the `top` object class, which is the root class for virtually all other object classes defined in the server schema.

Abstract Syntax Notation One

Abstract Syntax Notation One (ASN.1) is a mechanism for encoding data in a binary form. It uses a TLV structure, in which each element has a type, length, and value. The type component is a data type that indicates what kind of information is stored in the element and indicates how the value should be encoded. The length component specifies the number of bytes in the value, and the value is the actual data held by the element.

Examples of ASN.1 elements include the following:

Null	Null elements do not hold any value. They are generally used as placeholders when an element is required but no value is needed.
Octet string	Octet string elements hold a set of zero or more octets (bytes) of data. It can be used for holding string or binary data.
Boolean	Boolean elements hold values that represent either `true` or `false`.
Integer	Integer elements hold values that represent integer values.
Enumerated	Enumerated elements hold values that represent integer values where each value has a specific meaning.
Sequence	Sequence elements are containers that hold zero or more other ASN.1 elements in a manner where the order of the elements is significant.
Set	Set elements are containers that hold zero or more other ASN.1 elements in a manner where the order of the elements is not significant.

Note that ASN.1 is a general framework for binary encoding, but doesn't actually define how the data should be encoded. That is handled by an encoding rule, and there are a number of different kinds of ASN.1 encoding rules. LDAP uses the Basic Encoding Rules encoding, but other types include Distinguished Encoding Rules (DER), Canonical Encoding Rules (CER), and Packed Encoding Rules (PER).

access control

Access control provides a mechanism for restricting who can get access to various kinds of information in the Directory Server. The access control provider can be used to control a number of things, including:

- Whether or not a client can retrieve an entry from the server.
- Which attributes within the entry the client is allowed to retrieve.
- Which values of an attribute the client is allowed to retrieve.
- The ways in which the client is able to manipulate data in the directory.

A number of things can be taken into account when making access control decisions, including:

- The DN as whom the user is authenticated.
- The method by which the client authenticated to the server.
- Any groups in which that user is a member.
- The contents of the authenticated user's entry.
- The contents of the target entry.
- The address of the client system.
- Whether or not the communication between the client and server is secure.
- The time of day and/or day of week of the attempt.

See Chapter 9, "Controlling Access To Data," in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory* for details on the access control syntax.

In addition to the access control subsystem, the directory server also provides a privilege that can be used to control what a user will be allowed to do. One of the privileges available is the `bypass-acl` privilege, which can be used to allow that client to bypass any restrictions that the access control subsystem would otherwise enforce.

access control instruction (ACI)

See "access control rule" on page 15.

access control rule

An access control rule (also called an access control instruction, or ACI), is a rule which may be used to grant or deny a user or set of users access to perform some kind of operation in the server. The Directory Server access control policy comprises the complete set of access control rules defined in the server.

See Chapter 9, "Controlling Access To Data," in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory* for more information about the syntax used for access control rules and the operations that can be allowed or denied using them.

access log

The Directory Server access log provides a mechanism for keeping track of every operation processed by the server, including every request received and response returned. It may also be used to obtain information about the internal operations performed within the server.

The directory server provides an extensible framework for implementing access loggers (as well as error and debug loggers). The default access control log implementation writes information to a log file with two records per operation. The first record reflects the request received from the client and the second provides information about the result of the operation processing.

All messages will include a common set of elements including:

- The time that the message was logged.
- The type of operation being processed.
- The connection ID of the client connection that requested the operation.
- The operation ID of the operation on that client connection.
- The message ID of the message used to request the operation.

For abandon operations, request log messages include the message ID of the operation to abandon. There is no response to an abandon operation, but the server will nevertheless log a result message indicating whether the abandon was successful and the processing time in milliseconds.

For add operations, request log messages include the DN of the entry to add. The response log message may include the result code, diagnostic message, matched DN, the authorization ID for the operation, and the processing time in milliseconds.

For bind operations, request log messages include the authentication type (either `SIMPLE` or `SASL` followed by the mechanism name) and the bind DN. The response log message may include the result code, diagnostic message, matched DN, authentication ID, authorization ID, and processing time in milliseconds.

For compare operations, request log messages include the target entry DN and the attribute type. The response log message may include the result code, diagnostic message, matched DN, authorization ID, and the processing time in milliseconds.

For delete operations, request log messages include the target entry DN. The response log message may include the result code, diagnostic message, matched DN, authorization ID, and the processing time in milliseconds.

For extended operations, request log messages include the OID for the extended request. The response log message may include the OID of the extended response, the result code, diagnostic message, matched DN, and the processing time in milliseconds.

For modify operations, request log messages include the target entry DN. The response log message may include the result code, diagnostic message, matched DN, authorization ID, and the processing time in milliseconds.

For modify DN operations, request log messages include the target entry DN, the new RDN, a flag indicating whether to delete the old RDN values, and the new superior DN. The response log message may include the result code, diagnostic message, matched DN, authorization ID, and the processing time in milliseconds.

For search operations, request log messages include the search base DN, scope, filter, and search attributes. The response log message may include the result code, number of entries returned, diagnostic message, matched DN, authorization ID, and the processing time in milliseconds.

For unbind operations, the request message will simply indicate that an unbind request has been received. There is no response to an unbind request, and no result log message.

account expiration

Account expiration is a component of the Directory Server password policy that may be used to indicate that an account is no longer able to be used beyond a given date. This feature may be useful for creating temporary user accounts (for example, for use by contractors, interns, or other temporary workers) that will expire after a specified date.

Account expiration may be enabled by adding the `ds-pwp-account-expiration-time` operational attribute to the target user's entry. The value for this attribute should be a time stamp in generalized time format that specifies the time that the account should expire. Once the account expiration time has passed, the user will no longer be allowed to authenticate to the server.

account lockout

Account lockout is a component of the Directory Server password policy that may be used to lock user accounts after too many failed bind attempts. Once an account has been locked, that user will not be allowed to authenticate. The lockout may be temporary (automatically ending after a specified period of time) or permanent (remaining in effect until an administrator resets the user's password).

account status notification

An account status notification is a mechanism that can be used to provide indication that a user account has changed in a manner that is significant with regard to the server's password policy.

The types of account status notifications available for use in the server include:

- When the user's account has been locked due too many failed attempts
- When the user's account has been locked due to remaining idle too long
- When the user's account has been unlocked by an administrator
- When the user's account has been manually disabled or re-enabled by an administrator
- When the user's account has expired

- When the user's password has expired or is about to expire
- When the user's password has been reset by an administrator
- When the user's password has been changed by the end user

The directory server provides an extensible framework for handling account status notifications. The default handler writes messages to the server's error log, but the framework can be used to send email messages or take other actions that may be desired.

account usability control

The account usability control provides a pair of request and response controls that can be used to determine whether a user account may be used for authenticating to the server.

The request control has an OID of 1.3.6.1.4.1.42.2.27.9.5.8 and does not include a value. It should only be included in search request messages.

The corresponding response control has an OID of 1.3.6.1.4.1.42.2.27.9.5.8 (the same as the request control), and it will be included in any search result entry messages for a search request that includes the account usability request control.

The value for the account usability response control is encoded as follows:

```
ACCOUNT_USABLE_RESPONSE ::= CHOICE {
     is_available           [0] INTEGER, -- Seconds before expiration --
     is_not_available       [1] MORE_INFO }

     MORE_INFO ::= SEQUENCE {
     inactive               [0] BOOLEAN DEFAULT FALSE,
     reset                  [1] BOOLEAN DEFAULT FALSE,
     expired                [2] BOOLEAN DEFAULT_FALSE,
     remaining_grace        [3] INTEGER OPTIONAL,
     seconds_before_unlock  [4] INTEGER OPTIONAL }
```

If the user account is available, then the control will include the number of seconds until the user's password expires, or -1 if password expiration is not enabled. If the user's account is not available, then the control will provide the reason it is unavailable.

For an example of using this control in a search request, see “To Search Using the Account Usability Request Control” in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

ACID

ACID is an acronym that stands for Atomicity, Consistency, Isolation, and Durability. This term is standard database terminology that refers to the characteristics that can be achieved using the transactional nature of the database. These elements include:

Atomicity	Each transaction performed in the database is atomic. That is, it either completely succeeds or completely fails. It never partially succeeds such that some changes that are part of the transaction are applied while others are not.
Consistency	The database is always in a consistent state such that the integrity of its contents will be preserved. It should not be possible for a successful or failed transaction to leave the database in an inconsistent state.
Isolation	The operations performed as part of a transaction will be isolated from other operations performed in the database at the same time. If one transaction is used to make a number of changes to database contents, then it should not be possible for another transactional operation to see the effects of those changes until they have been committed.
Durability	Any transaction that the database has reported as complete and committed successfully is guaranteed to be on persistent storage. Even if the directory server, or the underlying JVM, operating system, or hardware should fail the instant after the notification of the successful commit, then that change will not be lost.

The Berkeley DB Java Edition used as the data store for the primary back end provides full support for ACID compliance, although it also provides methods for relaxing its compliance to these constraints if desirable for performance reasons. The directory server exposes some of this flexibility, particularly with regard to configuring how durable the changes will be (for example, it is possible to configure the server so that changes are not immediately flushed to disk, which may allow better write performance but could cause the loss of one or more changes in the event of a hardware or software failure).

add operation

The LDAP add operation can be used to create an entry in the Directory Server. The add request protocol op is defined as follows:

```
AddRequest ::= [APPLICATION 8] SEQUENCE {
     entry           LDAPDN,
     attributes      AttributeList }
```

The elements included in this request include the DN of the entry to add and the set of attributes to include in that entry.

The response to an LDAP add operation is an LDAP result element, defined as follows:

```
AddResponse ::= [APPLICATION 9] LDAPResult
```

alias

An alias is a special type of entry that references another entry in the server, much like a symbolic link in a UNIX file system. It should include the `alias` object class and the `aliasedObjectName` attribute with a value equal to the DN of the entry that it references.

Aliases are primarily used for search operations. In particular, the search request includes an element that specifies the dereference policy that should be used when aliases are encountered. The allowed dereference policy values include:

neverDerefAliases	The server should never dereference alias entries.
derefInSearching	The server should dereference any alias entries that it finds in the possible set of search result entries, but if the search base DN specifies an alias entry it will not be dereferenced.
derefFindingBaseObj	The server should dereference the search base entry if it is an alias, but it will not dereference any aliases within the possible set of search result entries.
derefAlways	The server should dereference any aliases encountered, whether in the search base entry or in the possible set of search result entries.

Note that aliases are an optional part of the LDAPv3 protocol, and the directory server does not currently support them.

AND search filter

An AND search filter is a type of search filter that is intended to serve as a container that holds zero or more other search filters. In order for an entry to match an AND filter, it must match all of the filters contained in that AND filter.

AND filters may be represented as a string by enclosing the entire filter in parentheses and placing an ampersand just after the opening parenthesis. For example, a filter of `(&(objectClass=person)(uid=john.doe))` represents an AND search filter that embeds the `(objectClass=person)` and `(uid=john.doe)` equality filters.

An AND filter that does not contain any embedded filters is called an LDAP true filter. The string representation for an LDAP true filter is an ampersand `(&)`, and LDAP true filters will always match any target entry.

anonymous bind

An anonymous bind is a type of bind operation using simple authentication with a zero-length bind DN and a zero-length password. It may be used to destroy any previous authentication performed on a connection and return it to an unauthenticated state.

Note that there is an ANONYMOUS SASL mechanism that has the same effect, but in general the term "anonymous bind" refers to the simple bind operation with no DN and password.

ANONYMOUS SASL mechanism

The ANONYMOUS SASL mechanism is a type of SASL authentication mechanism. It is different from other SASL mechanisms in that it is used to create an unauthenticated session, and will destroy any previous authentication that may have been performed on the connection.

The ANONYMOUS SASL mechanism provides the ability to include trace information in the request that may be included in the server's access log. This trace information can provide information about the client performing the bind, although because no authentication is performed the validity of the trace information cannot be guaranteed.

approximate index

An approximate index is a type of index that is used to efficiently identify which entries are approximately equal to a given assertion value. An approximate index can be maintained only for attributes that have a corresponding approximate matching rules. That matching rule are used to normalize values to use as index keys, and the value for that key is the ID list containing the entry IDs of the entries with values that are approximately equal to that normalized value.

approximate search filter

An approximate search filter is a type of search filter that can be used to identify entries that contain a value for a given attribute that is approximately equal to a given assertion value. The server will use an approximate matching rule to make the determination.

The string representation of an LDAP approximate filter comprises an opening parenthesis followed by the attribute name, a tilde, an equal sign, the attribute value, and the closing parenthesis. For example, an equality filter of `(givenName~=John` will match any entry in which the `givenName` attribute contains a value that is approximately equal to `John`.

ASN.1

See Abstract Syntax Notation One.

assertion value

An assertion value is the value of an attribute value assertion. The assertion value is provided to a matching rule in order to make a determination about the value of a specified attribute.

attribute

An attribute is a named set of values. An attribute has an attribute description, which contains the name of that attribute (which links it to an attribute type) and an optional set of attribute options, and a collection of one or more values.

An entry contains a collection of attributes. It is possible for an entry to have multiple attributes with the same attribute type but different sets of options.

attribute description

An attribute description is used to identify a given attribute in an entry. An attribute description contains a name or OID that ties it to an attribute type and zero or more attribute options. If the attribute description contains any attribute options, then they are separated from the attribute name/OID by a semicolon, and a semicolon is also used to separate individual attribute options if there is more than one option in the attribute description.

attribute option

An attribute option is a kind of tag that provides additional information about the way that an attribute should be interpreted. An attribute description consists of the attribute name or OID followed by zero or more attribute options. If there are attribute options, then they are separated from the attribute name and from each other using semicolons. For example, in the attribute description `userCertificate;binary`, the attribute name is `userCertificate` and the attribute option is `binary`.

Attribute options can be used for several purposes, including providing information about how the server should treat that attribute (for example, the binary encoding option as described in RFC 4522) They may also be provided for the benefit of clients in some form (for example, the language tag options as described in RFC 3866, which make it possible to provide an attribute value in different languages).

attribute syntax

An attribute syntax is a schema element that defines a kind of data type that is used to dictate the kind of information that may be stored in an attribute value. Any attempt to store an attribute value that violates the syntax for the associated attribute type should be rejected.

Common attribute syntaxes include:

Binary	Can hold any kind of data, whether textual or not, that should be compared on a byte-for-byte basis. Note that the binary syntax has been deprecated in favor of the octet string syntax.
Boolean	Can hold values of either `TRUE` or `FALSE`.
Directory String	Can hold any kind of string value (technically, binary values are allowed as well, but directory string values are typically strings).
Distinguished Name	Can hold values that are valid DNs.
Generalized Time	Can hold values that contain time stamps of varying precision (anywhere from an hour to a fraction of a second) including time zone information. For example, the value `20070525222745Z` represents a time stamp of May 25, 2007 at 10:27:45 PM in the UTC time zone.
IA5 String	Can hold values that contain ASCII strings (that is, use of non-ASCII characters is not allowed).
Integer	Can hold integer values. Positive, negative, and zero values are allowed.
Octet String	Can hold any kind of data that should be compared on a byte-for-byte basis.
Postal Address	Can hold a multi-line address, in which the lines of the address should be separated by dollar signs.
Printable String	Can hold a string containing any combination of printable characters. Printable characters include all uppercase and lowercase ASCII letters, the numeric digits, the space character, and the symbols `'()+,-.=/:?`.
Telephone Number	Can hold telephone number values.

The set of attribute syntaxes defined in the server may be determined by retrieving the `ldapSyntaxes` attribute of the subschema subentry. For more information about attribute syntaxes, see “Understanding Attribute Syntaxes” in *Oracle Fusion Middleware Architecture Reference for Oracle Unified Directory*.

attribute type

An attribute type is a schema element that correlates an OID and a set of names with an attribute syntax and a set of matching rules.

The components of an attribute type definition include:

- An OID used to uniquely identify the attribute type.
- A set of zero or more names that can be used to more easily reference the attribute type.
- An optional equality matching rule that specifies how equality matching should be performed on values of that attribute. If no equality matching rule is specified, then the default equality rule for the associated attribute syntax will be used. If the associated syntax doesn't have a default equality matching rule, then equality operations will not be allowed for that attribute.
- An optional ordering matching rule that specifies how ordering operations should be performed on values of that attribute. If no ordering matching rule is specified, then the default ordering rule for the associated attribute syntax will be used. If the associated syntax doesn't have a default ordering matching rule, then ordering operations will not be allowed for that attribute.
- An optional substring matching rule that specifies how substring matching should be performed on values of that attribute. If no substring matching rule is specified, then the default substring rule for the associated attribute syntax will be used. If the associated syntax doesn't have a default substring matching rule, then substring operations will not be allowed for that attribute.
- An optional syntax OID that specifies the syntax for values of the attribute. If no syntax is specified, then it will default to the directory string syntax.
- A flag that indicates whether the attribute is allowed to have multiple values.
- An optional attribute usage string indicating the context in which the attribute is to be used.
- An optional flag that indicates whether the attribute can be modified by external clients.

The set of attribute types defined in the server may be determined by retrieving the `attributeTypes` attribute of the subschema subentry. For more information about attribute types, see “Understanding Attribute Types” in *Oracle Fusion Middleware Architecture Reference for Oracle Unified Directory*.

attribute usage

An attribute type's attribute usage defines the contexts in which it may be used. There are four types of attribute usage:

userApplications	This should be used for all attribute types that are intended for use in holding user-defined data.
directoryOperation	This should be used for attribute types that are used for behind-the-scenes processing within the server.
distributedOperation	This should be used for attribute types that store operational data that need to be distributed (that is, replicated) throughout the directory environment.
dSAOperation	This should be used for attribute types that store operational data that should be stored only in one server and should not be replicated throughout the directory environment.

Attributes with a usage of `userApplications` are known as user attributes. Attributes with a usage of `directoryOperation`, `distributedOperation`, or `dSAOperation` are known as operational attributes.

attribute value

An attribute value describes an element of actual data held by an attribute. An attribute may have multiple values, if allowed by the associated attribute type. The way that the server should interact with the values of that attribute is governed by that attribute's syntax and matching rules.

attribute value assertion

An attribute value assertion (AVA) is a combination of an attribute description and an attribute value. The assertion value is used in conjunction with a matching rule in order to make the determination. If the matching rule is an equality matching rule, then it will be used to determine whether the attribute contains a given value. If it is an ordering matching rule, then the AVA will be used to determine whether the attribute contains a value that is greater than or equal to, or less than or equal to, the assertion value. If it is an approximate matching rule, then the AVA will be used to determine whether the attribute contains a value that is approximately equal to the assertion value. Substring matching is more complex and uses a substring assertion rather than a simple assertion value.

Attribute Value assertions are used in LDAP compare operations, as well as equality, greater or equal, less or equal, and approximate match search filters.

audit log

The audit log is a special type of access log that is used to log information about all changes that are made in the server. It provides a log of those changes in LDIF form so that administrators can see exactly what changes were made. This information can be used for diagnostic purposes when investigating a problem, to help better understand the kinds of changes that an application might make in the directory, or to help collect information about changes for replay to an alternate repository.

The name “audit log” is a legacy term referring to its use in the Netscape Directory Server. It should not be confused with a log that could be used for security auditing, as it only records changes to directory data and does not keep track of things like successful or failed authentication attempts. However, in many cases, the combination of the content from the traditional access log and the audit log can be used to obtain this kind of information. If desired, an administrator could also provide a custom access logging implementation to keep track of any kind of desired information.

authentication

Authentication is the process whereby a client identifies itself to the directory server and provides proof of its identity. In LDAP, this is performed through the use of a bind operation.

The authentication process has two phases:

Identification	The client identifies itself to the server in some way. In simple authentication, the DN provided in the bind request is used for this purpose. In SASL authentication, the identity of the client is obtained through some other means (for example, using a certificate, a Kerberos principal, or some other kind of identifier).
Verification of Identity	The client must provide sufficient proof that it is who it has identified itself to be. In simple authentication, this is done through the password. In SASL authentication, this verification is obtained in a manner specific to the associated mechanism (it may be a password, or it may be a certificate or some other form of proof).

Some authentication mechanisms may be considered stronger than others. For example, simple authentication may be considered less trustworthy if the client has a password that is easy to guess or obtain through some other means, whereas authentication using a certificate or Kerberos credentials might be considered much stronger and harder to forge. The directory server's access control implementation may be configured to take the client's authentication mechanism into account when determining whether a requested operation will be allowed.

authentication ID

An authentication ID is an identifier that is used by a client to identify itself to the Directory Server for certain kinds of SASL mechanisms (for example, CRAM-MD5, DIGEST-MD5, and PLAIN). It can be used to allow a client to identify itself with a username (or other friendly identifier) rather than a DN.

In most cases, an authentication ID should be specified in one of the following forms:

- The string `dn:` followed by the distinguished name of the target user (or just the string `dn:` if the authentication identity should be that of the anonymous user).
- The string `u:` followed by a username used to identify the user. An identity mapper will be used to map the provided username to the corresponding user entry.

authentication password syntax

The authentication password syntax defines a standard method for encoding a user password for storage in the server, ideally in a manner that makes it difficult or impossible to determine the clear-text value of that password.

The authentication password syntax is described in RFC 3112, which defines the `authPassword` attribute type and a corresponding `authPasswordObject` auxiliary object class that will allow the use of that attribute.

The basic form of a password encoded using the authentication password syntax is:

scheme `$`*authInfo* `$` *authValue*

where *scheme* is the name of the scheme used to encode the value, *authInfo* is some kind of modifier (for example, a salt) used in the encoding process, and *authValue* is the encoded password information. For example, the value `SHA1$RzqH67DY3uQ=$atAcDs1eS+IJwPy7V4UDXEoBrDI=` is encoded using the authentication password syntax The scheme is `SHA1`, the authInfo element is `RzqH67DY3uQ=`, and the authValue element is `atAcDs1eS+IJwPy7V4UDXEoBrDI=`.

