Password Policy for LDAP Directories

1. Abstract

Password policy as described in this document is a set of rules that controls how passwords are used and administered in LDAP directories. In order to improve the security of LDAP directories
and make it difficult for password cracking programs to break into directories, it is desirable to enforce a set of rules on password usage. These rules are made to ensure that users change their passwords periodically, passwords meet construction requirements, the re-use of old password is restricted, and users are locked out after a certain number of failed attempts.

2. Overview

LDAP-based directory services are currently accepted by many organizations as the access protocol for directories. The ability to ensure the secure read and update access to directory information throughout the network is essential to the successful deployment. Most LDAP implementations support many authentication schemes - the most basic and widely used is the simple authentication i.e., user DN and password. In this case, many LDAP servers have implemented some kind of policy related to the password used to authenticate. Among other things, this policy includes:

- Whether and when passwords expire.
- Whether failed bind attempts cause the account to be locked.
- If and how users are able to change their passwords.

In order to achieve greater security protection and ensure interoperability in a heterogeneous environment, LDAP needs to standardize on a common password policy model. This is critical to the successful deployment of LDAP directories.

2.1 Application of password policy

The password policy defined in this document can be applied to any attribute holding users’ passwords which are used for authentication by the use of the bind operation. Typically, it is applied to the userPassword attribute in the case of the LDAP simple authentication method [RFC-2251] or the case of password based SASL [RFC-2222] authentication such as CRAM-MD5 [RFC-2195] and HTTP-Digest [RFC-Digest].

The policy described in this document assumes that the password attribute holds a single value. No considerations are made for directories or systems that allow a user to maintain multiple passwords.

The password policy also defines some constraints to be checked during add or modify operations on these attributes. In this document, the term "user" represents any application which is an LDAP client using the directory to retrieve or store information.
identities (such as directory administrators) are not forced to comply with any of password policy. In this case, the password for a directory administrator never expires; the account is never locked, etc.

3. Articles of password policy

The following sections explain in general terms each aspect of the password policy defined in this document as well as the need for each. These policies are subdivided into the general groups of password usage and password modification. Implementation details are presented in Sections 6 and 7.

3.1. Password Usage Policy

This section describes policy enforced while users are using passwords to authenticate. The general focus of this policy is to minimize the threat of intruders once a password is in use.

3.1.1. Password Guessing limit

In order to prevent intruders from guessing a user’s password, a mechanism exists to track the number of failed authentication attempts, and take action when a limit is reached.

This policy consists of five parts:

- A configurable limit.
- A counter to track the number of failed authentication attempts.
- A timeframe in which the limit of consecutive failed authentication attempts must happen before action is taken.
- The action to be taken when the limit is reached. The action will either be nothing, or the account will be locked.
- An amount of time the account is locked (if it is to be locked). This can be indefinite.

3.2. Password Modification Policy

This section describes policy enforced while users are modifying passwords. The general focus of this policy is to ensure that when users add or change their passwords, the security and effectiveness of their passwords is maximized.

3.2.1. Password Expiration, Expiration Warning, and Grace binds
One of the key properties of a password is the fact that it is not well known. If a password is frequently changed, the chances of that user’s account being broken into are minimized.

Directory administrators may deploy a password policy that causes passwords to expire after a given amount of time - thus forcing users to change their passwords periodically.

As a side effect, there needs to be a way in which users are made aware of this need to change their password before actually being locked out of their accounts. One or both of the following methods handle this:

- The user is sent a warning sometime before his password is due to expire. If the user fails to heed this warning before the expiration time, his account will be locked.

- The user may bind to the directory a preset number of times after her password has expired. If she fails to change her password during one of her ‘grace’ binds, her account will be locked.

3.2.2. Password History

When the Password Expiration policy is used, an additional mechanism may be employed in order to prevent users from simply re-using a previous password (as this would effectively circumvent the expiration policy).

In order to do this; a history of used passwords is kept. The directory administrator sets the number of passwords to be stored at any given time. Passwords are stored in this history whenever the password is changed. Users aren’t allowed to specify any passwords that are in the history list while changing passwords.

3.2.3. Password Minimum Age

Users may circumvent the Password History mechanism by quickly performing a series of password changes. If they change their password enough times, their ‘favorite’ password will be pushed out of the history list.

This process may be made less attractive to users by employing a minimum age for passwords. If users are forced to wait 24 hours between password changes, they may be less likely to cycle through a history of 10 passwords.

3.2.4. Password Syntax and Minimum length
In order to prevent users from creating or updating passwords that are easy to guess, a password syntax policy may be employed. This policy consists of two general mechanisms – ensuring that passwords conform to a defined syntax and ensuring that they are of a minimum length.