The authentication password schemes supported by the directory server include the following:

MD5	Uses the MD5 message digest.
SHA1	Uses the SHA-1 variant of the Secure Hash Algorithm.
SHA256	Uses the 256-bit SHA-2 variant of the Secure Hash Algorithm.
SHA384	Uses the 384-bit SHA-2 variant of the Secure Hash Algorithm.
SHA512	Uses the 512-bit SHA-2 variant of the Secure Hash Algorithm.

authorization

Authorization is the process of determining whether a user will be allowed to perform a requested operation. A number of server components may be involved in the authorization process, including:

- The access control handler.
- The privilege subsystem.
- The password policy.
- Custom plug-ins installed in the server.

authorization ID

An authorization ID is an identifier that is used by a client to indicate that one or more operations should be performed under the authority of an alternate identity. This alternate authorization identity can last for a single operation (when used in conjunction with the proxied authorization control) or for the entire duration of an authentication session (when used in conjunction with an appropriate SASL mechanism, like DIGEST-MD5, GSSAPI, or PLAIN).

In most cases, an authorization ID should be specified in one of the following forms:

- The string `dn:` followed by the distinguished name of the target user (or just the string `dn:` if the authorization identity should be that of the anonymous user).
- The string `u:` followed by a username used to identify the user. An identity mapper maps the provided username to the corresponding user entry.

The ability for a client to use an alternate authorization identity is controlled by the `proxied-auth` privilege. In some cases, additional access control rights may also be required.

authorization identity control

The authorization identity controls are a pair of request and response controls defined in RFC 3829 that can be used in conjunction with an LDAP bind operation to allow the client to learn the authorization identity for the client connection.

The authorization identity request control has an OID of `2.16.840.1.113730.3.4.16` and does not have a value. The authorization identity response control has an OID of `2.16.840.1.113730.3.4.15` and the value of that control should be a string representing the authorization identify for that connection (or an empty string if the authorization identity is that of the anonymous user). The response control should only be included in the response if the authentication was successful.

Note that the authorization identity controls are only allowed for use in conjunction with the LDAP bind operation, and therefore cannot be used after the client has authenticated. The "Who Am I?" extended operation can be used to obtain the authorization identity at any time after the bind has completed.

For an example of using this control in a search request, see "To Search Using the Authorization Identity Request Control" in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

auxiliary object class

An auxiliary object class is one that does not define the core type of an entry, but defines additional characteristics of that entry. An entry can contain zero or more auxiliary object classes. The set of auxiliary classes allowed for use in an entry may be controlled by a DIT content rule associated with that entry's structural object class.

AVA

See attribute value assertion.

B

back end

A Directory Server back end provides a repository for storing data and a set of logic for interacting with that data. A back end will typically contain some kind of database and may maintain a set of indexes that allows the back end to quickly locate entries for various operations. All back ends will have the following qualities:

- A back end ID, which uniquely identifies that back end among all other back ends in the server.
- A set of one or more base DNs that indicate the data that the back end holds.
- A writability mode, which indicates whether the back end will accept write operations.

The logic provided by the back end includes:

- A method for determining whether a given entry exists, based on its DN
- A method for retrieving an entry, based on its DN

- A method of adding a new entry to the database (as part of processing an LDAP add operation)
- A method for removing an existing entry from the database (as part of processing an LDAP delete operation)
- A method for replacing an entry in the database (as part of processing an LDAP modify operation)
- A method for renaming an entry in the database (as part of processing an LDAP modify DN operation)
- A method for processing an LDAP search operation
- A method for exporting the contents of the database in LDIF form
- A method for importing data in LDIF form into the database
- A method for performing a backup of the data
- A method for performing a restore of a previous backup

backup

A backup is a transportable representation of the data in a Directory Server back end. Each back end is responsible for controlling whether or not it is possible to back up its contents, and ensuring that the backup information is suitable to be restored at a later time. Note that the term "back up" is a verb (the action of backing up the contents of the back end) and "backup" is a noun (what you get when you perform a backup).

There are a number of reasons that a back end may not provide a backup mechanism. Some reasons include:

- The back end only contains temporary, point-in-time information that doesn't make sense to archive or attempt to restore at a later time (for example, the root DSE or the monitor back ends).
- The back end stores its information in a remote repository that is not directly available to be archived. In cases like this, the external repository will likely have its own backup and restore mechanism.

The primary back end used by the directory server is one that uses the Berkeley DB Java Edition as its underlying database and that back end provides complete backup and restore capabilities. The backup mechanism is also very portable and can be transported across different platforms and different filesystem locations, and it is suitable for use as a binary copy mechanism.

base64 encoding

Base64 encoding is a way of representing binary data in a text-only form. It is commonly used in LDIF for values containing non-ASCII characters, or for values that could otherwise be ambiguous (for example, values that begin or end with spaces). It is also frequently used to encode certificate contents or the output of message digests like MD5 or SHA. The base64 encoding is described in section 5.2 of RFC 1341.

The basic principle of base64 encoding is that it defines a 64-character alphabet containing the following characters in the given order:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
```

Each of those characters is assigned a numeric value between 0 and 63 based on its position in the list (that is, `A` is 0, `B` is 1, `C` is 2, ... `+` is 62, and `/` is 63). A value is broken up into six-bit segments, and each of those six bits is converted into a numeric value between 0 and 63 and replaced with the specified character from the alphabet given above. This means that every three bytes of a binary value is converted into four characters from the base64 alphabet. If the length of the binary value is not a multiple of three bytes, then it is zero-padded and either one or two equal signs are appended to the base64-encoded value.

Basic Encoding Rules

The Basic Encoding Rules (BER) are a set of Abstract Syntax Notation One encoding rules that define a specific way in which information may be encoded in a binary form. It is used as the underlying mechanism for encoding message.

Basic Encoding Rules Overview

Many network protocols are text-based, which has the advantages of being relatively easy to understand if you examine the network traffic, and in many cases you can even interact with the target server by telnetting to it and typing in the appropriate commands. However, there are disadvantages as well, including that they are generally more verbose and less efficient to parse than they need to be. On the other hand, other protocols use a binary encoding that is more compact and more efficient. LDAP falls into this category, and uses the ASN.1 (abstract syntax notation one) mechanism, and more specifically the BER (basic encoding rules) flavor of ASN.1. There are a number of other encoding rules (such as DER, PER, and CER) that fall under the ASN.1 umbrella, but LDAP uses BER.

This section discusses the subset of BER that is used by LDAP in particular and does not address other cases.

BER elements use a TLV structure, where TLV stands for “type”, “length”, and “value”. That is, each BER element has one or more bytes (in LDAP, typically only a single byte) that indicates the data type for the element, one or more bytes that indicate the length of the value, and the encoded value itself (where the form of the encoded value depends on the data type), which can be zero or more bytes, as described in the following sections:

- The BER type
- The BER Length
- The BER Value

The BER Type

The BER type indicates the data type for the value of the element. The BER specification provides several different data types, but the most commonly used by LDAP include `OCTET STRING` (which can be either a text string or just some binary data), `INTEGER`, `BOOLEAN`, `NULL`, `ENUMERATED` (like an integer, but where each value has a special meaning), `SEQUENCE` (an ordered collection of other elements, similar to an array), and `SET` (the same as a sequence, except that the order doesn't matter). There is also a `CHOICE` element, but it typically allows one of a few different kinds of elements.

The BER type is typically only a single byte, and this byte has data encoded in it. The two most significant bits (the two leftmost bits, because BER uses big endian/network ordering) are used to indicate the class for the element, using these possible class values:

00	The universal class. Most BER elements have a universal type, so any element with a universal type specifies what kind of data it holds. Examples of universal types include 0x01 (`BOOLEAN`), 0x02 (`INTEGER`), 0x04 (`OCTET STRING`), 0x05 (`NULL`), 0x0A (`ENUMERATED`), 0x30 (`SEQUENCE`), and 0x31 (`SET`). The binary encodings for all of those type values have the leftmost two bits set to zero.
01	The application-specific class. This class allows an application to define its own types that are consistent throughout that application. In this context, LDAP is considered an application. For example, when 0x42 appears in LDAP, it indicates an unbind request protocol op, because RFC 2251 section 4.3 states that the unbind request protocol op has a type of `[APPLICATION 2]`.
10	The context-specific class. This class indicates that the type is specific to a particular usage within a given application. The same type can be re-used in different contexts in the same application as long as there is enough other information to determine which context is applicable in a given situation. For example, in the context of the credentials in a bind request protocol op, the context-specific type 0x80 is used to hold the bind password, but in the context of an extended operation it would be used to hold the request OID.
11	The private class, not typically used in LDAP.

The next bit (the third from the left) is the primitive/constructed bit. If it is set to zero (off), then the element is considered primitive, and the value is encoded in accordance with the rules of that data type. If it is set to one (on), then it means that the value is constructed from zero or more other ASN.1 elements that are concatenated together in their encoded forms. For example, for the universal `SEQUENCE` type of 0x30, the binary encoding is `00110000` and the primitive/constructed bit is set to one indicating that the value of the sequence is constructed from zero or more encoded elements.

The final five bits of the BER type byte specify the value of that type, and they are treated as a simple integer value (where `00000` is zero, `00001` is one, `00010` is two, `00011` is three, and so on). The only special value is `11111`, which means that the type value is larger than can fit in the five bits allowed, and so multiple bytes are required. This value is not used in LDAP.

The BER Length

The second component in the TLV structure of a BER element is the length. This specifies the size in bytes of the encoded value. For the most part, this uses a straightforward binary encoding of the integer value (for example, if the encoded value is five bytes long, then it is encoded as 00000101 binary, or 0x05 hex), but if the value is longer than 127 bytes then it is necessary to use multiple bytes to encode the length. In that case, the first byte has the leftmost bit set to one and the remaining seven bits are used to specify the number of bytes required to encode the full length. For example, if there are 500 bytes in the length (hex 0x01F4), then the encoded length will actually consist of three bytes: `82 01 F4`.

Note that there is an alternate form for encoding the length called the indefinite form. In this mechanism, only a part of the length is given at a time, similar to the chunked encoding that is available in HTTP 1.1. However, this form is not used in LDAP, as specified in RFC 2251 section 5.1.

The BER Value

The BER element contains the actual data of the element. Because BER is a binary encoding, the encodings can take advantage of that to represent the data in a compact form. As such, each data type has its own encoded form:

`NULL`	The `NULL` element never has a value, and therefore the length is always zero.
`OCTET STRING`	The value of this element is encoded as a concatenation of the raw bytes of the data being represented. For example, to represent the string `Hello`, the encoded value would be `48 65 6C 6C 6F`. The value can have a length of zero bytes.
`BOOLEAN`	The value of this element is always a single byte. If all the bits in that byte are set to zero (0x00), then the value is `FALSE`. If one or more of the bytes is set to one, then the value is `TRUE`. As a result, there are 255 different ways to encode a `BOOLEAN` value of `TRUE`, but in practice it is generally encoded as 0xFF (that is, all the bits are set to one).

`INTEGER`	The value of this element is encoded as a binary integer in two's complement form. Although BER itself does not place a limit on the magnitude of the values that can be encoded, many software implementations have a cap of four or eight bytes (that is, 32-bit or 64-bit integer values), and LDAP generally uses a maximum of 4 bytes (which allows encoding values within the plus or minus 2 billion range). There is always at least one byte in the value.
`ENUMERATED`	The value of this element is encoded in exactly the same way as the value of an `INTEGER` element.
`SEQUENCE`	The value of this element is simply a concatenation of the encoded BER elements contained in the sequence. For example, to encode a sequence with two octet string elements encoding the text `Hello` and `there`, the encoded sequence value is `04 05 48 65 6C 6C 6F 04 05 74 68 65 72 65`. A sequence value can be zero bytes if there are no elements in the sequence.
`SET`	The value of this element is encoded in exactly the same way as the value of a `SEQUENCE` element.

BER Encoding Examples

The example above for encoding a `SEQUENCE` value had two complete BER elements concatenated together: the `OCTET STRING` representations of the strings `Hello` and `there`:

```
04 05 48 65 6C 6C 6F
04 05 74 68 65 72 65
```

In both of these cases, the first byte is the type (0x04, which is the universal primitive `OCTET STRING` type), and the second is the length (0x05, indicating that there are five bytes in the value). The remaining five bytes are the encoded representations of the strings `Hello` and `there`.

The following example encodes the integer value `3` using a context-specific type value of `5` instead of the universal `INTEGER` type:

```
85 01 03
```

The next example encodes an LDAP bind request protocol op as defined in RFC 2251 section 4.2. A simplified BNF representation of this element is as follows:

```
BindRequest ::= [APPLICATION 0] SEQUENCE {
     version                  INTEGER (1 .. 127),
     name                     OCTET STRING,
     authentication           CHOICE {
          simple              [0] OCTET STRING,
          sasl                [3] SEQUENCE {
               mechanism      OCTET STRING,
                credentials      OCTET STRING OPTIONAL } } }
```

This example encodes a bind request using simple authentication for the user `cn=test` with a password of `password`. The complete encoding for this bind request protocol op is:

```
60 16 02 01 03 04 07 63 6E 3D 74 65 73 74 80 08 70 61 73 73 77 6F 72 64
```

In analysis, that string of bytes contains the following information:

- The first byte is 0x60 and it is the BER type for the bind request protocol op. It comes from the `[APPLICATION 0] SEQUENCE` portion of the definition. Because it is application-specific, then the class bytes are `01`, and because it is a `SEQUENCE`, it is constructed. Put that together with a type value of zero, the binary representation is `01100000`, which is 0x60 hex.
- The second byte is 0x16, which indicates the length of the bind request sequence. 0x16 hex is 22 decimal, and the number of bytes after the 0x16 is 22.
- The next three bytes are `02 01 03`, which is a universal `INTEGER` value of `3`. It corresponds to the version component of the bind request sequence, and it indicates that this is an LDAPv3 bind request.
- The next nine bytes are `04 07 63 6E 3D 74 65 73 74`, which is a universal `OCTET STRING` containing the text `cn=test`. It corresponds to the "name" component of the bind request sequence.
- The last component is `80 08 70 61 73 73 77 6F 72 64`, which is an element with a type of `context-specific primitive 0` and a length of eight bytes. As specified in the definition of the bind request protocol op, `context-specific` maps to the simple authentication type and that it should be treated as an `OCTET STRING`, and those eight bytes in the value do represent the encoded string `password`.

BER

See Basic Encoding Rules.

Berkeley DB Java Edition

The Berkeley DB Java Edition (also referred to as "Berkeley DB JE", "BDBJE", or "JE") is a pure Java database designed by Sleepycat Software, which was purchased by the Oracle Corporation. It provides a highly-scalable, high-performance, transactional B-Tree database, with support for full ACID semantics and it is used as the primary database for storing user data.

The directory server provides a back end that uses the Berkeley DB Java Edition for storing its information. This back end is often called the "JE Backend" or simply "JEB". It uses a Berkeley DB Java Edition environment that consists of multiple individual databases. The id2entry database provides a mechanism for mapping entry ID values to entry contents. Other databases serve as indexes that can be used to quickly find entry contents for processing various types of operations.

binary copy

Binary copy refers to the process of performing a backup of a Directory Server back end of one server instance and restore that back end into another instance of the server. This can provide a fast disaster recovery mechanism and can also be used as a replica initialization mechanism.

Not all Directory Server back ends necessarily support the use of binary copy, and those that do may not support it in all circumstances. The primary back end type used by the directory server is based on the use of the Berkeley DB Java Edition, and it does support the use of the binary copy mechanism, including across different operating systems and CPU architectures, and with different filesystem locations. However, it does require that both servers have the same set of base DNs and the same types of indexes defined.

bind operation

The LDAP bind operation can be used to authenticate to the Directory Server. There are two basic types of bind operations:

- A simple bind operation, which uses simple authentication involving a bind DN and password to authenticate to the server.
- A SASL bind operation, which uses the Simple Authentication and Security Layer to authenticate the client, which can use a variety of types of credentials based on the selected SASL mechanism.

The bind request protocol op is defined as follows:

```
BindRequest ::= [APPLICATION 0] SEQUENCE {
     version                 INTEGER (1 ..  127),
     name                    LDAPDN,
     authentication          AuthenticationChoice }

AuthenticationChoice ::= CHOICE {
     simple                  [0] OCTET STRING,
                                 -- 1 and 2 reserved
     sasl                    [3] SaslCredentials,
     ...  }

SaslCredentials ::= SEQUENCE {
     mechanism               LDAPString,
     credentials             OCTET STRING OPTIONAL }
```

The elements of the request include:

- The LDAP protocol version. Allowed values are 2 and 3, although LDAPv2 has been classified as a historical protocol and is no longer recommended for use.
- The bind DN. This is always used for simple authentication (although it may be a zero-length string for anonymous simple authentication), and is generally not used for SASL authentication.
- The credentials. The type of credentials provided vary based on the authentication type.
  - For simple authentication, the credentials should be the password for the target bind DN, or an empty string for anonymous simple authentication.
  - For SASL authentication, the credentials should include the name of the SASL mechanism to use, and may optionally include encoded credential information appropriate for the SASL mechanism.

The response to an LDAP bind operation is defined as follows:

```
BindResponse ::= [APPLICATION 1] SEQUENCE {
     COMPONENTS OF LDAPResult,
     serverSaslCreds    [7] OCTET STRING OPTIONAL }
```

This indicates that the bind response will include the elements in the LDAP result object and may also include a set of server SASL credentials if appropriate for the authentication type.

C

cancel extended operation

The LDAP Cancel extended operation is an extended operation that provides a function similar to the core LDAP abandon operation in that it can be used to request that the server stop processing on an operation in progress. The primary advantages of the Cancel extended operation over the abandon operation are that both the cancel request and the operation being canceled are guaranteed to get a response, whereas there is no response for the abandon request and there may not be a response for the operation being abandoned.

The Cancel extended operation is defined in RFC 3909. The value of the Cancel Request extended operation is encoded as follows:

```
cancelRequestValue ::= SEQUENCE {
     cancelID        MessageID
                     -- MessageID is as defined in [RFC2251]
}
```

CDDL

See Common Development and Distribution License.

certificate

A certificate is an element of public key cryptography that may be used to perform asymmetric encryption. In particular, a certificate consists of a pair of keys (called the “public key” and the “private key”, respectively) that are linked so that any data encrypted using the public key can be decrypted using the private key. With many public key algorithms, like RSA, the reverse is also true so that any data encrypted with the private key can be decrypted using the public key.

The term *certificate* has different meanings, based on the context in which it is used. In many cases, it refers to only the public key (in particular, whenever the server presents its certificate to the client, or if a client presents its certificate to the server, then only the public key is included). However, in other cases, it does include the private key (i.e., the server will require the use of the private key to establish a secure communication channel with the client, and the client will need access to its private key in order to send its own certificate to the server).

Certificates have two primary uses in the directory server. The first is for providing a secure communication mechanism, generally through the use of SSL or StartTLS. In this case, the negotiation process involves the client encrypting information using the server's public key so that only the server can decrypt it using its public key and that information will not be exposed to any third party that might be able to observe the communication. Certificates may also be used for data signing, in which case the server will encrypt information using its private key, and clients will know that the data is legitimately from the server if it can be decrypted using the server's public key.

certificate mapper

A certificate mapper provides the logic required to identify a user in the Directory Server that corresponds to a provided client certificate. The mapping may use any of the information contained in the certificate, although many certificate mappers are based primarily on the certificate's subject (the name of the certificate, which comprises a number of attribute-value pairs and looks very much like an LDAP distinguished name.

For more information about the certificate mappers available for use in the directory server, see the Certificate Mapper Configuration.

chaining

Chaining provides a mechanism for making data in a remote Directory Server instance appear as if it is part of the local server. That is, chaining is used to present a part of the DIT using data from another server. Any request that the server receives for data in a chained portion of the DIT will be transparently forwarded to the server that actually contains the request.

changelog

A changelog is a special kind of database that is used to keep track of the changes that occur in a Directory Server instance. There are two different kinds of changelogs:

- A replication changelog stores change information in a format needed for replication.
- An LDAP-accessible changelog that represents its data in the format specified in draft-good-ldap-changelog that allows clients to learn about the changes that have occurred in the directory environment.

cn=Directory Manager

See directory manager.

collective attribute

A collective attribute is a special type of virtual attribute that is defined in RFC 3671. Collective attributes enable you to define values that are assigned to attributes based on an entry's membership in a subentry.

Common Development and Distribution License

The Common Development and Distribution License (CDDL) is an OSI-approved open source license which is used by the OpenDS project, on which Oracle Unified Directory.

The CDDL is a file-based license, which means that any changes to files contained in the project need to remain licensed under the CDDL. New files, however, may be licensed under any license chosen by the author (including closed-source licenses). The CDDL is based on the Mozilla Public License (MPL) and includes a patent grant clause so that any technology covered by patents will be granted to other projects using the code.

The CDDL license contents may be found at http://www.opensource.org/licenses/cddl1.php.

compare operation

The LDAP compare operation can be used to determine whether a specified entry contains a given attribute value. The compare request protocol op is defined as follows:

```
CompareRequest ::= [APPLICATION 14] SEQUENCE {
     entry           LDAPDN,
     ava             AttributeValueAssertion }

AttributeValueAssertion ::= SEQUENCE {
     attributeDesc   AttributeDescription,
     assertionValue  AssertionValue }
```

The elements of the request include the following:

- The DN of the entry in which the comparison is to be made.
- The name of the attribute in which the comparison is to be made.
- The assertion value to try to find in the specified attribute.

The response to an LDAP compare operation is an LDAP result element as defined below:

```
CompareResponse ::= [APPLICATION 15] LDAPResult
```

connection handler

A connection handler is a component of the Directory Server that is responsible for accepting connections from clients, reading and parsing requests submitted by the clients, ensuring that they are processed by the server, and sending the corresponding responses back to the client. The connection handler manages all communication with the client and therefore needs to implement support for the associated protocol.

The directory server currently provides connection handlers capable of communicating using LDAP and JMX, as well as a special connection handler for internal use that may be used to allow components of the server (like plug-ins and other kinds of extensions) to perform operations. The server also provides an extensible connection handler API that may be used to implement support for additional network protocols.

connection ID

A connection ID is a unique integer identifier that is assigned to each connection maintained within the Directory Server. It is used primarily for logging purposes, so that it is possible to correlate the various operations performed on a given connection.

The connection ID counter starts at zero for the first connection received by the server and increments by one for each additional connection. The counter is reset whenever the server is restarted.

Internal connections, which are used for processing internal operations, are assigned negative values to distinguish them from connections from external clients.

control

An LDAP control is an element that may be included in an message. If it is included in a request message, it can be used to provide additional information about the way that the operation should be processed. If it is included in the response message, it can be used to provide additional information about the way the operation was processed.

Examples of LDAP controls include:

- Account usability control - This is a pair of request and response controls that indicate whether an account is able to authenticate to the server.
- Authorization identity control - This is a pair of request and response controls that may be used to determine the authorization identity for a user as part of a bind operation.
- Entry change notification control - This is a control that is included in search result entry messages performed as part of a persistent search to indicate how an entry has been updated.
- Get effective rights control - This is a request control that may be used to obtain information about what rights a user has for accessing a given entry.
- LDAP assertion control - This is a request control that may be used to ensure that an operation is only processed if the target entry matches a given assertion filter.
- LDAP no-op control - This is a request control that may be used to ensure that a write operation does not actually change any information in the server but attempts to determine whether the operation would otherwise be successful.
- LDAP post-read control - This is a pair of request and response controls that may be used to retrieve an entry as it appeared immediately after performing an add, modify, or modify DN operation.
- LDAP pre-read control - This is a pair of request and response controls that may be used to retrieve an entry as it appeared immediately before performing a delete, modify, or modify DN operation.
- Manage DSA IT control - This is a request control that may be used to request that the server treat smart referrals as regular entries rather than as referrals.
- Matched values control - This is a request control that may be used to request that entries returned from a search operation only include values matching a given filter.

- Persistent search control - This is a request control that may be used to receive notification whenever an entry matching a given set of criteria is updated in the server.
- Proxied authorization control - This is a request control that may be used to request that an operation be performed under the authorization of another user.
- Server-side sort control - This is a request control that may be used to request that the server sort the results before returning them to the client.
- Simple paged results control - This is a request control that may be used to request that the server retrieve only a subset of the results, and when used repeatedly can allow the client to page through the result set.
- Virtual list view control - This is a pair of request and response controls that may be used to retrieve an arbitrary page of search results from the server.

An LDAP control is defined as follows:

```
Control ::= SEQUENCE {
     controlType              LDAPOID,
.... criticality              BOOLEAN DEFAULT FALSE,
.... controlValue             OCTET STRING OPTIONAL }
```

A control includes these elements:

- An OID that specifies the type of control.
- A criticality, which indicates whether the control should be considered a critical part of the operation (that is, if the server cannot process the control, the operation should fail).
- An optional value, which can be used to provide additional information about the way the control should be processed.

CRAM-MD5 SASL mechanism

The CRAM-MD5 SASL mechanism provides a way for clients to authenticate to the Directory Server with a username and password in a manner that does not expose the clear-text password, so it is significantly safer than simple authentication or the PLAIN SASL mechanism when the connection between the client and the server is not secure.

The CRAM-MD5 SASL mechanism is described in the draft-ietf-sasl-crammd5-10 internet draft. The process is as follows:

1. The client sends an LDAP message to the server with a bind request protocol op type using an authentication type of `SASL` with a mechanism name of `CRAM-MD5` and no credentials.
2. The server sends a bind response message back to the client with a result code of 14 (SASL bind in progress) and a server SASL credentials element including randomly-generated data (the challenge).

3. The client responds with a second SASL bind request message to the server with a mechanism name of `CRAM-M5`, and this time provides SASL credentials containing the authentication ID used to identify the user and an MD5 digest that is computed by combining the server-provided challenge with the clear-text password.
4. The server uses the authentication ID to identify the user, and then retrieves the clear-text password for that user (if the clear-text password cannot be obtained, then authentication will fail) and uses it to determine whether the provided digest is valid. The server will then send an appropriate response to the client (usually with a result of either `success` or `invalid credentials`) indicating whether the authentication was successful.

The CRAM-MD5 SASL mechanism is very similar to DIGEST-MD5, but it is somewhat weaker because CRAM-MD5 only includes random data from the server whereas DIGEST-MD5 includes random data from both the client and the server. DIGEST-MD5 also provides a provision for ensuring connection integrity and/or confidentiality, which CRAM-MD5 does not offer.

crypt algorithm

See UNIX crypt algorithm.

D

database

A database is a repository that is used for storing information. In the directory server, databases are used as the mechanism for storing data in a back end. The primary database used by the directory server is the Berkeley DB Java Edition, although it is possible to create other back ends with different backing stores.

database cache

The database cache is a portion of memory that is reserved for holding content from the underlying database. Whenever an attempt is made to retrieve information from the database, the database will first check this cache before going to disk. The database cache can help significantly improve performance by avoiding costly disk I/O.

The database cache may be used either instead of or in addition to the server's entry cache. The database cache frequently creates a more compact representation of the data (which means that more data can be held in the cache in systems with limited memory), but the entry cache generally holds data in a format that can be more efficiently used by the server.

debug log

The debug log provides a mechanism for obtaining information that may be used for debugging problems that might occur in the server. Debug information is generally data that is useful only in the event of a problem, and is frequently too voluminous to maintain under normal operations. The debug log may be used to report information like:

- Detailed information about exceptions thrown within the server
- Information about data read from or written to network clients
- Information about information read from or written to the database
- Information about decisions made in areas like access control or password policy processing

delete operation

The LDAP delete operation can be used to remove an entry from the server (or when used in conjunction with the subtree delete control, a subtree). The delete request protocol op is defined as follows:

```
DelRequest ::= [APPLICATION 10] LDAPDN
```

The request includes only the DN of the entry to delete.

The response to an LDAP delete operation is an LDAP result element as defined below:

```
DelResponse ::= [APPLICATION 11] LDAPResult
```

deprecated password storage scheme

A deprecated password storage scheme is a password storage scheme that is available for use in the server, but is intended primarily for transitional use. If a user has a password encoded with a deprecated storage scheme, then the user will be allowed to authenticate but the password will be re-encoded using the set of default storage schemes defined in the password policy.

This mechanism is primarily intended for cases in which data has been migrated into the directory server from another server uses a password storage scheme that you do not want to continue using (for example, because it is weaker than the default schemes). As users authenticate to the server, their passwords will be transitioned from the deprecated schemes to the default schemes.

dereference policy

The dereference policy is an element of a search quest that specifies how the server should handle alias entries that may be encountered during search processing. Allowed alias dereference policy values include:

neverDerefAliases	The server should not attempt to dereference any aliases that it encounters during search processing.
derefInSearching	The server should dereference any entries within the scope of the search operation to determine whether they match the search criteria. The entry specified as the search base DN will not be dereferenced.
derefFindingBaseObj	The server should dereference the entry referenced as the search base DN if it is an alias, but any other alias entries within the scope of the search operation will not be dereferenced.
derefAlways	The server will dereference any alias entries within the scope of the search operation and will also dereference the base entry if it is an alias.

DIGEST-MD5 SASL mechanism

The DIGEST-MD5 SASL mechanism provides a way for clients to authenticate to the Directory Server with a username and password in a manner that does not expose the clear-text password, so it is significantly safer than simple authentication or the PLAIN SASL mechanism when the connection between the client and the server is not secure.

The DIGEST-MD5 SASL mechanism is described in RFC 2831, but a revised specification is contained in draft-ietf-sasl-rfc2831bis. The process is as follows:

1. The client sends an LDAP message to the server with a bind request protocol op type using an authentication type of `SASL` with a mechanism name of `DIGEST-MD5` and no credentials.
2. The server sends a bind response message back to the client with a result code of 14 (SASL bind in progress) and a server SASL credentials element including, among other things, some randomly-generated data (the nonce).
3. The client takes the nonce provided by the server, and some randomly generated data of its own (the cnonce), an authentication ID, an optional authorization ID, the user's clear-text password, and some other information and uses that to create an MD5 digest. The client then sends a second bind request message including that digest and some other clear-text information back to the server.
4. The server uses the authentication ID to identify the user, and then retrieves the clear-text password for that user (if the clear-text password cannot be obtained, then authentication will fail) and uses it to determine whether the provided digest is valid. The server will then

send an appropriate response to the client (usually with a result of either `success` or `invalid credentials`) indicating whether the authentication was successful.

5. If the client requested a quality of protection (QoP) value indicating that the connection should be protected with integrity and/or confidentiality, then the server will initiate the necessary negotiation with the client. Note that at the present time, the directory server does not support the use of the DIGEST-MD5 mechanism with the use of integrity or confidentiality protection.

The DIGEST-MD5 SASL mechanism is very similar to CRAM-MD5, but it is somewhat strong because CRAM-MD5 includes only random data from the server whereas DIGEST-MD5 includes random data from both the client and the server. DIGEST-MD5 also provides a provision for ensuring connection integrity and/or confidentiality, which CRAM-MD5 does not offer.

directory information tree

The directory information tree, or DIT, refers to the hierarchical structure of the data in a Directory Server. The DIT contains one or more naming contexts, which are the base entries for the server, and every other entry is descended from one of those naming context entries. That is, a naming context entry is special in that it does not have a parent entry.

Consider a scenario, where the entry `dc=example,dc=com` is the naming context, and it has two immediate children, with DNs of `ou=People,dc=example,dc=com` and `ou=Groups,dc=example,dc=com`, respectively, and each of those entries has its own subordinate entries. There is no predefined limit to the maximum depth of a directory tree, and any entry can potentially have one or more subordinate entries. An entry that does not contain any subordinates is said to be a leaf entry, and any entry that has at least one subordinate entry is called a non-leaf entry.

directory manager

The term `directory manager` is a common name used to refer to a root DN user in the Directory Server. It is so named because the default root user typically uses a bind DN of `cn=Directory Manager`. Unlike many other types of directory servers, the directory server allows multiple root DNs to be defined, although the default root DN is still `cn=Directory Manager`.

directory server

A directory server is a type of network daemon that stores data in a manner accessible to external clients. Directory servers typically use LDAP or DSML for communicating with clients, although some servers use other protocols like DAP or NDS.

Directory servers store data in a hierarchical form (called the directory information tree) and provide the ability for clients to interact with that information, including:

- search operations, which make it possible to find all entry matching a given set of criteria
- add operations, which make it possible to add new entries to the server
- delete operations, which make it possible to remove entries from the server
- modify operations, which make it possible to update existing information in the server
- modify DN operations, which make it possible to rename entries in the server
- bind operations, which make it possible to authenticate users to the server
- compare operations, which make it possible to determine whether entries have a particular attribute-value pair

The directory server uses LDAPv3 for communicating with network clients, and provides a DSML gateway that can be used to handle DSML requests.

directory server agent

A directory server agent (DSA) is a single instance of a directory server.

Directory Services Markup Language

The Directory Services Markup Language (DSML) is a protocol that may be used to communicate with directory servers. DSML is an alternative to LDAP, and uses an XML-based representation of requests and responses instead of the ASN.1 BER encoding that LDAP uses.

In general, DSML is seen as a relatively weak alternative to LDAP because it provides very little benefit and incurs a significant cost because the XML representation is much more verbose and expensive to process when compared with the BER encoding that LDAP uses. In most cases, it is recommended that LDAP be used instead of DSML to interact with the server.

distinguished name

A distinguished name (often referred to as a `DN`) is a string that uniquely identifies an entry in the Directory Server. It consists of zero or more relative distinguished name (RDN) components that identify the location of the entry in the DIT. An entry's distinguished name can be thought of as a kind of an analog to an absolute path in a filesystem in that it specifies both the name and hierarchical location.

The RDN components for a distinguished name are separated by commas and are ordered from right to left. The rightmost components of a DN are closest to the server's naming context, and the leftmost components are closest to the leaf entry. That is, if you think of a directory hierarchy as a kind of pyramid with the naming context at the top and the branches descending downward, then the order of RDN components in a DN are listed from bottom to top.

Even though a DN consists of a series of RDN components, when one refers to an entry's RDN, then it is a reference to the leftmost RDN component. The attributes contained in an entry's RDN must also be contained in that entry.

In a DIT, the top entry is the naming context and its DN is `dc=example,dc=com`. To conserve space, only the RDNs of the subordinate entries are displayed, but the full DNs can be obtained by appending the RDN components from bottom to top. For example, the DN of the leftmost entry on the bottom row would be `uid=ann,ou=People,dc=example,dc=com`.

See RFC 4514 for more information about LDAP distinguished names and the way in which they should be represented as strings.

distribution

Distribution is a Oracle Unified Directory proxy deployment type in which data is split into *partitions*. The split of data is determined by a distribution algorithm.

DIT

See directory information tree.

DIT content rule

A DIT content rule is a schema element that specifies which auxiliary object classes are allowed to be used with an entry, as well as which attribute types are required, allowed, and prohibited for use with an entry, based on its structural object class.

The components of a DIT content rule definition include:

- The numeric OID of the structural object class with which the DIT content rule is associated.
- An optional set of names for the DIT content rule.
- An optional set of auxiliary object class names or OIDs for the auxiliary classes that are allowed to be used with entries containing the associated structural class.
- An optional set of attribute type names or OIDs for attribute types that are required to be present in entries with the associated structural class. These attributes will be required even if they are not allowed by any of the object classes in the entry.
- An optional set of attribute type names or OIDs for attribute types that may optionally be present in entries with the associated structural class. These attributes will be allowed even if they are not allowed by any of the object classes in the entry.
- An optional set of attribute type names or OIDs for attribute types that are prohibited to be present in entries with the associated structural class. These attributes will be prohibited even if they are allowed by any of the object classes in the entry.

The set of DIT content rules defined in the server may be determined by retrieving the `dITContentRules` attribute of the subschema subentry. For more information about DIT content rules, see “Understanding DIT Content Rules” in *Oracle Fusion Middleware Architecture Reference for Oracle Unified Directory*.

DIT structure rule

A DIT structure rule is a schema element that may be used to define the hierarchical relationships between entries. In particular, it defines the kinds of parent entries (based on their structural object class) that an entry with a given structural class is allowed to have.

The components of a DIT structure rule definition include:

- An integer rule ID value that is used to uniquely identify the rule.
- An optional set of names for the DIT structure rule.
- The name or OID of the name form with which the DIT structure rule is associated. The name form in turn links the DIT structure rule to a structural object class.
- An optional set of superior rule IDs. If a set of superior rules is defined, then they are used to define the structural classes below which the structural class associated with the rule's name form is allowed to exist.

The set of DIT structure rules defined in the server may be determined by retrieving the `dITStructureRules` attribute of the subschema subentry. For more information about DIT structure rules, see the "Understanding DIT Structure Rules" in *Oracle Fusion Middleware Architecture Reference for Oracle Unified Directory*.

DN

See distinguished name.

DSA

See directory server agent.

DSA-specific entry

A DSA-Specific Entry (DSE) is a special type of entry that provides information about a directory server agent (DSA), which is a synonym for directory server.

LDAP defines a special entry called the root DSE that provides information about the information contained in the server and the types of operations that it supports.

DSE

See DSA-specific entry.

DSML

See Directory Services Markup Language.

DSML gateway

A DSML gateway (or DSML-to-LDAP gateway) is a special type of network daemon that is used to translate between DSML and LDAP. In general, a DSML gateway accepts DSML requests from clients, converts them to LDAP requests that it forwards to a directory server for processing. It then translates the LDAP response from the directory server back to DSML to return to the client.

The directory server supports DSML through a DSML gateway, which is implemented as a Web application that can run in an application server.

duration

Certain configuration properties take a duration as their allowed value.

A duration includes an integer, and a unit, specified in weeks (w), days (d), hours (h), minutes (m), seconds (s), or miliseconds (ms), or some combination with multiple specifiers. For example, you can specify one week as `1w`, `7d`, `168h`, `10080m`, or `604800s`. Or you can specify ten and a half days as `1w3d12h0m0s`.

Not all properties that require a duration support all duration specifiers (`w`, `d`, `h`, `m`, `s`, and `ms`).

A duration property can also include the following:

base unit	Specifies the minimum granularity that can be used to specify duration property values. For example, if the base unit is in seconds, values represented in milliseconds are not permitted.
maximum unit (optional)	Specifies the largest duration unit that can be used to specify duration property values. Values presented in units greater than this unit are not permitted.
lower limit	Specifies the smallest duration permitted by the property.
upper limit (optional)	Specifies the largest duration permitted by the property.
unlimited duration	Certain properties allow you to specify an unlimited duration. This is represented using the decoded value, -1, or the encoded string value `unlimited`.

dynamic group

A dynamic group is a type of group in the directory server that defines its membership using a set of search criteria in the form of an LDAP URL, as opposed to a static group in which the DNs of the members are explicitly specified.

Dynamic groups provide an efficient way to manage groups with very large numbers of members. They are much more scalable than static groups, and their membership is automatically updated as entries change so that the match or no longer match the group criteria.

E

entry

An entry is the structure that holds information in a directory server. It consists of the following components:

- A distinguished name that uniquely identifies the entry among all other entries in the server.
- A collection of object class values that are used to govern the contents of the entry.
- A collection of attribute that contain the actual data for the entry.

An entry must always have exactly one structural object class that defines what type of entry it is. It may have zero or more auxiliary object classes that may be used identify other characteristics for the entry. Together, the structural and auxiliary classes define a set of required attributes, which must be present in the entry, and optional attributes, which may be included in the entry but are not required.

entry cache

The entry cache is a mechanism that uses system memory for holding entries in a manner that may be quickly accessed so that it is not necessary to decode them from the database whenever they are needed. Entry caching mechanisms are particularly effective when used with applications that access the same entry multiple times in a sequence of operations. For example, an application which first searches to find a user entry and then binds as that user to verify a password, which is a very common usage pattern.

The entry cache may be used either instead of or in addition to the server's database cache. The database cache generally uses a more compact representation of the data, but the entry cache generally holds data in a format that can be more efficiently used by the server.

Unlike the database cache which is maintained by the underlying database, the entry cache is managed by the directory server itself. There are a number of different entry cache implementations that may be used.

entry change notification control

The entry change notification control is a control that is included in search result entries returned to clients in response to a search operation that uses the persistent search control. This control contains additional information about the change made to the entry, including the type of change made, the change number (which corresponds to an item in the server's change log, if the server supports a change log), and, if the entry was renamed, the old DN of the entry. The control is described in draft-ietf-ldapext-psearch-03 and has an OID of 2.16.840.1.113730.3.4.7.

The control is defined as follows:

```
EntryChangeNotification ::= SEQUENCE {
          changeType ENUMERATED {
               add              (1),
               delete           (2),
               modify           (4),
               modDN            (8)
          },
          previousDN   LDAPDN OPTIONAL,     -- modifyDN ops. only
          changeNumber INTEGER OPTIONAL     -- if supported
}
```

entryDN

An entryDN is an operational attribute that provides a copy of the entry's current DN. Because a DN is not an attribute of the entry, it cannot be used to perform attribute value assertions. The entryDN provides a mechanism to access an entry's DN and is described in RFC 5020.

entry ID

An entry ID is an integer value that is used to uniquely identify an entry in the Directory Server back end. Although the entry's distinguished name could be used for this purpose, the numeric entry ID is much more compact and more efficient to decode, so it is more appropriate for widespread use.

The entry ID is used as the key to the actual entry data in the id2entry database, and it is used in ID lists to identify entries matching the associated index key.

entryUUID

An entryUUID is a universally unique identifier that is contained in the `entryUUID` operational attribute and is assigned to each entry in the directory server. It is defined in RFC 4530 and it is intended to be a unique identifier that will not change over the life of the entry (as opposed to the distinguished name, which can change as a result of a modify DN operation). Because of the greater stability of the entryUUID, it is used by the replication subsystem to track entries even if the DN does change.

equality index

An equality index is a type of indexwhich is used to identify efficiently which entries are exactly equal to a given assertion value. An equality index may only be maintained for attributes that have a corresponding equality matching rule. That matching rule will be used to normalize values to use as index keys, and the value for that key will be the ID list containing the entry IDs of the entries with values that are equal to that normalized value.

equality search filter

An equality search filter is a type of search filter that can be used to identify entries that contain a specific value for a given attribute. The server will use an equality matching rule to make the determination.

The string representation of an LDAP equality filter comprises an opening parenthesis followed by the attribute name, an equal sign, the attribute value, and the closing parenthesis. For example, an equality filter of `(uid=john.doe)` will match any entry in which the `uid` attribute contains a value of `john.doe`.

error log

The error log provides a mechanism for reporting errors, warnings, and other significant events that happen in the life of the server. Each message written to the error log will include a category (indicating the area of the server in which the message was generated) and severity (indicating the relative importance of the message), along with an integer value that uniquely identifies the associated message string.

export

See LDIF export.

extended operation

The LDAP extended operation provides a degree of extensibility to the LDAP protocol by allowing clients to request operations not defined in the core protocol specification. Examples of LDAP extended operations include:

Cancel extended operation	This operation may be used to cancel a previously-requested operation.

Password Modify extended operation	This operation may be used to change a user password.
StartTLS extended operation	This operation may be used to initiate a secure communication channel over an existing connection.
“Who Am I?” extended operation	This operation may be used to determine the authorization identity associated with the client connection.

The extended request protocol op is defined as follows:

```
ExtendedRequest ::= [APPLICATION 23] SEQUENCE {
     requestName      [0] LDAPOID,
     requestValue     [1] OCTET STRING OPTIONAL }
```

The elements of the extended request include:

- The OID that is used to indicate the type of operation to perform.
- An optional value containing additional information to use during the course of processing the request.

The response to an LDAP extended operation is defined as follows:

```
ExtendedResponse ::= [APPLICATION 24] SEQUENCE {
     COMPONENTS OF LDAPResult,
     responseName     [10] LDAPOID OPTIONAL,
     responseValue    [11] OCTET STRING OPTIONAL }
```

The extended response includes these elements:

- The elements of the LDAP result object.
- An optional OID used to indicate the type of response.
- An optional encoded value with additional information to include in the response.

extensible match index

An extensible match index is a type of index that is used to help accelerate search operations using an extensible match filter. Index keys are values that have been normalized using a specified matching rule, and the corresponding ID list contains the entry IDs for all entries that match the value according to that matching rule.

extensible match search filter

An extensible match search filter is a type of search filter that can be used to identify matching entries using a specified matching rule.

An extensible matching filter contains the following components:

- The OID of the matching rule to use for the determination. This is an optional element, and if it isn't provided then the attribute type must be given and its default equality matching rule will be used.
- The name of the attribute type that will be targeted. If this is not provided, then all attributes contained in the entry will be examined.
- A flag that indicates whether the matching should be performed against the attributes of the entry's DN *and* the attributes contained in the entry.
- An assertion value that should be used as the target for the matching rule.

The string representation of an LDAP extensible match filter comprises the following components in order:

- An opening parenthesis
- The name of the attribute type, or an empty string if none was provided
- The string `:dn` if the `dnAttributes` flag is set, or an empty string if not
- If a matching rule ID is available, then a string composed of a colon followed by that OID, or an empty string if there is no matching rule ID
- The string `:=`
- The string representation of the assertion value
- A closing parenthesis

EXTERNAL SASL mechanism

The EXTERNAL SASL mechanism provides a way for clients to authenticate to the Directory Server using information that is available outside of the communication performed at the LDAP protocol level. The most common use of EXTERNAL authentication (and at present, the only form that the directory server supports) is for the server to identify the client based on a certificate that the client presented during SSL or StartTLS negotiation. The Directory Server will use a certificate mapper to map the client's certificate to a user in the directory, and may optionally perform additional validation (for example, ensuring that the presented certificate actually exists in the user's entry).

F

failover algorithm

A load balancing algorithm in which all client requests are sent to a main remote LDAP data source. If the main remote LDAP goes down, the request are forwarded to a secondary remote LDAP server, and so on. This ensures the continuation of the service after failure of one or more remote LDAP servers.

false filter

See LDAP false filter.

G

generalized time

Generalized time is a form at may be used to represent time stamps, along with time zone information. A generalized time value contains the following components:

- Four digits to signify the year.
- Two digits to signify the month (01 for January, 02 for February, ..., 12 for December).
- Two digits to signify the day of the month (01 through 28/29/30/31 depending on the month and whether it's a leap year).
- Two digits to signify the hour of the day (00 for midnight through 23 for 11 pm).
- An optional two digits that specify the minute of the hour (between 00 and 59).
- An optional two digits that specify the second of the minute (between 00 and 59, or 60 for leap seconds). This may only be included if the time stamp value also contains the minute of the hour.
- An optional period followed by one or more digits that specify the fraction of a second. This may only be included if the time stamp value contains minute and second information.
- A time zone indicator. This may be either the capital letter `Z` to indicate that the value is in the UTC time zone, or a plus or minus sign followed by two or four digits that specify the offset from UTC time zone.

An example of a time stamp in a generalized time format is `20070508200557Z`, which specifies a time (in the UTC time zone) of 8:05:57 PM on May 28, 2007. An equivalent value in the United States central daylight savings time (a five hour offset from UTC) would be `20070508150557-0500`.

get effective rights control

The get effective rights control is a type of control that can be used to determine the rights that a given user has when interacting with a given entry. The control has an OID of 1.3.6.1.4.1.42.2.27.9.5.2 and uses the following definition:

```
GetRightsControl ::= SEQUENCE {
     authzId    authzId
     attributes  SEQUENCE OF AttributeType
}

-- Only the "dn:DN form is supported.
```

For an example of using this control in a search request, see "To Search Using the Get Effective Rights Control" in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

global index

In the Oracle Unified Directory proxy, the global index maps the data entries to the *distribution partition* where the data is stored. Global indexes map a specific attribute (such as `telephonenumber`). For example, the global index could map `telephonenumber=5551212` to distribution partition 1, while `telephonenumber=4441212` to partition 2.

global index catalog

A global index catalog contains one or more *global indexes*. A global index catalog can be used with a Oracle Unified Directory proxy distribution deployment, in order to diminish the need for broadcasts, since the values of some attributes are mapped to the partition in which the entry is held.

greater than or equal to search filter

An greater or equal search filter is a type of search filter that can be used to identify entries that contain a specific value for a given attribute that is greater than or equal to the provided assertion value. The server will use an ordering matching rule to make the determination.

The string representation of an LDAP greater or equal search filter comprises an opening parenthesis followed by the attribute name, a greater than sign, an equal sign, the assertion value, and the closing parenthesis. For example, a greater or equal filter of `(createTimestamp>=20070101000000Z)` will match any entry that has a `createTimestamp` value that is greater than or equal to `20070101000000Z`.

group

A group is a special type of entry in the Directory Server that is used to represent a set of users in the server. Groups may be used within the server in a number of different ways, like access control and virtual attributes, and they may also be used by clients for various purposes.

There are several different types of groups defined in the server, including:

- Static groups provide an explicit list of members
- Dynamic groups obtain their membership information from a set of search criteria
- Virtual static groups appear to be static groups but obtain their membership information from another type of group, like a dynamic group

GSSAPI SASL mechanism

The GSSAPI SASL mechanism provides a way for clients to authenticating to the Directory Server using a Kerberos V5 session. Kerberos is a protocol that is commonly used for single sign-on purposes, and provides the option of using integrity and/or confidentiality to protect the communication between the client and the server (although the directory server does not at present support GSSAPI for protecting network content but only for authenticating clients).

The GSSAPI SASL mechanism is described in RFC 4752.

I

ID list

An ID list is used as the value of a Directory Server index. It contains a set of entry ID for all entries that match the associated index key.

In some cases, an ID list can have a special value that indicates that there are more entries matching the index key than allowed by the index entry limit. In that case, the index key will no longer be maintained.

id2entry database

The id2entry database is a type of database that maps an entry ID to the contents of the corresponding entry. The entry ID is used in ID lists within indexes.

identity mapper

An identity mapper provides logic that can be used to map an authentication ID or authorization ID value to a corresponding user entry. Identity mappers are used in conjunction with a number of SASL mechanisms, as well as the proxied authorization control and the Password Modify extended operation.

idle account lockout

Idle account lockout is a part of the Directory Server password policy that may be used to lock user accounts that remain unused for a significant period of time. It requires that the last login time feature be enabled so that user authentication times will be recorded, and any bind operation by a user that has not authenticated within a specified period of time will be rejected.

If a user's account has been locked due to remaining idle for too long, then it may be unlocked by an administrative password reset.

in-core restart

An in-core restart is a process by which the server may be restarted without actually existing the JVM used to run the server. It can be used to apply any change that requires a server restart other than one that requires the modification of a JVM argument. An in-core restart may be faster than stopping and re-starting the server process, and it has the added benefit of maintaining the JIT cache that has been accumulated from observing processing performed within the JVM.

index

An index is a mechanism used by the Directory Server database that can be used to efficiently find entries matching search criteria. An index maps a key to an ID list, which is the set of entry ID for the entries that match that index key.

The directory server uses six primary types of indexes:

- Approximate indexes are used to identify entries containing attribute values approximately equal to a given assertion value.
- Equality indexes are used to identify entries containing an attribute value that exactly matches a given assertion value.
- Extensible match indexes are used to identify entries that match a given extensible match filter. This index is not currently supported.
- Ordering indexes are used to identify entries that have values that are greater than or equal to, or less than or equal to, a given assertion value.
- Presence indexes are used to identify entries that contain at least one value for a given attribute.
- Substring indexes are used to identify entries that contain an attribute value matching a given substring assertion.

index entry limit

The index entry limit is a configuration limit that can be used to control the maximum number of entries that is allowed to match any given index key (that is, the maximum size of an ID list). This provides a mechanism for limiting the performance impact for maintaining index keys that match a large percentage of the entries in the server. In cases where large ID lists might be required, performing an unindexed search can often be faster than one that is indexed.

The index entry limit in the directory server is analogous to the ALL IDs threshold in Oracle Directory Server Enterprise Edition.

intermediate response

See LDAP intermediate response.

Internet Draft

An Internet Draft is a form of specification defined through the IETF. Internet drafts are short-lived specifications that typically go through multiple revisions, and may change significantly between revisions. Internet Drafts that reach a point of stability may be promoted to requests for comments. Other drafts may stagnate and become no longer maintained, although in some cases they may still describe viable functionality that is worth implementing in the server.

J

Java Management Extensions

Java Management Extensions (JMX) is a framework is a Java technology that can be used for accessing monitoring and configuration information.

Oracle Unified Directory uses JMX for publishing information from monitor entry. It also uses the JMX notification mechanism for administrative alerts in the event of significant problems or events in the server.

JMX

See "Java Management Extensions" on page 62.

K

key manager provider

A key manager provider is a component of the server that can provide access to private key information for server certificate.

The key manager providers available for use in the server include the following:

- A mechanism for accessing key information in a JKS keystore
- A mechanism for accessing key information in a PKCS#12 file
- A mechanism for accessing key information in a PKCS#11 token

L

last login time

The last login time feature of the Directory Server is a mechanism that can be used to write the time that the user last authenticated to the server using a bind operation. The last login time may be written to a specified attribute with a user-defined format.

Note that in many servers, it may be desirable to define the last login time format to contain only the date but not the time of day. If this format is used, then the value will be only updated once per day, thereby reducing the potential impact on performance for users that authenticate several times throughout the day.

The last login time may be maintained for informational purposes, but it can also be used to enable the idle account lockout feature.

lastmod plug-in

The lastmod plug-in is a pre-operation idle account lockout that can be used to add the `creatorsName` and `createTimestamp` attributes to an entry as part of an add operation, or update the `modifiersName` and `modifyTimestamp` attributes in an entry as part of a modify operation or modify DN operation operation.

LDAP assertion control

The LDAP assertion control is a type of control that may be used to perform an operation only if the target entry matches a given assertion filter. It may be used in conjunction with compare operation, delete operation, modify operation, modify DN operation, and search operation.

The LDAP assertion control is described in RFC 4528 and has an OID of `1.3.6.1.1.12`. The value of the control should be encoded as an LDAP search filter.

For an example of using this control in a search request, see "To Search Using the LDAP Assertion Control" in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

ldapcompare command

The ldapcompare command-line may be used to request LDAP compare operation.

See "ldapcompare" in *Oracle Fusion Middleware Command-Line Usage Guide for Oracle Unified Directory* for detailed information on using this command.

LDAP Data Interchange Format

The LDAP Data Interchange Format (LDIF) is a mechanism form representing directory data in text form. The LDIF specification is contained in RFC 2849 and describes a format not only for representing directory data but also a mechanism for making changes to that data.

In general, an LDIF record consists of a series of name-value pairs. The name can be followed by a single colon, zero or more spaces, and associated value, or it can be followed by two colons, zero or more spaces, and the base64 encoding representation of the value. Each name-value pair is given on a separate line, and long lines may be wrapped onto two or more lines using an end-of-line character followed by exactly one space at the beginning of the next line. LDIF records should be separated from each other by at least one blank line. Any line that begins with an octothorpe (#) character will be treated as a comment and ignored.

For an LDIF representation of an entry, the first line should contain the distinguished name of the entry. The remaining lines of the LDIF record will represent the attribute of the entry, with the attribute description used as the name. Multivalued attributes will be represented with a separate line per value.

The following provides an example of a user entry represented in the LDAP Data Interchange Format:

```
dn: uid=john.doe,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: john.doe
givenName: John
sn: Doe
cn: John Doe
mail: john.doe@example.com
userCertificate;binary:: MIIB5TCCAU6gAwIBAgIERloIajANBgkqhkiG9w0BAQUFADA3M
 QswCQYDVQQGEwJVUzEVMBMGA1UEChMMRXhhbXBsZSBDb3JwMREwDwYDVQQDEwhKb2huIERvZT
 AeFw0wNzA1MjcyMjM4MzRaFw0wNzA4MjUyMjM4MzRaMDcxCzAJBgNVBAYTAlVTMRUwEwYDVQQ
 KEwxFeGFtcGxlIENvcnAxETAPBgNVBAMTCEpvaG4gRG9lMIGfMA0GCSqGSIb3DQEBAQUAA4GN
 ADCBiQKBgQCWNZB4qs1UvjYgvGvB9udmiUi4X4DeaSm3o0p8PSwpOFxSqgWdSwKgUugZ1EJVy
 YoakljDFsJ0GVown+dIB24V4ozNs6wa0YotIKTV2AcySQkmzzP3e+OnE9Aa1wlB/PVnh1CFLg
 k1UOoruLE10bac5HA8QiAmfNMorU26AwFTcwIDAQABMA0GCSqGSIb3DQEBBQUAA4GBAGrzMKN
 bBRWn+LIfYTfqKYUc258XVbhFri1OV0oF82vyvciYWZzyxLc52EPDsymLmcDh+CdWxy3bVkjd
 Mg1WEtMGr1GsxOVi/vWe+kT4tPhinnB4Fowf8zgqiUKo9/FJN26y7Fpvy1IODiBInDrKZRvNf
 qemCf7o3+Cp00OmF5ey
```

```
userPassword: {SSHA}s4Bd9M0tCpRDr8/U+IXetRcAbd8bJY3AFKsn+A==
```

To represent an LDAP add operation in LDIF, the format is exactly the same as to represent an entry, with the exception that the line immediately after the DN should indicate a `changetype` of `add`, as shown in the following example:

```
dn: uid=john.doe,ou=People,dc=example,dc=com
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: john.doe
givenName: John
sn: Doe
cn: John Doe
mail: john.doe@example.com
userCertificate;binary:: MIIB5TCCAU6gAwIBAgIERloIajANBgkqhkiG9w0BAQUFADA3M
 QswCQYDVQQGEwJVUzEVMBMGA1UEChMMRXhhbXBsZSBDb3JwMREwDwYDVQQDEwhKb2huIERvZT
 AeFw0wNzA1MjcyMjM4MzRaFw0wNzA4MjUyMjM4MzRaMDcxCzAJBgNVBAYTAlVTMRUwEwYDVQQ
 KEwxFeGFtcGxlIENvcnAxETAPBgNVBAMTCEpvaG4gRG9lMIGfMA0GCSqGSIb3DQEBAQUAA4GN
 ADCBiQKBgQCWNZB4qs1UvjYgvGvB9udmiUi4X4DeaSm3o0p8PSwpOFxSqgWdSwKgUugZ1EJVy
 YoakljDFsJ0GVown+dIB24V4ozNs6wa0YotIKTV2AcySQkmzzP3e+OnE9Aa1wlB/PVnh1CFLg
 k1UOoruLE10bac5HA8QiAmfNMorU26AwFTcwIDAQABMA0GCSqGSIb3DQEBBQUAA4GBAGrzMKN
 bBRWn+LIfYTfqKYUc258XVbhFri1OV0oF82vyvciYWZzyxLc52EPDsymLmcDh+CdWxy3bVkjd
 Mg1WEtMGr1GsxOVi/vWe+kT4tPhinnB4Fowf8zgqiUKo9/FJN26y7Fpvy1IODiBInDrKZRvNf
 qemCf7o3+Cp00OmF5ey
userPassword: password
```

To represent an LDAP delete operation in LDIF, the format is simply a line containing the DN of the entry followed by a line indicating a `changetype` of `delete`, like:

```
dn: uid=john.doe,ou=People,dc=example,dc=com
changetype: delete
```

To represent an LDAP modify operation in LDIF, the format is a little more complex. The first line should contain the DN of the entry, and the second should contain a `changetype` of `modify`. The third line should specify the attribute modification type (add, delete, replace, or increment) followed by the attribute description, and there may be additional lines that specify specific values for that change, with the name portion being the attribute description and the value being the corresponding attribute value. There may be multiple attribute modifications described in a single modify change record, with each of them separated by a line containing only a dash, as shown in the following example:

```
dn: uid=john.doe,ou=People,dc=example,dc=com
changetype: modify
replace: userPassword
userPassword: newpassword
-
replace: description
description: This is the first description value
description: This is the second description value
```

To represent an LDAP modify DN operation in LDIF, the first line should contain the DN of the entry, and the second line should contain a `changetype` of `moddn`. The third line should have a name of `newrdn` with a value equal to the new RDN to assign to the entry, and the fourth should have a name of `deleteoldrdn` followed by a value of either 1 (if the `deleteOldRDN` flag should be `true`) or `0` (if it should be false). There can be an optional fifth line with a name of `newsuperior` and a value of the new superior DN if one is included in the request. For example:

```
dn: uid=john.doe,ou=People,dc=example,dc=com
changetype: moddn
newrdn: uid=johnathan.doe
deleteoldrdn: 1
```

ldapdelete command

The ldapdelete command may be used to request LDAP delete operation.

See the “ldapdelete” in *Oracle Fusion Middleware Command-Line Usage Guide for Oracle Unified Directory* for detailed information on using this command.

LDAP false filter

An LDAP false filter is a special type of OR search filter that does not contain any embedded filter components. It is called an “LDAP false filter” because it always evaluates to `false` and will never match any entry.

The string representation for an LDAP true filter is `(|)`. LDAP false filters are described in RFC 4526.

LDAP intermediate response

The LDAP intermediate response message is a special type of protocol op that allows the server to send additional messages providing information about the state of an operation before it has completed processing and the final response message is sent. Prior to the introduction of the intermediate response in RFC 3771, only search operations were allowed to send multiple responses.

The intermediate response protocol op is defined as follows:

```
IntermediateResponse ::= [APPLICATION 25] SEQUENCE {
          responseName     [0] LDAPOID OPTIONAL,
          responseValue    [1] OCTET STRING OPTIONAL }
```

At present, the directory server does not support any operations that make use of intermediate response messages.

LDAP message

The LDAP message is the fundamental protocol data unit for LDAP communication. It is the container that is used to hold all request and response elements.

The LDAP message is defined as shown in the following example:

```
LDAPMessage ::= SEQUENCE {
     messageID       MessageID,
     protocolOp      CHOICE {
          bindRequest           BindRequest,
          bindResponse          BindResponse,
          unbindRequest         UnbindRequest,
          searchRequest         SearchRequest,
          searchResEntry        SearchResultEntry,
          searchResDone         SearchResultDone,
          searchResRef          SearchResultReference,
          modifyRequest         ModifyRequest,
          modifyResponse        ModifyResponse,
          addRequest            AddRequest,
          addResponse           AddResponse,
          delRequest            DelRequest,
          delResponse           DelResponse,
          modDNRequest          ModifyDNRequest,
          modDNResponse         ModifyDNResponse,
          compareRequest        CompareRequest,
          compareResponse       CompareResponse,
          abandonRequest        AbandonRequest,
          extendedReq           ExtendedRequest,
          extendedResp          ExtendedResponse,
          ...,
          intermediateResponse  IntermediateResponse },
     controls       [0] Controls OPTIONAL }
```

The LDAP message includes these elements:

- The message ID, which is the unique identifier that is used to correlate requests and responses. The client includes a message ID in the request, and all response messages for that request will have the same message ID.
- The protocol op, which is the container for the actual request or response.
- An optional set of control that can be used to provide additional information about the way that the request should be processed, or additional information about the response from the server.

LDAP modify DN operation

The LDAP modify DN operation can be used to change the distinguished name of an entry in the Directory Server. It can alter the RDN of the entry and/or it can move the entry below a new parent. If the target entry has subordinate entries, then it may be used to move or rename that subtree.

The modify DN request protocol op is defined as follows:

```
ModifyDNRequest ::= [APPLICATION 12] SEQUENCE {
     entry           LDAPDN,
     newrdn          RelativeLDAPDN,
     deleteoldrdn    BOOLEAN,
     newSuperior     [0] LDAPDN OPTIONAL }
```

The modify DN request includes these elements:

- The DN of the entry to rename and/or move.
- The new RDN to use for the entry. If the entry is simply to be moved below a new parent, then it may be the same as the current RDN.
- A flag that indicates whether the current RDN attribute values should be removed from the entry.
- An optional DN specifying the new parent for the entry.

The response to an LDAP modify DN operation is an LDAP result as defined as follows:

```
ModifyDNResponse ::= [APPLICATION 13] LDAPResult}
```

LDAP modify operation

The LDAP modify operation can be used to alter an existing entry in the Directory Server. The modify request protocol op is defined as follows:

```
ModifyRequest ::= [APPLICATION 6] SEQUENCE {
     object          LDAPDN,
     changes         SEQUENCE OF change SEQUENCE {
          operation       ENUMERATED {
               add     (0),
               delete  (1),
               replace (2),
               ...  },
          modification    PartialAttribute } }
```

The modify request includes these elements:

- The DN of the entry to modify
- One or more modification elements indicating the changes to make in the entry

The response to an LDAP modify operation is an LDAP result defined as shown here:

```
ModifyResponse ::= [APPLICATION 7] LDAPResult
```

ldapmodify command

The ldapmodify command may be used to request LDAP add, delete, modify, and modify DN operations.

See “ldapmodify” in *Oracle Fusion Middleware Command-Line Usage Guide for Oracle Unified Directory* for detailed information on using this command.

LDAP no-op control

The LDAP no-op control is a type of control that may be attached to an LDAP add operation, delete operation, modify operation, or modify DN operation to indicate that it should not actually make any change to the content in the server.

The LDAP no-op control is defined in draft-zeilenga-ldap-noop. This is a specification that is still in progress, but the directory server does provide basic support for this control using an OID of 1.3.6.1.4.1.4203.1.10.2. The control does not have a value.

The following example shows the use of the no-op control in an `ldapmodify` operation.