Forcing a password to comply with a syntax may imply a variety of things including:

- Disallowing trivial or well-known words make up the password.
- Forcing a certain number of digits be used.
- Disallowing anagrams of the user’s name.

The implementation of this policy meets with the following problems:

- If the password to be added or updated is encrypted by the client before being sent, the server has no way of enforcing this policy. Therefore, the onus of enforcing this policy falls upon client implementations.
- There are no specific definitions of what ‘syntax checking’ means. This can lead to unexpected behavior in a heterogeneous environment.

3.2.5. User Defined Passwords

In some cases, it is desirable to disallow users from adding and updating their own passwords. This policy makes this functionality possible.

This implies that certain other policy, such as password expiration is not enforced.

3.2.6. Password Change After Reset

This policy forces the user to update her password after it has been set for the first time, or has been reset by the directory administrator.

3.2.7 Safe modification

As directories become more commonly used, it will not be unusual for clients to connect to a directory and leave the connection open for
an extended period. This opens up the possibility for an intruder to make modifications to a user’s password while that user’s computer is connected but unattended.

This policy forces the user to prove his identity by specifying the old password before modifying it.

3.3 Restriction of the Password Policy

The password policy has defined in this document can apply to any attribute containing a password. Because the state information in the user’s entry is on a per attribute basis and not per password attribute value, the server should enforce that the password attribute subject to the password, contains one and only one password value.

4. Schema used for Password Policy

The schema elements defined here fall into two general categories. A password policy object class is defined which contains a set of administrative password policy attributes, and a set of operational attributes are defined which hold general password policy state information for each user.

4.1. The pwdPolicy Object Class

This object class contains the attributes defining a password policy in effect for a set of users. See Section 8 subtrees of the DIT.

```
{ 1.3.6.1.4.1.42.2.27.8.2.1
   NAME 'pwdPolicy'
   AUXILIARY
   SUP top
   MUST ( pwdAttribute )
   MAY ( pwdMinAge $ pwdMaxAge $ pwdInHistory $ pwdCheckSyntax $ pwdMinLength $ pwdExpireWarning $ pwdGraceLoginLimit $ pwdLockout $ pwdLockoutDuration $ pwdMaxFailure $ pwdFailureCountInterval $ pwdMustChange $ pwdAllowUserChange $ pwdSafeModify )
```

4.2. Attribute Types used in the pwdPolicy ObjectClass

Following are the attribute types used by the pwdPolicy object class.

4.2.1 pwdAttribute

This holds the name of the attribute to which the password policy is applied. For example, the password policy may be applied to the
userPassword attribute.

( 1.3.6.1.4.1.42.2.27.8.1.1
  NAME 'pwdAttribute'
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

4.2.2 pwdMinAge

This attribute holds the number of seconds that must elapse between modifications to the password. If this attribute is not present, 0 seconds is assumed.

( 1.3.6.1.4.1.42.2.27.8.1.2
  NAME 'pwdMinAge'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )

4.2.3 pwdMaxAge

This attribute holds the number of seconds after which a modified password will expire.

If this attribute is not present, or if the value is 0 the password does not expire.

( 1.3.6.1.4.1.42.2.27.8.1.3
  NAME 'pwdMaxAge'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )

4.2.4 pwdInHistory

This attribute specifies the maximum number of used passwords stored in the pwdHistory attribute.

If this attribute is not present, or if the value is 0, used passwords are not stored in the pwdHistory attribute and thus may be reused.

( 1.3.6.1.4.1.42.2.27.8.1.4
  NAME 'pwdInHistory'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )

4.2.5 pwdCheckSyntax
This attribute indicates how the password syntax will be checked while being modified or added. If this attribute is not present, or if the value is ‘0’, syntax checking will not be enforced. A value of ‘1’ indicates that the server will check the syntax, and if the server is unable to check the syntax (due to a hashed password or other reasons) it will be accepted. A value of ‘2’ indicates that the server will check the syntax, and if the server is unable to check the syntax it will return an error refusing the password.

( 1.3.6.1.4.1.42.2.27.8.1.5
  NAME 'pwdCheckSyntax'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )

4.2.6 pwdMinLength

This attribute holds the minimum number of characters that must be used in a password, if syntax checking is enabled. If this attribute is not present, no minimum password length will be enforced. If the server is unable to check the length (due to a hashed password or otherwise), the server will, depending on the value of the pwdCheckSyntax attribute, either accept the password without checking it ('0' or '1') or refuse it ('2').

( 1.3.6.1.4.1.42.2.27.8.1.6
  NAME 'pwdMinLength'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )

4.2.7 pwdExpireWarning

This attribute specifies the maximum number of seconds before a password is due to expire that expiration warning messages will be returned to an authenticating user. If this attribute is not present, or if the value is 0 no warnings will be sent.

( 1.3.6.1.4.1.42.2.27.8.1.7
  NAME 'pwdExpireWarning'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )

4.2.8 pwdGraceLoginLimit
This attribute specifies the number of times an expired password can be used to access an account. If this attribute is not present or if the value is 0, the account cannot be accessed once the password has expired.

( 1.3.6.1.4.1.42.2.27.8.1.8
  NAME 'pwdGraceLoginLimit'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )

4.2.9 pwdLockout

This attribute indicates, when its value is "TRUE", that users will be locked out of the directory after a specified number of consecutive failed bind attempts. The maximum number of consecutive failed bind attempts is specified in pwdMaxFailure.

If this attribute is not present, or if the value is "FALSE", the account will not be locked when the number of failed bind attempts has been reached.

( 1.3.6.1.4.1.42.2.27.8.1.9
  NAME 'pwdLockout'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE )

4.2.10 pwdLockoutDuration

This attribute holds the number of seconds that an account will remain locked due to too many failed bind attempts. If this attribute is not present, or if the value is 0 the account will be locked until reset by an administrator.

( 1.3.6.1.4.1.42.2.27.8.1.10
  NAME 'pwdLockoutDuration'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )

4.2.11 pwdMaxFailure

This attribute specifies the number of consecutive failed bind attempts after which a users account will be locked. If this attribute is not present, or if the value is 0, the account will not be locked due to failed bind attempts and the value of pwdLockout will be ignored.
4.2.12 pwdFailureCountInterval

This attribute holds the number of seconds after which the password failures are purged from the failure counter, even though no successful authentication occurred.

If this attribute is not present, or if its value is 0, the failure counter is only reset by a successful authentication.

4.2.13 pwdMustChange

This attribute specifies with a value of "TRUE" that users must change their passwords when they first bind to the directory after a password is set or reset by the administrator. If this attribute is not present, or if the value is "FALSE", users are not required to change their password upon binding after the administrator sets or resets the password.

4.2.14 pwdAllowUserChange

This attribute indicates whether users can change their own passwords. If this attribute is not present, a value of "TRUE" is assumed.
This attribute specifies whether or not the existing password must be sent when changing a password. If this attribute is not present, a "FALSE" value is assumed.

```plaintext
( 1.3.6.1.4.1.42.2.27.8.1.15
   NAME 'pwdSafeModify'
   EQUALITY booleanMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
   SINGLE-VALUE )
```

### 4.3. Attribute Types for Password Policy State Information

Password policy state information must be maintained for each user. The information is located in each user entry as a set of operational attributes. These operational attributes are: `pwdChangedTime`, `pwdAccountLockedTime`, `pwdExpirationWarned`, `pwdFailureTime`, `pwdHistory`, `pwdGraceUseTime`, `pwdReset`.

Since the password policy could apply to several attributes used to store passwords, each of the above operational attributes must have an option to tell which attribute is applies to.

The password policy option is defined as the following:

```plaintext
pwd-<passwordAttribute>
```

For example, if the `pwdPolicy` object has for `pwdAttribute"userPassword"` then the `pwdChangedTime` operational attribute, in a user entry, will be like this:

```plaintext
pwdChangedTime;pwd-userPassword: 20000103121520Z
```

### 4.3.1 pwdChangedTime

This attribute specifies the time the entry’s password was last changed. This is used by the password expiration policy. If this attribute does not exist, the password will never expire.

```plaintext
( 1.3.6.1.4.1.42.2.27.8.1.16
   NAME 'pwdChangedTime'
   DESC 'The time the password was last changed'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
   EQUALITY generalizedTimeMatch
   ORDERING generalizedTimeOrderingMatch
   SINGLE-VALUE
   USAGE directoryOperation)
```

### 4.3.2 pwdAccountLockedTime

This attribute holds the time that the user’s account was locked. A 0 value means that the account has been locked permanently, and that...
only an administrator can unlock the account.

( 1.3.6.1.4.1.42.2.27.8.1.17
  NAME 'pwdAccountLockedTime'
  DESC 'The time an user account was locked'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
  EQUALITY generalizedTimeMatch
  ORDERING generalizedTimeOrderingMatch
  SINGLE-VALUE
  USAGE directoryOperation)

4.3.3 pwdExpirationWarned

This attribute contains the time when the password expiration warning was first sent to the client. The password will expire in the pwdExpireWarning time.