```
ldapmodify -h localhost -p 1389 -D "cn=directory manager" -w password \
-J 1.3.6.1.4.1.4203.1.10.2
dn: uid=aaltay,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: +1 995 589 3333

Processing MODIFY request for uid=aaltay,ou=People,dc=example,dc=com
MODIFY operation failed
Result Code:  16654 (No Operation)
Additional Information:  The modify operation was not actually performed in the
Directory Server back end because the LDAP no-op control was present in the request
```

LDAP post-read control

The LDAP post-read control is a type of control that may be attached to an LDAP add operation, modify operation, or modify DN operation operation to request that the server return a copy of the target entry exactly as it was at the end of the processing for that operation. It is one of the LDAP read entry controls defined in RFC 4527.

The post-read request control has an OID of 1.3.6.1.1.13.2, and the value should be encoded in the same way as the search attributes in a search operation. The response control has an OID of 1.3.6.1.1.13.2 (the same as the OID for the request control), and the value should be encoded in the same was as a search result entry.

The following example shows the use of the post-read control in an `ldapmodify` request:

```
$ ldapmodify -h localhost -p 1389 -D "cn=directory manager" -w password \
--postReadAttributes=telephoneNumber
dn: uid=aaltay,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: +1 995 589 3333

Processing MODIFY request for uid=aaltay,ou=People,dc=example,dc=com
MODIFY operation successful for DN uid=aaltay,ou=People,dc=example,dc=com
Target entry after the operation:
dn: uid=aaltay,ou=People,dc=example,dc=com
telephoneNumber: +1 995 589 3333
```

LDAP pre-read control

The LDAP pre-read control is a type of control that may be attached to an LDAP delete operation, modify operation, or modify DN operation operation to request that the server return a copy of the target entry exactly as it was immediately before the processing for that operation. It is one of the LDAP read entry controls defined in RFC 4527.

The pre-read request control has an OID of 1.3.6.1.1.13.1, and the value should be encoded in the same way as the search attributes in a search operation. The response control has an OID of 1.3.6.1.1.13.1 (the same as the OID for the request control), and the value should be encoded in the same was as a search result entry.

The following example shows the use of the pre-read control in an `ldapmodify` request:

```
$ ldapmodify -h localhost -p 1389 -D "cn=directory manager" -w password \
--preReadAttributes=telephoneNumber
dn: uid=aaltay,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: +1 995 589 4444

Processing MODIFY request for uid=user.199,ou=People,dc=exampele,dc=com
MODIFY operation successful for DN uid=aaltay,ou=People,dc=example,dc=com
Target entry before the operation:
dn: uid=aaltay.199,ou=People,dc=example,dc=com
telephoneNumber: +1 995 589 3333
```

LDAP result

The LDAP result element is a generic protocol op that is used for the responses of several types of LDAP operations. The basic definition for the LDAP result is as follows:

```
LDAPResult ::= SEQUENCE {
     resultCode         ENUMERATED {
          success                      (0),
          operationsError              (1),
          protocolError                (2),
          timeLimitExceeded            (3),
          sizeLimitExceeded            (4),
          compareFalse                 (5),
          compareTrue                  (6),
          authMethodNotSupported       (7),
          strongerAuthRequired         (8),
               -- 9 reserved --
          referral                     (10),
          adminLimitExceeded           (11),
          unavailableCriticalExtension (12),
          confidentialityRequired      (13),
          saslBindInProgress           (14),
          noSuchAttribute              (16),
          undefinedAttributeType       (17),
          inappropriateMatching        (18),
          constraintViolation          (19),
          attributeOrValueExists       (20),
          invalidAttributeSyntax       (21),
               -- 22-31 unused --
          noSuchObject                 (32),
          aliasProblem                 (33),
          invalidDNSyntax              (34),
               -- 35 reserved for undefined isLeaf --
          aliasDereferencingProblem    (36),
               -- 37-47 unused --
          inappropriateAuthentication  (48),
          invalidCredentials           (49),
          insufficientAccessRights     (50),
          busy                         (51),
          unavailable                  (52),
          unwillingToPerform           (53),
          loopDetect                   (54),
               -- 55-63 unused --
          namingViolation              (64),
          objectClassViolation         (65),
          notAllowedOnNonLeaf          (66),
          notAllowedOnRDN              (67),
          entryAlreadyExists           (68),
          objectClassModsProhibited    (69),
               -- 70 reserved for CLDAP --
          affectsMultipleDSAs          (71),
               -- 72-79 unused --
          other                        (80), ...  },
     matchedDN          LDAPDN,
     diagnosticMessage  LDAPString,
     referral           [3] Referral OPTIONAL }
```

The elements of the LDAP result are:

Result code	An integer value that provides generic information about the result of the operation. The definition above specifies several result codes, but a number of other values are defined in other specifications.
Matched DN	A DN value that may specify the DN of the closest superior entry found if the request specified an entry that did not exist. It may be an empty DN if the matched DN element is not appropriate for the response.
Diagnostic Message	A human-readable message that provides additional information about the result of the processing. It is typically used for error messages, but it may also be present in successful operations. It may be an empty string if there is no message.
Referral	A set of LDAP URLs to other servers in which the client may attempt the operation. This element may be absent if there are no referrals.

LDAPS

LDAPS is a term that is used to refer to LDAP communication over Secure Sockets Layer.

LDAP search filter

A search filter provides a mechanism for defining the criteria for defining matching entries in an LDAP search operation. There are ten different types of search filters defined in LDAP:

AND search filter	Serve as a container for holding zero or more search filter elements. All search filters contained in the AND filter must match the target entry for the AND filter to match.
OR search filter	Serve as a container for holding zero or more search filter elements. At least one of the search filters contained in the OR filter must match the target entry for the OR filter to match.
NOT filters	Serves as a container for exactly one search filter element. The embedded filter must not match the target entry for the NOT filter to match.
equality search filter	Provides a mechanism for identifying entries that contain a specified value for a given attribute.
substring search filter	Provides a mechanism for identifying entries with attribute values matching a specified substring.

greater than or equal to search filter	Provides a mechanism for identifying entries with attribute values greater than or equal to a specific value.
less than or equal to search filter	Provides a mechanism for identifying entries with attribute values less than or equal to a specific value.
presence search filter	Provides a mechanism for identifying entries that contain at least one value for a specified attribute.
approximate search filter	Provides a mechanism for identifying entries with attribute values that are approximately equal to a given value.
extensible match search filter	Provides a mechanism for using a matching rule to identify matching entries using an extensible mechanism.

See RFC 4515 for more information about LDAP search filters and a mechanism for representing them as strings.

ldapsearch command

The ldapsearch command may be used to request LDAP search operation.

See "ldapsearch" in *Oracle Fusion Middleware Command-Line Usage Guide for Oracle Unified Directory* for detailed information on using this command.

LDAP true filter

An LDAP true filter is a special type of AND search filter that does not contain any embedded filter components. It is called an "LDAP true filter" because it always evaluates to `true` and will match any entry.

The string representation for an LDAP true filter is `(&)`. LDAP true filters are described in RFC 4526.

LDAP Subentry

An LDAP subentry is a type of entry that contains the `ldapSubEntry` object class. These entries are meant to hold operational data for the server. They are kind of like operational attribute in that they are not returned to clients unless explicitly requested by including a request control with an OID of 1.3.6.1.4.1.7628.5.101.1 and no value. This behavior is described in draft-ietf-ldup-subentry.

For an example of using this control in a search, see "To Search Using the LDAP Subentry Control" in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

LDAP URL

An LDAP URL is a type of URL that may be used to reference an entry or set of search criteria. The format of an LDAP URL is described in RFC 4516 and may include the following elements:

- The address of the directory server
- The port number of the directory server
- The search base DN
- A set of search attributes
- The scope for the search
- A search filter for identifying the entries to match
- A set of extensions that provide information about the way in which the search should be processed

All of these elements are optional. Technically, all that is required of an LDAP URL is the string `ldap://`. However, a more complete URL might be `ldap://directory.example.com:389/dc=example,dc=com?cn,givenName,sn?sub?(uid=john.doe)`.

LDIF export

An LDIF export operation is a process by which all or part of the content in a Directory Server back end is written to a file using the LDAP Data Interchange Format. An LDIF export may be initiated using either the export-ldif command or an LDIF export task.

LDIF import

An LDIF import operation is a process by which data can be added to a Directory Server back end from a file with information in the LDAP Data Interchange Format. An LDIF import provides a significantly more efficient means of adding a large number of entries to the server than LDAP add operations.

An LDIF import operation can be initiated using the import-ldif command or with the LDIF import task.

leaf entry

A leaf entry is an entry that does not have any subordinate entries in the server.

less than or equal to search filter

An less or equal search filter is a type of search filter that can be used to identify entries that contain a specific value for a given attribute that is less than or equal to the provided assertion value. The server will use an ordering matching rule to make the determination.

The string representation of an LDAP less or equal search filter is composed of an opening parenthesis followed by the attribute name, a less than sign, an equal sign, the assertion value, and the closing parenthesis. For example, a less or equal filter of `(createTimestamp<=20070101000000Z)` will match any entry that has a `createTimestamp` value that is less than or equal to `20070101000000Z`.

lexico algorithm

A proxy distribution algorithm, in which the data is split into partitions based on alphabetical delimitations. For example, [A-E[for one partition and [E-H[for the next partition.

Lightweight Directory Access Protocol

The Lightweight Directory Access Protocol (LDAP) is a protocol that may be used to communicate with a directory server. It is an open standard that uses the Basic Encoding Rules subset of Abstract Syntax Notation One to encode communication into message.

The core LDAPv3 specification is in RFC 4510, with RFC 4511 defining the actual encoding for the protocol. A number of other specifications are defined in a number of request for comments and Internet Draft.

LDAP defines a number of different types of operations, including:

abandon operation	Provides a way to abort the processing for an operation in progress
add operation	Provides a way to add a new entry to the server
bind operation	Provides a way to authentication to the server
compare operation	Provides a way to determine whether an entry has a specified attribute value assertion
delete operation	Provides a way to remove entries from the server

extended operation	Provides a way to perform custom processing implemented as an extension to the core LDAP protocol
modify operation	Provides a way to alter the contents of an entry in the server
modify DN operation	Provides a way to rename an entry in the server
search operation	Provides a way to identify all entries that match a given set of criteria
unbind operation	Provides a way to indicate that the client wishes to disconnect from the server

load balancing

Load balancing is a proxy deployment type which provides single access to a set of replicated remote LDAP servers. The choice of the remote LDAP server to which a client requests is sent is determined by a load balancing algorithm.

lookthrough limit

The lookthrough limit is a configuration option within the Directory Server that can be used to enforce a limit on the number of entries that the server will examine in the course of processing a search operation. This limit applies to all entries that the server examines, regardless of whether it matches the provided search criteria.

The lookthrough limit configuration attribute can be used to limit the impact of unindexed searches, or searches with a very large candidate list.

For information about configuring the lookthrough limit, see “Setting Resource Limits on a User Account” in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory* and “Setting Root User Resource Limits” in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

M

MakeLDIF command

The MakeLDIF command provides a mechanism for generating entry in LDIF form. The entries will be generated based on a template containing a number of tags that can be used to control the way that the data is generated.

See "make-ldif" in *Oracle Fusion Middleware Command-Line Usage Guide for Oracle Unified Directory* for information on using this command. "Creating MakeLDIF Template Files" in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory* describes the valid structure and content for MakeLDIF template files.

manage DSA IT control

The Manage DSA IT control is a type of control that can be used to request that the server treat smart referral as regular entries. It can be attached to a delete operation, modify operation, or modify DN operation operation to request that the server apply the operation to the entry containing the smart referral rather than sending the referral back to the client. It may also be attached to a search operation to indicate that the server should return the entries containing the smart referrals as search result entry rather than search result reference.

The Manage DSA IT control is defined in RFC 3296. It has an OID of `2.16.840.1.113730.3.4.2` with no value.

For an example of using this control in a search request, see "To Search Using the Manage DSA IT Control" in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

matched DN

A matched DN is an element of an LDAP result object that can provide additional information about the closest matching entry found in the server. It is generally used when a request targets an entry that does not exist, in which case the matched DN should contain the distinguished name of an entry that does exist in the server and is the closest ancestor of the specified entry. For example, if an operation targeted an entry `uid=doesnt.exist,ou=People,dc=example,dc=com` that did not exist but the entry `ou=People,dc=example,dc=com` does exist in the server, then that may be returned as the matched DN.

There is no guarantee that a matched DN is returned from an operation targeting an entry that does not exist, in which case the matched DN element of the LDAP result will be an empty string. This may be used, for example, if the request targeted an entry that does not have any hierarchical relationship with any other entry in the server.

matched values control

The matched values control is a type of control that can be attached to a search operation to indicate that only values matching a specified filter should be included in entries returned to the client. It is described in RFC 3876.

The request control should have an OID of 1.2.826.0.1.3344810.2.3. The value should be encoded as follows:

```
          ValuesReturnFilter ::= SEQUENCE OF SimpleFilterItem

          SimpleFilterItem ::= CHOICE {
               equalityMatch   [3] AttributeValueAssertion,
               substrings      [4] SubstringFilter,
               greaterOrEqual  [5] AttributeValueAssertion,
               lessOrEqual     [6] AttributeValueAssertion,
               present         [7] AttributeDescription,
               approxMatch     [8] AttributeValueAssertion,
               extensibleMatch [9] SimpleMatchingAssertion }

          SimpleMatchingAssertion ::= SEQUENCE {
               matchingRule    [1] MatchingRuleId OPTIONAL,
               type            [2] AttributeDescription OPTIONAL,
--- at least one of the above must be present
               matchValue      [3] AssertionValue}
```

There is no corresponding response control.

For an example of using this control in a search request, see "To Search Using the Matched Values Filter Control" in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

matching rule

A matching rule is a schema element that defines how the server should interact with values of an attribute. There are three standard types of matching rules:

- Equality matching rules are used to determine whether one attribute value is equal to another. This determination is generally made based on the normalized value, and ignores insignificant differences (for example, differences in capitalization or extra spaces).
- Ordering matching rules are used to determine the relative order between two values in a sorted list. This is used when performing server-side sort control, but it is also used for greater than or equal to search filter and less than or equal to search filter filter components.
- Substring matching rules are used to determine whether a value contains a given substring search filter.

In addition to these standard matching rules, the directory server defines a fourth type, approximate matching rules, which are used to determine whether one value is approximately equal to another. The definition of "approximately equal to" can vary, but one common use is "sounds like".

Common examples of matching rules include:

booleanMatch	An equality matching rule that determines whether two Boolean values are equal to each other.
caseExactMatch	An equality matching rule that determines whether two string values are equal to each other, without ignoring differences in capitalization.
caseExactOrderingMatch	An ordering matching rule that is used to determine the relative order between two string values, without ignoring differences in capitalization.
caseExactSubstringsMatch	A substring matching rule that is used to determine whether a string value contains a given substring, without ignoring differences in capitalization.
caseIgnoreMatch	An equality matching rule that determines whether two string values are equal to each other, ignoring differences in capitalization.
caseIgnoreOrderingMatch	An ordering matching rule that is used to determine the relative order between two string values, ignoring differences in capitalization.
caseIgnoreSubstringsMatch	A substring matching rule that is used to determine whether a string value contains a given substring, ignoring differences in capitalization.
distinguishedNameMatch	An equality matching rule that determines whether two distinguished name are equal to each other, ignoring extra spaces around commas separating RDN components and equal signs separating RDN names from values. The individual RDN values will be compared based on the matching rules associated with the corresponding RDN attributes.
generalizedTimeMatch	An equality matching rule that determines whether two generalized time values are equal to each other.
generalizedTimeOrderingMatch	An ordering matching rule that is used to determine the relative order between two generalized time values.
integerMatch	An equality matching rule that determines whether two integer values are equal to each other.
integerOrderingMatch	An ordering matching rule that is used to determine the relative order between two integer values.
octetStringMatch	An equality matching rule that determines whether two values are exactly equal to each other using a byte-for-byte

comparison.

In most cases, the directory server will use matching rules in a completely "behind the scenes" manner without the client needing to know about it. Whenever the client references a given attribute type, then the server will automatically know to use the appropriate matching rules for that attribute. However, it is also possible for the client to request that the server use a specific matching rule when performing an operation through the use of an extensible match filter.

The set of matching rules defined in the server may be determined by retrieving the `matchingRules` attribute of the subschema subentry. For more information about matching rules, see "Understanding Matching Rules" in *Oracle Fusion Middleware Architecture Reference for Oracle Unified Directory*.

matching rule use

A matching rule use is a schema element that can be used to determine which attribute type can be used in conjunction with a given matching rule. Note that this only applies when using extensible match filters.

A matching rule use definition includes an OID for the matching rule that it applies to and a list of the names or OIDs of the attribute types that may be used in conjunction with that matching rule. If an attribute is not included in this list, then it cannot be used in conjunction with the associated matching rule. If there is no matching rule use defined for a given matching rule, then it should be assumed that the matching rule can be used with any attribute type.

The set of matching rule uses defined in the server may be determined by retrieving the `matchingRuleUse` attribute of the subschema subentry. For more information about matching rule uses, see "Understanding Matching Rule Uses" in *Oracle Fusion Middleware Architecture Reference for Oracle Unified Directory* document.

MD5

MD5 is a one-way message digest algorithm defined in RFC 1321. It can be used to encode a value of an arbitrary length into a 128-bit value that cannot be reversed to determine the original cleartext. It is commonly used as a mechanism for checksumming data, and it is also commonly used for encoding passwords and other sensitive information.

Note that recent advances in cryptography have discovered weaknesses in the MD5 algorithm. These discoveries do not directly impact the security of the way that the MD5 algorithm is used by the directory server, but nevertheless it may be wise to use a stronger mechanism like the Secure Hash Algorithm.

message

See LDAP message.

message ID

The message ID is an integer value that is contained in the message and is used to correlate request and response messages. The client chooses a message ID value to include in the request message, and the server will use the same message ID in all response messages. This makes it possible for the client to have multiple requests in progress on the same connection at any given time. All requests in progress at any given time must have different message IDs. The client will typically keep a sequentially-increasing counter for all request messages so that each request gets a different message ID than the last.

Note that unsolicited notification messages will always have a message ID value of zero. All other LDAP messages should have a message ID value between 1 and 2147483647.

modification

A modification is an element of an LDAP modify operation that describes a change to a single attribute. A modify request may include one or more modifications to the target entry.

A modification consists of a modification type that describes the type of change (add, delete, replace, or increment), and the attribute including the attribute description and zero or more attribute values.

modification type

A modification type describes one of the four ways in which an attribute can have its attribute value altered in a modification. The defined modification types are:

add	One or more values are to be added to the target attribute. If the attribute does not exist in the target entry, then it will be added with the given values; otherwise the provided values will be appended to the set of values already defined for that attribute. An add modification type must always supply at least one value.
delete	One or more values are to be removed from the target attribute, or that attribute is to be removed entirely from the target entry. If one or more specific values are given, then only those values are to be removed from the target attribute (and if they represent the entire set of values for that attribute, then that attribute will be removed from the entry). If no values are given, then the entire attribute (regardless of the number of values it contains) is to be removed from the entry.

replace	The set of values for the target attribute should be replaced with the given set of values. A replace can have zero or more values, and the behavior is as follows: ■ If the target attribute already exists in the entry with one or more values, and the replace modification does not have any of its own values, then the target attribute will be removed from the entry. ■ If the target attribute already exists in the entry with one or more values, and the replace modification has one or more of its own values, then the existing set of values will be replaced with the new set of values. ■ If the target attribute does not exist in the entry and the replace modification does not have any of its own values, then no action will be taken. ■ If the target attribute does not exist in the entry and the replace modification has one or more of its own values, then the attribute will be created in the entry with the specified set of values.
increment	The value of the target attribute should be incremented by the specified amount. The target attribute must exist in the entry with exactly one value, and that value must be an integer. The increment modification must also include exactly one value and that value must be an integer. The existing value is to be incremented by an amount specified by the increment value. If the increment value is negative, then the existing value will be deprecated by an amount equal to the absolute value of the increment value.

modify DN operation

See LDAP modify DN operation.

modify operation

See LDAP modify operation.

monitor entry

A monitor entry is a type of entry in the server that provides information about a server component. It may provide statistical information for performance monitoring, information about the health of the server, or other information that could be of value.

The directory server provides a general-purpose monitor entry with a distinguished name of `cn=monitor`. A number of other monitor entries exist below that point, including:

- Information about each back end configured in the server
- Information about each connection handler configured in the server
- General information about the system on which the server is running
- Information about the state of the server work queue
- Version information for the server
- A stack trace of all threads currently active in the server

N

name form

A name form is a schema element that may be used to control which attribute type may be used in the RDN for an entry based on its structural object class.

A name form definition include these components:

- An OID used to uniquely identify the name form.
- A set of zero or more names that can be used to more easily reference the name form.
- The name or OID of the structural object class with which the name form is associated. Any entry with that structural class will be required to have an RDN which conforms to the requirements of the name form.
- An set of one or more attribute type names or OIDs for attributes that must be present in the RDN of entries with the associated structural class.
- An optional set of one or more attribute type names or OIDs for attributes that may optionally be present in the RDN of entries with the associated structural class.

The set of name forms defined in the server may be determined by retrieving the `nameForms` attribute of the subschema subentry. For more information about name forms, see the "Understanding Name Forms" in *Oracle Fusion Middleware Architecture Reference for Oracle Unified Directory*.

naming context

A naming context, also called a suffix, is a top-level entry in the server's directory information tree. It is an entry that does not have a parent.

The set of naming contexts defined in the server is listed in the `namingContexts` attribute of the root DSE. Naming contexts are visible through workflows.

network group

A network group contains a set of criteria that define categories of client connection. If the client request that is sent to the server meets the policies that are attached to the network group, the network group forwards the request to a *workflow*.

non-leaf entry

A non-leaf entry is an entry that has at least one subordinate entry in the server.

normalized value

A normalized value is a value that has been processed in a way that makes it possible to be efficiently compared against other values. The normalization process is performed using matching rules and varies based on the type of matching rule. Some kinds of transformations that may be made include:

- Converting all characters to lowercase (or uppercase) to eliminate insignificant differences in capitalization
- Eliminating unnecessary spaces in the value
- Converting values which may have multiple representations into a common form

notice of disconnection unsolicited notification

The notice of disconnection is a type of unsolicited notification that can be used to indicate that the server is about to close the connection to the client for some reason (for example, the server is being shut down, or the client has remained idle for too long).

The OID for the extended response containing the notice of disconnection is `1.3.6.1.4.1.1466.20036`. It will not have a response value, but the result code may provide an indication of the reason for the disconnection, and the diagnostic message may provide a human-readable explanation.

NOT search filter

A NOT search filter is a type of search filter that is intended to serve as a container that holds exactly one embedded search filter. The NOT filter is essentially an inverse operation, and in order for an entry to match a NOT filter, it must not match the embedded filter.

NOT filters may be represented as a string by enclosing the entire filter in parentheses and placing an exclamation point just after the opening parentheses. For example, a filter of `(!(objectClass=person))` will only match an entry if it does not have an object class value of `person`.

numeric algorithm

A proxy distribution algorithm in which data is split into partitions based on numerical delimitations. For example, [1–1000[for one partition, and [1000–2000[for the next partition.

O

object class

An object class is a schema element that correlates an OID and a set of names with a set of required and optional attribute type.

The components of an object class definition include:

- An OID used to uniquely identify the object class.
- A set of zero or more names that can be used to more easily reference the object class.
- An optional superior class, which may define additional required and/or optional attribute types.
- An optional object class type value that indicate whether the object class is structural, auxiliary, or abstract.
- An optional set of one or more attribute type names or OIDs for attributes that must be present in entries containing the object class.
- An optional set of one or more attribute type names or OIDs for attributes that may optionally be present in entries containing the object class.

Every entry must have exactly one structural object class, and it may have zero or more auxiliary classes. The complete set of object classes in an entry define the set of attribute types that are required or allowed to be present. The structural class may also be used to link the entry with a name form, DIT content rule, and/or DIT structure rule.

The set of object classes defined in the server may be determined by retrieving the `objectClasses` attribute of the subschema subentry. For more information about object classes, see the "Understanding Object Classes" in *Oracle Fusion Middleware Architecture Reference for Oracle Unified Directory* document.

object class type

An object class type is used to define the category for an object class. There are three object class type values:

structural object class	A structural object class is used to define the primary type for an entry. Each entry must have exactly one structural class, and it defines the core type of object that the entry represents.
auxiliary	An auxiliary object class is used to define a characteristic of an entry. An entry may have zero or more auxiliary classes. The set of auxiliary classes that an entry may have may be controlled by a DIT content rule that is associated with the entry's structural class.
abstract	An abstract object class is not intended to be used directly in entries but should be subclassed by a structural or auxiliary class.

The inheritance model used for LDAP object classes is very similar to the inheritance model for Java classes. Just like an entry must only exactly one structural object class, a Java class must have exactly one superclass. Similarly, while an entry may have multiple auxiliary classes, a Java class may implement multiple interfaces. Finally, it is not possible to instantiate an abstract Java class, just as it is not possible to create an entry containing only an abstract object class.

object identifier

An object identifier (OID) is a string that comprises a series of integers separated by periods. It is used as a unique identifier for various types of elements in the Directory Server, including:

- attribute syntax
- matching rule
- attribute type
- object classes
- name forms
- controls
- extended operations
- supported features

operation ID

An operation ID is an integer identifier that is assigned to each operation performed on a client connection. It is used primarily for logging purposes, so that it is possible to correlate a response log message with the corresponding request message.

The first operation performed on a client connection is assigned an operation ID of zero, and it is incremented by one for each additional request received on that client connection.

operational attribute

A user attribute is an attribute type with an attribute usage of `directoryOperation`, `distributedOperation`, or `dSAOperation`. Operational attributes are used for storing information needed for processing by the server itself or for holding any other data maintained by the server that was not explicitly provided by clients.

Operational attributes are not included in entries returned from search operations unless they are explicitly included in the list of search attributes. An explicit value of + (the plus sign) may also be included to request that all operational attributes be returned.

ordering index

An ordering index is a type of index that is used to keep track of the relative order of values for an attribute. It is very similar to an equality index except that it uses an ordering matching rule instead of an equality matching rule to normalized value the values. Ordering indexes may not be maintained for attributes that do not have a corresponding ordering matching rule.

OR search filter

An OR search filter is a type of search filter that is intended to serve as a container that holds zero or more other search filters. In order for an entry to match an OR filter, it must match at least one of the filters contained in that OR filter.

OR filters may be represented as a string by enclosing the entire filter in parentheses and placing a pipe symbol (|) just after the opening parenthesis. For example, a filter of `(|(uid=john.doe)(uid=jane.doe))` represents an OR search filter that embeds the `(uid=john.doe)` and `(uid=jane.doe)` equality filters.

An OR filter that does not contain any embedded filters is called an LDAP false filter. The string representation for an LDAP false filter is `(|)`, and LDAP false filters will never match any target entry.

P

partition

In a proxy distribution deployment, the data is split into smaller chunks of data, each of which is known as a partition. A partition of data is typically stored on a separate remote LDAP server, or on a set of replicated remote LDAP servers to ensure high availability.

password

A password is a secret value that may be used to provide proof of identity in some authentication mechanisms. In particular, a password is used in simple authentication, as well as the CRAM-MD5, DIGEST-MD5 , and PLAIN SASL mechanisms.

The security that a password provides is based entirely on the fact that only the password's owner knows what the password is. If someone else learns a user's password through some means, then that third party can impersonate that user and may be able to perform any operation available to that user.

The Directory Server provides a number of password policy features that can be used to help ensure that passwords are not discovered by third-party individuals (for example, helping to ensure that users aren't allowed to use weak passwords, providing protection against brute-force attacks, requiring authentication attempts and password changes from being performed in a secure manner), but nevertheless passwords are often considered weaker forms of protection than other kinds of identification like certificate.

password expiration

Password expiration is an element of the Directory Server password policy that can be used to limit the length of time that a user can continue to use the same password. If password expiration is enabled, once a user changes his or her password, they can use it for a length of time specified as the maximum password age. As the password expiration time draws near, the user may receive warning messages in the form of control in the bind response. Once the password has expired, the user will no longer be allowed to authentication.

Once the user's password has expired, it may be necessary for an administrator to password reset before the account may be used. Alternately, if the password policy is configured appropriately, the user may also be able to change their own expired password using the Password Modify extended operation.

password generator

A password generator is a piece of logic that may be used to generate a password for a user as part of a Password Modify extended operation. It will be used if the password modify request does not include a new password.

Password Modify extended operation

The Password Modify extended operation is a type of extended operation that may be used to change or reset user passwords. It is defined in RFC 3062 and both the request and response operations have an OID of 1.3.6.1.4.1.4203.1.11.1.

The value for the password modify request is:

```
PasswdModifyRequestValue ::= SEQUENCE {
     userIdentity    [0]  OCTET STRING OPTIONAL
     oldPasswd       [1]  OCTET STRING OPTIONAL
     newPasswd       [2]  OCTET STRING OPTIONAL }
```

The value for the password modify response is:

```
PasswdModifyResponseValue ::= SEQUENCE {
     genPasswd       [0]     OCTET STRING OPTIONAL }
```

password policy

The Directory Server password policy provides a mechanism for controlling how passwords will be stored and maintained in the server, and how users will be allowed to authenticate.

Elements of the password policy include:

- The attribute used to store user passwords. By default, this is the `userPassword` attribute.
- The default set of password storage schemes that will be used to encode passwords stored in the server.
- A set of deprecated password storage schemes that can be used to authenticate users but cause the password to be re-encoded using the default schemes upon a successful bind.
- A flag that indicates whether users will be allowed to change their own passwords.
- A number of settings related to password expiration, including the maximum age for passwords, warnings before expiration, and whether users will be allowed to change their passwords after they expire.
- A number of settings related to account lockout, which can be used to prevent users from authenticating after too many failed attempts.

- Flags that indicate whether users will be required to change their passwords the first time they authenticate and/or whether they will be required to change their passwords after they have been reset by an administrator.
- A set of password validators that can be used to determine whether proposed new password values are acceptable for use.
- A flag that indicates whether users will be required to provide their current passwords to be allowed to change their passwords.
- A flag that indicates whether clients will be allowed to specify new passwords that have already been encoded using one of the password storage schemes defined in the server. Allowing pre-encoded passwords may be necessary for some applications, but may allow the user to bypass certain restrictions, like password validators, that might otherwise be enforced.
- Settings related to maintaining the last login time, including the attribute to use to store its value, the format to use for the time stamp, and whether to lock an account after too much time has elapsed without authenticating.
- Flags that control whether the user will be required to authenticate in a secure manner and/or whether they will be required to change their passwords in a secure manner.

password policy control

The password policy request control is a type of LDAP control that can be used to request information about the current password policy state for a user entry. It is defined in draft-sisbehera-ldap-password-policy. Both the request and response controls have an OID of 1.3.6.1.4.1.42.2.27.8.5.1. The request control does not have a value. The response control value is encoded as follows:

```
PasswordPolicyResponseValue ::= SEQUENCE {
     warning [0] CHOICE {
          timeBeforeExpiration [0] INTEGER (0 .. maxInt),
          graceAuthNsRemaining [1] INTEGER (0 .. maxInt) } OPTIONAL,
     error   [1] ENUMERATED {
          passwordExpired           (0),
          accountLocked             (1),
          changeAfterReset          (2),
          passwordModNotAllowed     (3),
          mustSupplyOldPassword     (4),
          insufficientPasswordQuality (5),
          passwordTooShort          (6),
          passwordTooYoung          (7),
          passwordInHistory         (8) } OPTIONAL }
```

For an example of using this control in a search request, see “To Search Using the Password Policy Control” in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

password reset

A password reset is the act of a server administrator changing a user's password. A password reset is a password change that is performed by any user other than the one that owns the account.

password storage scheme

A password storage scheme provides a mechanism for encoding user passwords for storage in the server. In most cases, the password is encoded in a manner that prevents users from determining what the clear-text password is, while still allowing the server to determine whether the user-supplied password is correct. Password storage schemes currently available for use include:

3DES	The password will be encoded using triple DES. Triple DES is a variation of the Data Encryption Standard (DES) that is three times slower than its predecessor but provides stronger reliability. The algorithm uses three 64-bit keys for a combined key length of 192 bits. The data is encrypted with the first key, decrypted with the second key, and then re-encrypted with the third key. You must ensure that all three keys, the first and the second key, or the second and the third keys are not identical.
AES	The Advanced Encryption Standard uses a symmetric block cipher that processes data blocks of 128 bits, using cipher keys with lengths of 128 (AES-128), 192 (AES-192), and 256 (AES-256) bits and is based on the Rijndael algorithm
BASE64	The password will be base64–encoded, which provides a very weak form of protection and should only be used for cases in which clients require this storage scheme.
BlowFish	The password will be encoded using the BlowFish Algorithm with a 128 bits key length.
CLEAR	The password will be stored in clear-text. It will not provide any protection at all, so this should only be used for cases in which clients require this storage scheme.
CRYPT	The password will be encoded using the UNIX crypt algorithm. This is a one-way algorithm, but it is considered weak by current standards and should generally only be used for clients which require this storage scheme.
MD5	The password will be encoded using an unsalted version of the MD5 message digest algorithm. This is relatively secure, although a salted hash is preferred, and one of the SHA variants are considered stronger than MD5.

RC4	The password will be encoded using RC4, a stream cipher using a variable key-size stream cipher with byte-oriented operations. The algorithm is based on the use of a random permutation.
SMD5	The password will be encoded using a salted version of the MD5 message digest algorithm.
SHA	The password will be encoded using an unsalted version of the SHA-1 Secure Hash Algorithm. The salted variant of this algorithm is preferred.
SSHA	The password will be encoded using a salted version of the SHA-1 Secure Hash Algorithm. This is the default password storage scheme used by the directory server
SSHA256	The password will be encoded using a salted 256-bit version of the SHA-2 Secure Hash Algorithm.
SSHA384	The password will be encoded using a salted 384-bit version of the SHA-2 Secure Hash Algorithm.
SSHA512	The password will be encoded using a salted 512-bit version of the SHA-2 Secure Hash Algorithm.

Note that the directory server also supports the use of the authentication password syntax.

password validator

A password validator is a component of the directory server password policy that is used to determine whether a proposed password is acceptable for use. The directory server provides an extensible API for developing custom password validators, but it does come with a number of different types of password validators, including:

- A validator that can be used to reject a password if the value exists in any of the attributes contained in the user's entry.
- A validator that can be used to reject a password if the value does not contain characters from an acceptable range of character sets.
- A validator that can be used to reject a password if it is a word that can be found in a dictionary.
- A validator that can be used to reject a password if it is too long or too short.
- A validator that can be used to reject a password if it contains a string of too many repeated characters.
- A validator that can be used to reject a password if it is too similar to the user's current password.

- A validator that can be used to reject a password if it does not contain enough unique characters.

persistent search control

The persistent search control is a type of LDAP control that may be used for clients to be notified of changes to entries that match the criteria from the associated LDAP search operation. The persistent search control is described in draft-ietf-ldapext-psearchand has an OID of 2.16.840.1.113730.3.4.3. It is defined as follows:

```
PersistentSearch ::= SEQUENCE {
     changeTypes INTEGER,
     changesOnly BOOLEAN,
     returnECs BOOLEAN
}
```

Search result entries returned as part of this search may optionally include the entry change notification control to describe the way in which the entry changed. For an example of using this control in a search, see “To Search Using the Persistent Search Control” in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

PLAIN SASL mechanism

The PLAIN Simple Authentication and Security Layer mechanism provides a way for clients to authentication to the Directory Server with a username and password. In general, it is very similar to simple authentication, with the exception that the client can identify itself with a username rather than a distinguished name. It also provides the ability for the client to specify an alternate authorization ID.

Like simple authentication, the PLAIN SASL mechanism does not provide any form of protection for the user password, so it may be advisable to only use this authentication method over secure communication channels like those provided by Secure Sockets Layer or StartTLS.

plug-in

A plug-in is a piece of code that can be used to interject some custom logic into the way that the Directory Server performs its processing. The directory server supports a number of different types of plug-ins, including:

- Pre-parse plug-ins, which allow the server to alter the contents of a request before the server begins processing on it. Pre-parse plug-ins are available for all types of operations.
- Pre-operation plug-ins, which allow the server to take some action just before the core processing for an operation. Pre-operation plug-ins are available for all types of operations except abandon and unbind..
- Post-operation plug-ins, which allow the server to take some action just after the core processing for an operation but before the response has been sent to the client (it may be used to alter the response to the client). Post-operation plug-ins are available for all types of operations.
- Post-response plug-ins, which allow the server to take some action after all other processing for an operation has completed. Post-response plug-ins are available for all types of operations except abandon and unbind.
- Search result entry plug-ins, which alter the contents of a search result entry being sent as part of a search operation.
- Search result reference plug-ins, which alter the contents of a search result reference being sent as part of a search operation.
- Intermediate response plug-ins, which alter the contents of an intermediate response being sent to a client.
- Startup plug-ins, which perform some processing when the server is first starting.
- Shutdown plug-ins, which perform some processing when the server is performing a graceful shutdown.
- Post-connect plug-ins, which perform some processing as part of accepting a new client connection.
- Post-disconnect plug-ins, which perform some processing immediately after a connection is terminated.
- LDIF import plug-ins, which alter the contents of entries being imported from an LDIF file.
- LDIF export plug-ins, which alter the contents of entries being exported from a server back end.

presence index

A presence index is a type of index that is used to keep track of the entries that have at least one value for a specified attribute. There is only a single presence index key per attribute, and its ID list contains the entry IDs for all entries that contain the specified attribute.

presence search filter

A presence search filter is a type of search filter that can be used to identify entries that have at least one value for a specified attribute. The string representation of an LDAP presence filter comprises an opening parenthesis followed by the attribute name, an equal sign, an asterisk, and the closing parenthesis. For example, an equality filter of `(aci=*)` will match any entry containing at least one value for the `aci` attribute.

privilege

The directory server provides a privilege subsystem, which can be used to define capabilities that will be granted to users. The privilege subsystem works in conjunction with the access control implementation in the process of determining whether a user will be allowed to perform a certain operation.

Some of the privileges defined in the directory server include:

`bypass-acl`	Allows the user to bypass access control evaluation
`modify-acl`	Allows the user to modify access control rule defined in the server.
`config-read`	Allows the user to have read access to the server configuration
`config-write`	Allows the user to have write access to the server configuration
`server-shutdown`	Allows the user to request that the server shut down
`server-restart`	Allows the user to request that the server perform an in-core restart
`proxied-auth`	Allows the user to request an operation with a different authorization ID
`unindexed-search`	Allows the user to request an unindexed search
`password-reset`	Allows the user to password reset for other users
`update-schema`	Allows the user to update the server schema

See Chapter 6, “Directory Server Root Users and the Privilege Subsystem,” in *Oracle Fusion Middleware Architecture Reference for Oracle Unified Directory* for more information on the privilege subsystem.

proportional algorithm

A proxy load balancing algorithm in which client requests are distributed to a set of replicated remote LDAP servers. How many requests are sent to each remote LDAP server is determined by the weight set.

protocol data unit

A protocol data unit (PDU) is a single complete element of network communication. For LDAP, the PDU is the message.

protocol op

The protocol op is the element in the message that contains the heart of the request or response. That is, it indicates what type of message it is. There are several different kinds of protocol op elements, including:

- The abandon request protocol op
- The add request and add response protocol ops
- Thebind request and bind response protocol ops
- The compare request and compare response protocol ops
- The delete request and delete response protocol ops
- The extended request and extended response protocol ops
- The modify request and modify response protocol ops
- The modify DN request and modify DN response protocol ops
- The search request, search result entry, search result reference, and search result done protocol ops
- The unbind request protocol op
- The intermediate response protocol op

proxied authorization control

The proxied authorization control is a type of control that can be used to request that the associated operation be performed under the authorization of another user.

There are actually two different forms of the proxied authorization control, both of which are request controls that may be attached to an add operation, compare operation, delete operation, modify operation, modify DN operation, or search operation operation.

The proxied authorization v1 control is defined in early versions of draft-weltman-ldapv3-proxy. It has an OID of 2.16.840.1.113730.3.4.12 and the control value should be encoded as:

```
proxyAuthValue::= SEQUENCE {
      proxyDN LDAPDN
}
```

The proxied authorization v2 control is defined in RFC 4370. It has an OID of 2.16.840.1.113730.3.4.18 and the value is a string containing the desired authorization ID.

For an example of using this control in a search request, see "To Search Using the Proxied Authorization Control" in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

Q

quality of protection

Quality of protection (QoP) is a property of certain SASL mechanisms (especially the DIGEST-MD5 and GSSAPI mechanisms) that can be used to protect the communication between the client and the server.

There are three different QoP levels:

`auth`	This indicates that the associated SASL mechanism should only be used to authenticate the client connection. It should not provide any other protection for the client-server communication
`auth-int`	This indicates that the associated SASL mechanism should be used for authentication, and then should also provide integrity protection for the communication between the client and server. Integrity protection will not prevent third-party observers from understanding the communication, but it will ensure that a man-in-the-middle is unable to alter that communication in an undetectable manner
`auth-conf`	This indicates that the associated SASL mechanism should be used for authentication, and then should also provide integrity and confidentiality protection for the communication between the client and the server. This will ensure that third-party observers will be unable to understand the communication

At the present time, the directory server supports only the `auth` quality of protection. It does not support either the `auth-int` or `auth-conf` levels.

R

real attributes only control

The real attributes only control is a control that may be used to request that the server only include real attributes in matching entries. That is, virtual attributes are excluded from search result entry.

The real attributes only control has a request OID of `2.16.840.1.113730.3.4.17` and no value.

In the following search, the `numsubordinates` virtual attribute is requested and returned:

```
$ ldapsearch -D "cn=directory manager" -w password -b "ou=people,dc=example,dc=com" \
  -s base "objectclass=*" numsubordinates
version: 1
dn: ou=People,dc=example,dc=com
numSubordinates: 50
```

In the following search, the `numsubordinates` virtual attribute is requested but is not returned because the real attributes only control is used:

```
$ ldapsearch -D "cn=directory manager" -w password -J "2.16.840.1.113730.3.4.17" \
  -b "ou=people,dc=example,dc=com" -s base "objectclass=*" numsubordinates
version: 1
dn: ou=People,dc=example,dc=com
```

referential integrity

Referential integrity is a mechanism for ensuring that any references to an entry are updated whenever that entry is removed or altered. Historically, referential integrity is primarily used to ensure that attributes with a distinguished name syntax (especially group membership attributes like `member` and `uniqueMember`) are properly maintained in the event of delete operation and modify DN operation operations. For a delete operation, any references to the target entry will be removed. For modify DN operations, any references to the target entry will be renamed accordingly.

The directory server provides a configurable referential integrity plug-in that you can install using the `dsconfig` command.

referral

A referral provides a reference to an alternate location in which an operation may be processed. A referral may be included in an LDAP result object with a result code of `10` and an appropriate set of LDAP URLs. It may also be returned to clients in a search result reference.

relative distinguished name

A relative distinguished name, or RDN, is a single component within a distinguished name. It comprises one or more name-value pairs, in which the name and the value are separated by an equal sign (for example, for an RDN of `uid=ann`, the name is `uid` and the value is `ann`), and if there are multiple name-value pairs then they should be separated by plus signs (for example, for an RDN of `cn=Jon Doe+employeeNumber=12345`, the name-value pairs are `cn=John Doe` and `employeeNumber=12345`). In practice, RDNs containing multiple name-value pairs (called “multivalued RDNs”) are rare, but they can be useful at times when either there is no unique attribute in the entry or you want to ensure that the entry's DN contains some useful identifying information.

Even though a DN may be composed of multiple RDN components, the leftmost component is typically referred to as the entry's RDN. For example, in a DN of `uid=john.doe,ou=People,dc=example,dc=com`, the RDN would be `uid=john.doe`. The attribute values specified in an entry's RDN must be contained in that entry, so the entry `uid=john.doe,ou=People,dc=example,dc=com` must have a `uid` value of `john.doe`.

replica

A replica is a Directory Server instance that participates in replication.

replication

Replication is a form of data synchronization that is used to ensure that changes in the directory environment are reflected in each instance of the server. That is, whenever a change is made in one Directory Server instance, that same change is also made in every other instance.

replication repair control

The replication repair control is a control that can be used to resolve replication inconsistencies on a single server in a topology.

The replication repair control has a request OID of `1.3.6.1.4.1.26027.1.5.2` and no value.

For an example of using the replication repair control, see "Detecting and Resolving Replication Inconsistencies" in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

request for comments

A request for comments (RFC) is an IETF specification that has been promoted from an Internet Draft and may be considered significantly more stable than drafts.

restore

A restore operation provides a mechanism for replacing the contents of a Directory Server back end with information taken from a previous backup. It can serve as a disaster recovery mechanism, and in some cases can be used for binary copy initialization of a replica.

result

See LDAP result.

result code

A result code is an integer value that provides general information about the result of the operation. Defined result codes include:

Value	Name	Description
0	Success	This is used to indicate that the associated operation completed successfully.
1	Operations Error	This is used to indicate that the associated request was out of sequence with another operation in progress (for example, a non-bind request in the middle of a multi-stage SASL bind).
2	Protocol Error	This is used to indicate that the client sent data to the server that did not comprise a valid LDAP request.

Value	Name	Description
3	Time Limit Exceeded	This is used to indicate that processing on the associated request was terminated because it took too long to complete. For a search operation, it is possible that some of the matching entries had been returned when the time limit was reached.
4	Size Limit Exceeded	This is used to indicate that there were more entries matching the criteria contained in a search operation than were allowed to be returned by the size limit configuration.
5	Compare False	This is used to indicate that a compare operation completed successfully, but the provided attribute value assertion did not match the target entry.
6	Compare True	This is used to indicate that a compare operation completed successfully, and the provided attribute value assertion matched the target entry.
7	Auth Method Not Supported	This is used to indicate that the Directory Server does not support the requested authentication method.
8	Strong Auth Required	This is used to indicate that the Directory Server requires that the client use a strong authentication mechanism.
10	Referral	This is used to indicate that the requested operation could not be processed in the target server but may be attempted in elsewhere.

Value	Name	Description
11	Admin Limit Exceeded	This is used to indicate that processing on the requested operation could not be completed because an administrative limit was reached. For a search operation, it is possible that some of the matching entries had been returned when the administrative limit was reached.
12	Unavailable Critical Extension	This is used to indicate that the request included a critical control that could not be processed by the server.
13	Confidentiality Required	This is used to indicate that the requested operation requires a secure communication channel between the client and the server.
14	SASL Bind In Progress	This is used to indicate that a SASL bind operation requires multiple stages and the response containing this result code is one of the intermediate stages.
16	No Such Attribute	This is used to indicate that the associated request targeted an attribute or attribute value that does not exist in the specified entry.
17	Undefined Attribute Type	This is used to indicate that the associated request included an attribute type that is not defined in the server schema.
18	Inappropriate Matching	This is used to indicate that the associated search request included a filter with a component targeting an attribute type for which no appropriate matching rule is defined.

Value	Name	Description
19	Constraint Violation	This is used to indicate that the requested operation could not be completed because it would have violated some constraint defined in the server (for example, it would have duplicated a value for a unique attribute).
20	Attribute or Value Exists	This is used to indicate that an operation attempted to create an attribute value in an entry that already existed in the entry, or that it attempted to create an additional value for a single-valued attribute.
21	Invalid Attribute Syntax	This is used to indicate that requested operation attempted to specify a value that violated the syntax for the associated attribute type.
32	No Such Object	This is used to indicate that the requested operation targeted an entry that does not exist in the server.
33	Alias Problem	This is used to indicate that an operation targeted an alias entry and that operation is not allowed on alias entries.
34	Invalid DN Syntax	This is used to indicate that the requested operation included an entry DN that was malformed.
35	Is Leaf	This is used to indicate that the requested operation targeted a leaf entry but the operation requires a non-leaf entry.
36	Alias Dereferencing Problem	This is used to indicate that the associated search operation encountered an alias that could not be properly dereferenced.

Value	Name	Description
48	Inappropriate Authentication	This is used to indicate that the client attempted to bind in a manner that is inappropriate for the target user (for example, the user attempted simple authentication but does not have a password).
49	Invalid Credentials	This is used to indicate that the client attempted to authenticate with invalid credentials (for example, the target DN or password was incorrect).
50	Insufficient Access Rights	This is used to indicate that the client was not allowed to perform the requested operation.
51	Busy	This is used to indicate that the server is too busy to process the requested operation.
52	Unavailable	This is used to indicate that the server is unavailable for processing operations.
53	Unwilling to Perform	This is used to indicate that the server is unwilling to perform the requested operation for some reason.
54	Loop Detect	This is used to indicate that the server encountered a loop of some type (for example, a chaining loop or an alias loop).
60	Sort Control Missing	This is used to indicate that the client requested a search operation containing the virtual list view control that did not also include the server-side sort control.
61	Offset Range Error	This is used to indicate that the request included a virtual list view control that specified an invalid offset (for example, one that was beyond the end of the result set).

Value	Name	Description
64	Naming Violation	This is used to indicate that the operation attempted to create an entry with a DN that violated a naming constraint (for example, using an RDN attribute that is not allowed by the associated name form).
65	Object Class Violation	This is used to indicate that the operation attempted to create or modify an entry so that the set of attributes it contained were in violation of the associated object class definitions (for example, it included an attribute that was not allowed or was missing a required attribute).
66	Not Allowed On Nonleaf	This is used to indicate that the associated operation was not allowed on non-leaf entries (for example, an attempt to delete an entry that has one or more subordinate entries).
67	Not Allowed On RDN	This is used to indicate that the associated operation is not allowed on the RDN attribute for an entry.
68	Entry Already Exists	This is used to indicate that the add or modify DN operation would have resulted in an entry with a DN that already exists in the server.
69	Object Class Mods Prohibited	This is used to indicate that the requested operation attempted to alter the structural object class for the entry in a manner that was not allowed.
71	Affects Multiple DSAs	This is used to indicate that the requested operation would have impacted multiple servers (for example, a modify DN operation would have moved an entry from one server to another through a chained back end).

Value	Name	Description
76	Virtual List View Error	This is used to indicate that the associated search operation could not be completed successfully because a problem occurred while processing the virtual list view request.
80	Other	This indicates that the operation failed for some reason that is not more appropriately classified by any other defined result code.
81	Server Down	This is a client-side result code that is used to indicate that the client detected that an established connection was no longer available.
82	Local Error	This is a client-side result code that is used to indicate that some client-side problem occurred that prevented it from completing the associated processing successfully.
83	Encoding Error	This is a client-side result code that is used to indicate that an error occurred while attempting to encode the request to send to the server.
84	Decoding Error	This is a client-side result code that is used to indicate that an error occurred while attempting to decode the response received from the server.
85	Timeout	This is a client-side result code that is used to indicate that the client did not receive a response in an acceptable length of time.
86	Authentication Type Unknown	This is a client-side result code that is used to indicate that the client does not support the requested authentication method.
87	Filter Error	This is a client-side result code that is used to indicate that a provided filter string could not be parsed as a valid filter.

Value	Name	Description
88	User Canceled	This is a client-side result code that is used to indicate that the client canceled the request.
89	Parameter Error	This is a client-side result code that is used to indicate that there was a problem with a parameter provided for a request element.
90	No Memory	This is a client-side result code that is used to indicate that the client ran out of memory while attempting to process the requested operation (for example, while queueing the search result entries).
91	Connect Error	This is a client-side result code that is used to indicate that the client could not establish a connection to the target server.
92	Not Supported	This is a client-side result code that is used to indicate that the requested operation is not supported by the client.
93	Control Not Found	This is a client-side result code that is used to indicate that a response did not include an expected control.
94	No Results Returned	This is a client-side result code that is used to indicate that the server did not return any results for a search request when at least one was expected.
95	More Results to Return	This is a client-side result code that is used to indicate that there are more results to return than those that have already been retrieved.
96	Client Loop	This is a client-side result code that is used to indicate that the client detected a referral loop.
97	Referral Limit Exceeded	This is a client-side result code that is used to indicate that the client received too many referrals in the course of processing a request.

Value	Name	Description
100	Invalid Response	This is a client-side result code that is used to indicate that the result received for the associated operation is invalid.
101	Ambiguous Response	This is a client-side result code that is used to indicate that the result received from the server was ambiguous (for example, there was more than one response received fro the associated operation).
112	TLS Not Supported	This is used to indicate that the server does not support the StartTLS extended operation.
113	Intermediate Response	This result code is used for intermediate response messages sent by the server in the course of processing the request.
114	Unknown Type	This is used to indicate that the server received a request with an invalid or unknown protocol op type.
118	Canceled	This is used to indicate that the server canceled processing on the request at the request of the client.
119	No Such Operation	This is used to indicate that the client attempted to cancel a request that was unknown to the server (for example, because it had already completed processing).
120	Too Late	This is used to indicate that the client attempted to cancel a request that had already been processed beyond a point at which it could no longer be canceled.
121	Cannot Cancel	This is used to indicate that the client attempted to cancel an operation that could not be canceled (for example, a bind, unbind, abandon, cancel, or StartTLS request).

Value	Name	Description
122	Assertion Failed	This is used to indicate that the associated operation was not processed because the request included an LDAP assertion control with an assertion filter that did not match the target entry.
123	Authorization Denied	This is used to indicate that the associated operation was not processed because the request included a proxied authorization control but the client was not allowed to use that control.

root DN

A root DN (or root user) is a type of account that exists in the Directory Server which is generally given full access to all data in the server, much like the root user in UNIX systems. Root users by default will be allowed to bypass access control evaluation, will have full access to the server configuration, and perform most other types of operations.

The directory server is different from most other servers with regard to root users in two key ways:

- The directory server can be configured with multiple root users. This is a good thing because it allows each root user to have a different set of credentials so that each administrator can have a separate root account that is independent from the others rather than a single account that is shared by all administrators.
- All of the rights given to root users are assigned through privileges. Using the privilege subsystem, it is possible to create non-root users with some or all of the capabilities normally available only to root users. It is also possible to take away privileges from root users if so desired.

For more information on root users and the privilege subsystem, see the Chapter 6, “Directory Server Root Users and the Privilege Subsystem,” in *Oracle Fusion Middleware Architecture Reference for Oracle Unified Directory* document.

root DSE

The root DSE is a special entry that provides information about the contents and capabilities of the server. The distinguished name is a zero-length string with no RDN components, also called the null DN.

The attribute contained in the root DSE include:

`namingContexts`	Lists the naming context for the server
`supportedAuthPasswordSchemes`	Lists the OID of the supported password storage schemes using the authentication password syntax
`supportedControl`	Lists the OIDs of the supported controls in the server
`supportedExtension`	Lists the OIDs of the supported extended operations in the server
`supportedFeatures`	Lists the OIDs of the supported features in the server
`supportedSASLMechanisms`	Lists the OIDs of the supported SASL mechanisms in the server
`vendorName`	Provides the name of the vendor for the server
`vendorVersion`	Provides a product version string

The following example demonstrates how to use the `ldapsearch` command to read the root DSE. In this example the file `/tmp/pwd.txt` contains the Directory Manager password. The server is listening for LDAP requests on port 1389.

```
$ ldapsearch -D "cn=Directory Manager" -j /tmp/pwd.txt -p 1389 -b "" \
  -s base "(objectclass=*)" +
dn:
supportedLDAPVersion: 2
supportedLDAPVersion: 3
supportedExtension: 1.3.6.1.4.1.4203.1.11.3
supportedExtension: 1.3.6.1.4.1.4203.1.11.1
supportedExtension: 1.3.6.1.4.1.26027.1.6.1
supportedExtension: 1.3.6.1.4.1.26027.1.6.3
supportedExtension: 1.3.6.1.4.1.26027.1.6.2
supportedExtension: 1.3.6.1.1.8
supportedExtension: 1.3.6.1.4.1.1466.20037
vendorName: Oracle Corporation
entryDN:
ds-private-naming-contexts: cn=admin data
ds-private-naming-contexts: cn=ads-truststore
ds-private-naming-contexts: cn=backups
ds-private-naming-contexts: cn=config
ds-private-naming-contexts: cn=monitor
ds-private-naming-contexts: cn=schema
ds-private-naming-contexts: cn=tasks
supportedControl: 1.2.826.0.1.3344810.2.3
supportedControl: 1.2.840.113556.1.4.319
supportedControl: 1.2.840.113556.1.4.473
supportedControl: 1.2.840.113556.1.4.805
supportedControl: 1.3.6.1.1.12
supportedControl: 1.3.6.1.1.13.1
supportedControl: 1.3.6.1.1.13.2
supportedControl: 1.3.6.1.4.1.26027.1.5.2
supportedControl: 1.3.6.1.4.1.42.2.27.8.5.1
supportedControl: 1.3.6.1.4.1.42.2.27.9.5.2
supportedControl: 1.3.6.1.4.1.42.2.27.9.5.8
```

```
supportedControl: 1.3.6.1.4.1.4203.1.10.2
supportedControl: 1.3.6.1.4.1.7628.5.101.1
supportedControl: 2.16.840.1.113730.3.4.12
supportedControl: 2.16.840.1.113730.3.4.16
supportedControl: 2.16.840.1.113730.3.4.17
supportedControl: 2.16.840.1.113730.3.4.18
supportedControl: 2.16.840.1.113730.3.4.19
supportedControl: 2.16.840.1.113730.3.4.2
supportedControl: 2.16.840.1.113730.3.4.3
supportedControl: 2.16.840.1.113730.3.4.9
supportedSASLMechanisms: PLAIN
supportedSASLMechanisms: EXTERNAL
supportedSASLMechanisms: CRAM-MD5
supportedSASLMechanisms: DIGEST-MD5
supportedFeatures: 1.3.6.1.1.14
supportedFeatures: 1.3.6.1.4.1.4203.1.5.1
supportedFeatures: 1.3.6.1.4.1.4203.1.5.2
supportedFeatures: 1.3.6.1.4.1.4203.1.5.3
subschemaSubentry: cn=schema
hasSubordinates: true
entryUUID: d41d8cd9-8f00-3204-a980-0998ecf8427e
numSubordinates: 1
namingContexts: dc=example,dc=com
vendorVersion: Oracle Unified Directory 11.1.1.5.0
supportedAuthPasswordSchemes: MD5
supportedAuthPasswordSchemes: SHA1
supportedAuthPasswordSchemes: SHA256
supportedAuthPasswordSchemes: SHA384
supportedAuthPasswordSchemes: SHA512
```

For more information on how to search the root DSE entry, see "Using Advanced Search Features" in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

route

In proxy mode, the path on which requests are sent to the remote LDAP server when using a load balancing algorithm.

S

salt

A salt is a collection of random data that may be combined with clear-text data (often a password) that can be used to change the way that it is encoded. In particular, the salt is used to introduce randomness into the encoding process to help thwart dictionary attacks. In general, the salt is appended to the clear-text password, which is the encoded using the desired message digest algorithm, and then the clear-text salt is appended to the message digest and the resulting value is base64 encoded. This makes it possible to determine what the salt was so that it can be used to determine whether a user-supplied password is correct.

The UNIX crypt algorithm uses a relatively weak 12-bit salt, which means that there are only 4096 ways of encoding any value. This is a relatively low number, and therefore it is possible to construct dictionaries of every possible encoding for a wide range of values for use in breaking user passwords. Other password storage schemes in the directory server use a 64-bit salt which provide 18446744073709551616 different ways of encoding any one value.

saturation algorithm

A proxy load balancing algorithm in which client requests are routed to a priority remote LDAP server. When the main remote LDAP server reaches its *saturation threshold*, the requests are routed to a secondary remote LDAP server.

saturation alert

The limit at which a notification is sent to the administrator to indicate that the remote LDAP server is overloaded. Usually, the saturation alert is set higher than the *saturation threshold*.

saturation threshold

The saturation threshold is the limit at which the data source is considered overloaded and can no longer handle incoming requests in an optimal way. The saturation threshold is used as part of the proxy saturation algorithm.

schema

The schema of a Directory Server defines a set of rules that govern the kinds of information that the server can hold. Directory schema includes a number of different elements, including:

Attribute syntax	Provide information about the kind of information that can be stored in an attribute.
Matching rule	Provide information about how to make comparisons against attribute values.
Matching rule uses	Indicate which attribute types may be used in conjunction with a particular matching rule.

Attribute type	Define an OID and a set of names that may be used to refer to a given attribute, and associates that attribute with a syntax and set of matching rules.
Object class	Define named collections of attributes and classify them into sets of required and optional attributes.
Name forms	Define rules for the set of attributes that should be included in the RDN for an entry.
DIT content rules	Define additional constraints about the object classes and attributes that may be used in conjunction with an entry.
DIT structure rules	Define rules that govern the kinds of subordinate entries that a given entry may have.

Attributes are the elements responsible for storing information in a directory, and the schema defines the rules for which attributes may be used in an entry, the kinds of values that those attributes may have, and how clients may interact with those values.

Clients may learn about the schema elements that the server supports by retrieving an appropriate subschema subentry.

schema checking

Schema checking is the process of ensuring that an entry conforms to the constraints defined by the server schema. This includes:

- Make sure the entry contains exactly one structural object class.
- If there is a name form for the entry's structural class, ensure that the RDN attributes conform with that name form.
- If there is a DIT content rule for the entry's structural class, make sure that all of the auxiliary classes are defined.
- Make sure that all of the object classes contained in the entry are defined in the schema.
- Make sure that all of the attribute contained in the entry are defined in the schema and allowed by the object classes and/or DIT content rule.
- Make sure that all attributes required by the entry's object classes and/or DIT content rule are present.
- Make sure that all single-valued attributes contained in the entry only have one value.
- Make sure that the entry's position in the directory information tree conforms with DIT structure rule definitions.

search attributes

The search attributes element of a search operation provides a way of representing the attribute that should be included in search result entry. In general, the set of search attributes is a list of zero or more attribute description for the attributes to return. If values are specified, then all user attribute and no operational attribute will be returned.

In addition to specific attribute descriptions, a number of special values can be provided with various meanings:

- The string `1.1` indicates that no attributes should be included in matching entries.
- The string `*` (the asterisk) indicates that all user attributes should be included in matching entries. This is needed if the server returns all user attributes in addition to one or more operational attributes.
- The string `+` (the plus sign) indicates that all operational attributes should be included in matching entries.
- An object class name can be provided, prefixed with the `@` character. This indicates that all attributes referenced by that object class should be included in matching entries.

search base DN

The search base DN is an element of the search operation that works in conjunction with the search scope to define the subtree of entries that should be considered when processing the search operation. Only entries at or below the search base DN and within the scope will be considered candidates for matching against the search filter.

search filter

See LDAP search filter.

search operation

The LDAP search operation can be used to identify entries in the Directory Server that match a given set of criteria. It may return zero or more entries, and also zero or more referrals.

The search request protocol op is defined as follows:

```
SearchRequest ::= [APPLICATION 3] SEQUENCE {
      baseObject      LDAPDN,
      scope           ENUMERATED {
          baseObject              (0),
          singleLevel              (1),
          wholeSubtree             (2),
          ...  },
     derefAliases    ENUMERATED {
          neverDerefAliases        (0),
          derefInSearching         (1),
          derefFindingBaseObj      (2),
          derefAlways              (3) },
     sizeLimit       INTEGER (0 ..  maxInt),
     timeLimit       INTEGER (0 ..  maxInt),
     typesOnly       BOOLEAN,
     filter          Filter,
     attributes      AttributeSelection }
```

The elements of the search request include:

- The search base DN, which specifies the location in the directory information tree in which to perform the search.
- The search scope, which specifies the scope of entries at or below the base DN to consider when processing the search.
- The dereference policy to use if any aliases are encountered during processing.
- The size limit, which specifies the maximum number of entries that should be returned from the search (or zero if there should not be any maximum number of entries).
- The time limit, which specifies the maximum length of time in seconds that the server should spend processing the search (or zero if there should not be a maximum number of entries).
- The typesOnly flag, which indicates whether the entries returned should include attribute types only or both types and values.
- The search filter, which specifies the criteria to use to identify matching entries.
- The search attributes that indicate which attributes should be included in matching entries, or an empty list to indicate that all user attribute should be returned.

There are three types of result elements that can be returned in response to a search request: zero or more search result entry, zero or more search result reference, and exactly one search result done message. The entries and references can be returned in any order (and with search entries and references interspersed), and the search result done message will come last to indicate that there are no more results.

The search result entry protocol op is defined as follows:

```
SearchResultEntry ::= [APPLICATION 4] SEQUENCE {
     objectName      LDAPDN,
     attributes      PartialAttributeList }

PartialAttributeList ::= SEQUENCE OF
                     partialAttribute PartialAttribute
```

Each search result entry includes the DN of the entry and zero or more attributes (potentially including only the attribute type names without the values if the `typesOnly` element of the request is true) as defined in the search attribute list.

The search result reference protocol op is defined as follows:

```
SearchResultReference ::= [APPLICATION 19] SEQUENCE
                          SIZE (1..MAX) OF uri URI
```

Each search result reference includes one or more LDAP URL specifying an alternate location in which the client may search for additional matching entries.

The search result done message is an LDAP result defined as follows:

```
SearchResultDone ::= [APPLICATION 5] LDAPResult
```

search result done

A search result done message is a message provided as part of a search operation to indicate that the search has completed and that there will be no more search result entry or search result reference messages.

search result entry

A search result entry is an entry returned as part of a search operation. It will contain at least the distinguished name of the entry, and can contain zero or more attributes. The attributes can contain only attribute type names or both types and values (based on the value of the typesOnly flag from the search request). The attributes returned can be based on the search attributes from the client request, but can be pared down based on the server's access control configuration.

search result reference

A search result reference provides a mechanism for returning information to clients as part of a search operation that indicates an alternate location in which the client may perform the search to locate additional matching entries. The alternate locations will be specified in the form of LDAP URLs.

search scope

The LDAP search scope indicates the set of entries at or below the search base DN that may be considered potential matches for a search operation.

There are four defined search scope values:

baseObject	This specifies that the search operation should only be performed against the entry specified as the search base DN. No entries below it will be considered. Consider a scenario of DIT, which has a baseObject scope with a search base DN of `dc=example,dc=com`.
singleLevel	This specifies that the search operation should only be performed against entries that are immediate subordinates of the entry specified as the search base DN. The base entry itself is not included, nor are any entries below the immediate subordinates of the search base entry.
wholeSubtree	This specifies that the search operation should be performed against the entry specified as the search base and all of its subordinates to any depth.
subordinateSubtree	This specifies that the search operation should be performed against all subordinate entries below the search base to any depth, but the search base entry itself should not be included.

Secure Hash Algorithm

The Secure Hash Algorithm (SHA) is a one-way message digest algorithm. There are actually two different forms of the Secure Hash Algorithm:

- SHA-1 is defined in RFC 3174 and generates a 160-bit digest.
- SHA-2 is defined in RFC 4634 and can be used to generate 256-bit, 384-bit, or 512-bit digests.

All forms of the Secure Hash Algorithm are considered stronger than the MD5 algorithm. There have been recent advancements that may indicate a weakening of the SHA-1 variant, but nevertheless there is no evidence to suggest that the way it is used in the directory server is under any danger, nor is there any concern about any of the SHA-2 encodings.

Secure Sockets Layer

The Secure Sockets Layer (SSL) is a mechanism for wrapping network communication in a security layer that can be used to encrypt communication between the client and the server. It also provides an integrity mechanism to ensure that the communication is not altered between the client and the server. The encryption is based on cryptography using certificate.

SSL was originally a proprietary protocol developed by Netscape Communications. It has since been standardized, but the name has been changed to Transport Security Layer. Nevertheless, SSL is still a commonly-used term to refer to this capability, and it is the term used throughout the directory server in order to avoid confusion with the StartTLS extended operation.

server-side sort control

The server-side sort control is a type of control that can be attached to a search operation to request that the results be sorted before they are returned to the client. It is defined in RFC 2891.

The request control has an OID of 1.2.840.113556.1.4.473 and the value is encoded as follows:

```
SortKeyList ::= SEQUENCE OF SEQUENCE {
           attributeType   AttributeDescription,
           orderingRule    [0] MatchingRuleId OPTIONAL,
           reverseOrder    [1] BOOLEAN DEFAULT FALSE }
```

For an example of using this control in a search request, see “To Search Using the Server-Side Sort Control” in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

The response control has an OID of 1.2.840.113556.1.4.474 and its value is encoded as follows:

```
SortResult ::= SEQUENCE {
     sortResult  ENUMERATED {
          success                   (0), -- results are sorted
          operationsError           (1), -- server internal failure
          timeLimitExceeded         (3), -- timelimit reached before
                                         -- sorting was completed
          strongAuthRequired        (8), -- refused to return sorted
                                        -- results via insecure
                                         -- protocol
          adminLimitExceeded       (11), -- too many matching entries
                                         -- for the server to sort
          noSuchAttribute          (16), -- unrecognized attribute
                                         -- type in sort key
          inappropriateMatching    (18), -- unrecognized or
                                         -- inappropriate matching
                                         -- rule in sort key
          insufficientAccessRights (50), -- refused to return sorted
                                         -- results to this client
          busy                     (51), -- too busy to process
          unwillingToPerform       (53), -- unable to sort
          other                    (80)
          },
     attributeType [0] AttributeDescription OPTIONAL }
```

simple authentication

Simple authentication is the process of authentication to the Directory Server using a distinguished name and password. This is done using an bind operation (and when the bind is performed using simple authentication, it is often called a "simple bind"). The client uses the provided DN to identify itself to the server, and the password is used to verify that the client is who it claims to be.

Note that simple authentication does not protect the password in any way, and therefore it is generally recommended that it only be used over a secure communication channel like that provided by Secure Sockets Layer or StartTLS.

Simple Authentication and Security Layer

The Simple Authentication and Security Layer (SASL) is an extensible framework that is primarily used for authentication users, but in some cases it may also be used for protecting the underlying communication channel. The core functionality of SASL is described in RFC 4422, but a number of SASL mechanisms are described in other specifications.

The SASL mechanisms supported by the directory server include:

ANONYMOUS	This mechanism doesn't actually authenticate users to the server, but can be used to destroy a previous authentication session.
CRAM-MD5	This mechanism provides a way for users to authenticate to the server using a password in a manner that does not expose the password itself. It is similar to, but weaker than, the DIGEST-MD5 SASL mechanism, and doesn't provide any way for ensuring connection integrity or confidentiality.
DIGEST-MD5	This mechanism provides a way for users to authenticate to the server using a password in a manner that does not expose the password itself. It is similar to, but stronger than, the CRAM-MD5 SASL mechanism, and also provides a way to ensure connection integrity and/or confidentiality.
EXTERNAL	This mechanism provides a way for users to authenticate to the server using information available outside of the LDAP communication that has been performed (for example, the certificate that a client presented when performing SSL or StartTLS negotiation).
GSSAPI	This mechanism provides a way for users to authenticate to the server using a Kerberos V5 session. It also provides a mechanism that can be used to ensure connection integrity and/or confidentiality.
PLAIN	This mechanism provides a way for users to authenticate to the server with a username and password. It is similar to the protection offered by simple authentication, but may be more convenient in that users can identify

themselves with a username rather than a distinguished name.

simple paged results control

The simple paged results control is a type of control that can be attached to a search operation to indicate that only a subset of the results should be returned. It may be used to iterate through the search results a page at a time. It is similar to the virtual list view control with the exception that it doesn't require the results to be sorted and can only be used to iterate sequentially through the search results.

The simple paged results control is defined in RFC 2696. The same control is used in both the search request and search result done messages. It has an OID of 1.2.840.113556.1.4.319, and the value is encoded as follows:

```
realSearchControlValue ::= SEQUENCE {
     size            INTEGER (0..maxInt),
                             -- requested page size from client
                             -- result set size estimate from server
     cookie          OCTET STRING
}
```

For an example of using this control in a search request, see “To Search Using the Simple Paged Results Control” in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

size limit

The server size limit is a configuration option that controls the maximum number of entries that may be returned from any single search operation. This is a server-wide setting and may be overridden by a per-user configuration in the `ds-rlim-size-limit` operational attribute in the user's entry.

The server size limit (or per-user value) may also be restricted by the size limit element in the search request message.

smart referral

A smart referral is a special type of entry that can be placed in the directory information tree that reference content in another server and/or location of the DIT. Smart referral entries contain the `referral` object class with one or more instances of the `ref` attribute containing LDAP URLs that should be used in the referral.

StartTLS extended operation

The StartTLS extended operation is a type of extended operation that can be used to initiate a TLS-secured communication channel over an otherwise clear-text connection. It allows clients to use the same network port for both secure and insecure communication.

The StartTLS extended operation is defined in RFC 4511 and further described in RFC 4513. It uses an OID of 1.3.6.1.4.1.1466.20037 with no value. The response includes an OID of 1.3.6.1.4.1.1466.20037 (the same as the request OID) with no value.

static group

A static group is a type of group in the directory server that defines its membership by providing an explicit set of distinguished name of the entry that are members of the group.

Static groups are very well supported by external clients, but are not as scalable as dynamic groups when handling large numbers of members.

structural object class

A structural object class is one of the primary object class types. A structural object class is special in that it defines the core type for any entry that contains it. An entry must have exactly one structural class (although that structural class may inherit from other structural or abstract classes).

The structural object class for an entry may be used by other schema elements for defining constraints on directory data. It may be used by a name form definition to control the attributes used in the RDN for the entry, and in turn by a DIT structure rule to control the types of parent entries that it may have. The structural object class may also be used by a DIT content rule to control the set of auxiliary classes and required, allowed, and prohibited attribute types for the entry.

subentry

See LDAP Subentry.

subschema subentry

A subschema subentry is a special entry within the Directory Server that provides information about the schema elements defined in the server. Attributes in this entry include:

`ldapSyntaxes`	The set of attribute syntaxes defined in the server schema.
`matchingRules`	The set of matching rules defined in the server schema.
`matchingRuleUse`	The set of matching rule uses defined in the server schema.
`attributeTypes`	The set of attribute types defined in the server schema.
`objectClasses`	The set of object classes defined in the server schema.
`nameForms`	The set of name forms defined in the server schema.
`dITContentRules`	The set of DIT content rules defined in the server schema.
`dITStructureRules`	The set of DIT structure rules defined in the server schema.

Note that all of these are operational attribute and therefore will not be returned unless explicitly requested.

Also note that it is technically possible for directory servers to have multiple subschema subentries with different sets of schema definitions that govern different portions of the directory information tree. The schema that applies to any given entry may be determined by retrieving the `subschemaSubentry` virtual attribute from that entry. The directory server currently supports only a single schema, and by default publishes that schema at `cn=schema`.

substring assertion

A substring assertion is the argument provided to a substring matching rule in the process of determining whether an attribute has any attribute value that matches a given substring.

The substring assertion contains at least one component from the following set:

- Zero or one subInitial element, which must appear at the beginning of the target value.
- Zero or more subAny elements, which may appear anywhere in the middle of the value. If there are multiple subAny elements, then a matching attribute value must contain all of the subAny elements in the order they appear in the substring assertion with no overlap (i.e., no character in an attribute value can be part of two different substring assertion components). If subInitial and/or subFinal components are present, then none of the subAny elements may overlap with them either.
- Zero or one subFinal element, which must appear at the end of the target value.

The substring assertion is used when processing a substring search filter.

substring index

A substring index is a type of index that is used to keep track of which entries contain specific substrings. Index keys for a substring index consist of six-character substrings taken from attribute values and the corresponding values are ID lists containing the entry IDs of the entries containing those substrings. The attribute's substring matching rule is used to normalize the values for the index keys, and substring indexes cannot be defined for attributes that do not contain substring matching rules.

substring search filter

A substring search filter is a type of search filter that can be used to identify entries that contain a value for a given attribute that matches a specified substring. The server will use a substring matching rule to make the determination.

The substring search filter must contain a substring assertion, which will have at least one component from the following types:

- A subInitial component, whose value should be contained at the start of any matching value. There may be either zero or one subInitial component in a substring filter.
- A set of subAny components, whose values should be contained anywhere in the matching value. There may be zero or more subAny components in a substring filter, and they should be contained in the value in the order they appear in the substring filter, after any subInitial component and before any subFinal component.
- A subFinal component, whose value should be contained at the end of a matching value. There may be either zero or one subFinal component in a substring filter.

The string representation of an LDAP substring filter comprises an opening parenthesis followed by the attribute name, an equal sign, the substring assertion with the individual components separated by asterisks, and the closing parenthesis. For example, a substring filter of `(cn=ab*def*mno*stu*yz)` contains a subInitial component of `ab`, subAny components of `def`, `mno`, and `stu`, and a subFinal component of `yz`.

subtree

There are two definitions for the term “subtree”.

The general definition for the term is simply a portion of the directory information tree, including an entry and all of its subordinates.

The term subtree is also described in RFC 3672 in the form of a subtree specification. A subtree specification provides a mechanism for grouping entries based on a given set of criteria.

subtree delete control

The subtree delete control is a type of control that can be attached to a delete operation that will allow the entry and all of its subordinate entries to be deleted. Normal delete operations may target only leaf entries, but the subtree delete control may be used to target non-leaf entries.

The subtree delete request control has an OID of `1.2.840.113556.1.4.805` with no value. There is no corresponding response control.

The following example shows the use of this control to delete the `ou=People,dc=example,dc=com` subtree.

```
$ ldapdelete -p 1389 -h localhost -D cn=directory manager -w password \
  -J 1.2.840.113556.1.4.805
ou=People,dc=example,dc=com
Processing DELETE request for ou=People,dc=example,dc=com
```

supported control

A supported control is a mechanism for identifying the request control supported by the Directory Server. The OID of these controls are listed in the `supportedControl` attribute of the server's root DSE.

For a list of all controls currently supported in the directory server, see “Supported LDAP Controls” in *Oracle Fusion Middleware Architecture Reference for Oracle Unified Directory*

supported extension

A supported extension is a mechanism for identifying the extended operation supported by the Directory Server. The OID of these extended operations are listed in the `supportedExtension` attribute of the server's root DSE.

For a list of all supported extensions for the directory server, see “Supported Extended Operations” in *Oracle Fusion Middleware Architecture Reference for Oracle Unified Directory*.

supported feature

A supported feature is a mechanism for identifying optional capabilities that the Directory Server supports. A number of the features supported by the server are listed in the `supportedFeatures` attribute of the server's root DSE, which lists the OID of the supported features.

Some of the supported features for the directory server include:

1.3.6.1.4.1.4203.1.5.1	Indicates that the server supports the use of the + indicator when requesting all operational attribute as specified in RFC 3673.
1.3.6.1.4.1.4203.1.5.2	Indicates that the server supports the ability to include one or more object class names in the set of search attributes as specified in RFC 4529.
1.3.6.1.1.14	Indicates that the server supports the increment modification type, which is part of the increment modify extension as described in RFC 4525.
1.3.6.1.4.1.4203.1.5.3	Indicates that the server supports LDAP true filters and LDAP false filters as described in RFC 4526.

synchronization

Data synchronization is a mechanism for keeping track of changes in the directory environment and allowing them to be reflected elsewhere.

The primary type of data synchronization provided by the directory server is replication.

T

task

A task provides a set of logic for performing some type of processing in the server. Tasks are generally used to perform administrative functions within the server. Examples of tasks available for use include:

- Adding a new file to the server schema
- Back up up the contents of a server back end
- Restore a previous backup
- Performing an LDIF import operation
- Performing an LDIF export operation
- Initializing a replica in the server replication environment
- Performing an in-core restart
- Performing a server shutdown

Tasks can be recurring, that is scheduled to execute at regular intervals according to a specific schedule. For example, backup tasks can be made recurring in order to back up the server data on a regular basis. See “Scheduling and Configuring Tasks” in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory* for information about scheduling tasks.

time limit

The server time limit is a configuration option that controls the maximum length of time in seconds that the server may spend processing a search operation. This is a server-wide setting and may be overridden by a per-user configuration in the `ds-rlim-time-limit` operational attribute in the user's entry.

The server time limit (or per-user value) may also be restricted by the time limit element in the search request message.

transaction

A transaction is a collection of one or more read and/or write operations that occur within a database. Transactions may be described by the acronym ACID, which stands for atomicity, consistency, isolation, and durability. The directory server uses transactions in the Berkeley DB Java Edition to ensure that multiple changes made as part of a single LDAP operation (for example, updates to both the id2entry database and to indexes).

Even though the Directory Server uses transactions internally for its operations in the database, it does not currently expose a transactional mechanism that allows clients to perform several operations as a single atomic unit. There is an Internet Draft that describes a potential mechanism for exposing transactions (draft-zeilenga-ldap-txn), but the directory server does not currently support this capability.

Transport Security Layer

The Transport Security Layer (TLS) is a mechanism for securing network communication between clients and servers. It is the name given to the standardized form of the Secure Sockets Layer.

In most cases, the term “SSL” is preferred over “TLS” because it is the more popular term, and also to avoid confusion with the StartTLS extended operation.

true filter

See “LDAP true filter” on page 73

trust manager provider

A trust manager provider is a component of the server that can provide information that can be used to determine whether to trust certificates presented to the server.

See the Trust Manager Provider Configuration for information about the trust manager providers available for use in the directory server.

typesOnly flag

The TypesOnly flag is an element of an search operation that indicates whether attributes returned as part of search result entry should include only the attribute description or both the attribute description and the attribute value.

U

unbind operation

The LDAP unbind operation is used to indicate that the client wants to disconnect from the server.

Note that the unbind operation cannot be used to destroy an authentication session while leaving the underlying connection established. If the client does not close the connection after sending an unbind request, then the server will. If there is a need to revert a connection to an unauthenticated state, then an anonymous bind operation should be performed.

The LDAP unbind request protocol op is defined as follows:

```
UnbindRequest ::= [APPLICATION 2] NULL
```

An unbind request does not contain any elements, and the server will not send a response to an unbind request.

unindexed search

An unindexed search is one that cannot be processed using the set of index defined in the server. It will necessitate iterating through most or all of the entries in the database.

Unindexed searches can be expensive for the server to process, users will only be allowed to perform unindexed searches if they have the `unindexed-search` privilege.

For more information, see “Indexing Directory Data” in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

UNIX crypt algorithm

The UNIX crypt algorithm is a standard mechanism for encoding user passwords using a DES-based encryption scheme that ultimately results in a one-way message digest. It is called the “UNIX crypt” algorithm because it has historically been used as the default mechanism for encoding passwords in UNIX-based systems.

Note that the UNIX crypt algorithm is considered weak because it is based on a 56-bit encryption algorithm and uses only a 12-bit salt. Therefore, it should only be used in cases where clients expect to be able to retrieve the password from the server and compare its value against what the user supplied instead of attempting to verify it using an bind operation.

unsolicited notification

An unsolicited notification is a type of extended operation message that is special in that the server generates this kind of message without any corresponding request from the client. It may be used to notify the client of some important information.

The directory server currently supports a single unsolicited notification: the notice of disconnection unsolicited notification, which can be used to inform the client that the server is closing the connection.

URL

See LDAP URL.

user attribute

A user attribute is an attribute type with an attribute usage of `userApplications`. User attributes are used for actually storing information in the directory, as opposed to operational attribute which are used for storing state information used for internal server processing.

Whenever a search operation does not request any specific attributes to be returned, then all user attributes in matching entries will be returned. An explicit value of * (the asterisk) may also be included to explicitly include all user attributes.

V

virtual attribute

A virtual attribute is a type of attribute in which the attribute value are not actually stored in the back end but are instead dynamically generated in some manner. The values can be obtained in various manners, depending on the type of virtual attribute. Some virtual attributes use a hard-coded value, while others compute their values at runtime based on some kind of logic.

See the Virtual Attribute Configuration for information about the types of virtual attributes available for use in the directory server.

virtual attributes only control

The virtual attributes only control requests that the server include only virtual attributes in matching entries. That is, real attributes are excluded from search result entries.

The virtual attributes only control has a request OID of `2.16.840.1.113730.3.4.19` and no value.

The following example shows a search on the base DN without the virtual attributes only control:

```
$ ldapsearch -p 1389 -D "cn=directory manager" -w password -b "dc=example,dc=com" \
  -s base "objectclass=*"
version: 1
dn: dc=example,dc=com
objectClass: domain
objectClass: top
dc: example
```

The following example shows the same search with the virtual attributes only control:

```
$ ldapsearch -p 1389 -D "cn=directory manager" -w password \
  -J "2.16.840.1.113730.3.4.19"   -b "dc=example,dc=com" -s base "objectclass=*"
version: 1
dn: dc=example,dc=com
```

virtual directory

A virtual directory is a type of network daemon that communicates with clients using LDAP but obtains the underlying data from a combination of different sources. Virtual directories may have a number of different capabilities, including:

- Providing an LDAP front end to a different repository, like a relational database or a flat file
- Providing a mechanism to merge data from multiple repositories

virtual list view control

The virtual list view (VLV) control can be attached to a search operation to indicate that only a subset of the results are to be returned. It can be used to iterate through the search results a page at a time. It is similar to the simple paged results control with the exception that it can be used to retrieve an arbitrary subset of the results from the server, and it requires that the search request also include the server-side sort control to ensure that the results are consistently sorted across requests.

The VLV control is defined in draft-ietf-ldapext-ldapv3-vlv-09. The request control has an OID of 2.16.840.1.113730.3.4.9 and the value is encoded as follows:

```
VirtualListViewRequest ::= SEQUENCE {
     beforeCount    INTEGER (0..maxInt),
     afterCount     INTEGER (0..maxInt),
     target       CHOICE {
          byOffset        [0] SEQUENCE {
               offset          INTEGER (1 .. maxInt),
               contentCount    INTEGER (0 .. maxInt) },
          greaterThanOrEqual [1] AssertionValue },
     contextID     OCTET STRING OPTIONAL }
```

The response control has an OID of 2.16.840.1.113730.3.4.10 and the value is encoded as shown below:

```
VirtualListViewResponse ::= SEQUENCE {
     targetPosition    INTEGER (0 .. maxInt),
     contentCount     INTEGER (0 .. maxInt),
     virtualListViewResult ENUMERATED {
          success (0),
          operationsError (1),
          protocolError (3),
          unwillingToPerform (53),
          insufficientAccessRights (50),
          timeLimitExceeded (3),
          adminLimitExceeded (11),
          innapropriateMatching (18),
          sortControlMissing (60),
          offsetRangeError (61),
         other(80),
          ... },
     contextID     OCTET STRING OPTIONAL }
```

For an example of using this control in a search request, see "Searching Using the Virtual List View Control" in *Oracle Fusion Middleware Administration Guide for Oracle Unified Directory*.

virtual static group

A virtual static group is a special type of group that appears to be static to external clients but obtains its membership information from another group (like a dynamic group) in the server.

Virtual static groups are primarily used in cases where a client application only supports static groups but have a very large number of members that are better suited for maintaining in a dynamic group.

VLV index

A virtual list view (VLV) index is a mechanism used by the Directory Server database that can be used to efficiently process searches with virtual list view control. A VLV index effectively notifies the server that a virtual list view, with specific query and sort parameters, will be performed. This index also allows the server to collect and maintain the information required to make using the virtual list view faster. A VLV index stores sorted blocks of ID lists, which are a set of entry IDs and the attribute values of the entry to sort on.

W

"Who Am I?" extended operation

The "Who Am I?" extended operation provides an extended operation for determining the authorization identity of a client connection. It is defined in RFC 4532.

The request OID for the "Who Am I?" extended operation is `1.3.6.1.4.1.4203.1.11.3`, and there should not be a request value. The response should not include a response OID, and the value should be a string containing the client's authorization identity (or it may be an empty string if the authorization identity is that of the anonymous user).

The information provided by the "Who Am I?" extended operation is similar to that provided by the authorization identity control except that it can be used at any time after the client has authenticated, whereas the authorization identity control can only be included with a bind request.

work queue

The Directory Server work queue is the mechanism that it uses to keep track of outstanding requests and ensuring that they are processed in an appropriate manner. The work queue functionality is provided by an extensible API, but the default implementation is relatively simple: a queue is serviced by a number of worker thread. As long as there are free worker threads, then the queue will generally remain empty. If all worker threads are busy, then subsequent requests will be placed in the work queue so that they are processed in a FIFO manner.

worker thread

A worker thread is a thread used to process requests in the Directory Server. Worker threads are associated with the work queue, and they will operate in a loop that includes picking up a request from the queue (waiting for a request to arrive if necessary), processing that request appropriate, and then returning to the queue for the next request.

workflow

A workflow defines the processing for a given naming context. The overall processing is split into a set of ordered and synchronized tasks, defined by *workflow elements*.

workflow element

A workflow element is the key building block of a workflow processing. It defines how the client request sent to the server will be treated. The workflow elements implement the main tasks in the Oracle Unified Directory proxy, including for example, load balancing and distribution.

writability mode

The writability mode of the Directory Server is used to control whether write operations are allowed. The writability mode configuration can be restricted to a single back end or it can apply to the entire server.

The following writability modes are available:

enabled	The server attempts to process all write operations
disabled	The server rejects all write operations

internal-only	The server attempts to process write operations initiated as internal operations or through synchronization but rejects any request coming from an external client

An entryDN is an operational attribute that provides a copy of the entry's current DN. Because a DN is not an attribute of the entry, it cannot be used to perform attribute value assertions. The entryDN provides a mechanism to access an entry's DN and is described in RFC 5020.