( 1.3.6.1.4.1.42.2.27.8.1.18
  NAME 'pwdExpiration Warned'
  DESC 'The time the user was first warned about the coming expiration of the password'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
  EQUALITY generalizedTimeMatch
  ORDERING generalizedTimeOrderingMatch
  SINGLE-VALUE
  USAGE directoryOperation )

4.3.4 pwdFailureTime

This attribute holds the timestamps of the consecutive authentication failures.

( 1.3.6.1.4.1.42.2.27.8.1.19
  NAME 'pwdFailureTime'
  DESC 'The timestamps of the last consecutive authentication failures'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
  EQUALITY generalizedTimeMatch
  ORDERING generalizedTimeOrderingMatch
  USAGE directoryOperation )

4.3.5 pwdHistory

This attribute holds a history of previously used passwords.

Values of this attribute are transmitted in string format as given by the following ABNF:
pwdHistory = time "#" syntaxOID "#" length "#" data

time = <generalizedTimeString as specified in 6.14 of [RFC2252]>
syntaxOID = numericoid ; the string representation of the ; dotted-decimal OID that defines the ; syntax used to store the password. ; numericoid is described in 4.1 of ; [RFC2252].

length = numericstring ; the number of octets in data. ; numericstring is described in 4.1 of ; [RFC2252].

data = <octets representing the password in the format specified by syntaxOID>.

This format allows the server to store, and transmit a history of passwords that have been used. In order for equality matching to function properly, the time field needs to adhere to a consistent format. For this purpose, the time field MUST be in GMT format.

{ 1.3.6.1.4.1.42.2.27.8.1.20
   NAME 'pwdHistory'
   DESC 'The history of user’s passwords'
}

4.3.6 pwdGraceUseTime

This attribute holds the timestamps of grace login once a password has expired.

{ 1.3.6.1.4.1.42.2.27.8.1.21
   NAME 'pwdGraceUseTime'
   DESC 'The timestamps of the grace login once the password has expired'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
   EQUALITY generalizedTimeMatch
   SINGLE-VALUE
   USAGE directoryOperation)

4.3.7 pwdReset

This attribute holds a flag to indicate if the password has been reset and therefore must be changed by the user on first authentication.
5. Controls used for Password Policy

This section details the controls used while enforcing password policy. A request control is defined that is sent by a client with a request operation in order to elicit a response control. The response control contains various warnings and errors associated with password policy.

5.1 Request Control

This control MAY be sent with any LDAP request message in order to convey to the server that this client is aware of, and can process the response control described in this document. When a server receives this control, it will return the response control when appropriate and with the proper data.

The controlType is 1.3.6.1.4.1.42.2.27.8.5.1 and the criticality MUST be FALSE. There is no controlValue.

passwordPolicyRequest

controlType: 1.3.6.1.4.1.42.2.27.8.5.1
criticality: FALSE
controlValue: None

5.2 Response Control

If the client has sent a passwordPolicyRequest control, the server sends this control with the following operation responses: bindResponse, modifyResponse, addResponse, compareResponse and possibly extendedResponse, to inform of various conditions and MAY be sent with other operations (in the case of the changeAfterReset error).

passwordPolicyResponse

controlType: 1.3.6.1.4.1.42.2.27.8.5.1
criticality: FALSE
controlValue: an OCTET STRING, whose value is the BER encoding of the following type:
PasswordPolicyResponseValue ::= SEQUENCE {
  warning   [0] CHOICE OPTIONAL {
    timeBeforeExpiration [0] INTEGER (0 .. maxInt),
    graceLoginsRemaining [1] INTEGER (0 .. maxInt) }
  error     [1] ENUMERATED OPTIONAL {
    passwordExpired       (0),
    accountLocked         (1),
    changeAfterReset      (2),
    passwordModNotAllowed (3),
    mustSupplyOldPassword (4),
    invalidPasswordSyntax (5),
    passwordTooShort      (6),
    passwordTooYoung      (7),
    passwordInHistory     (8) }
}

The `timeBeforeExpiration` warning specifies the number of seconds before a password will expire. The `graceLoginsRemaining` warning specifies the remaining number of time a user will be allowed to authenticate with an expired password. The `passwordExpired` error signifies that the password has expired and must be reset. The `changeAfterReset` error signifies that the password must be changed before the user will be allowed to perform any operation other than bind and modify. The `passwordModNotAllowed` is set when a user is restricted from changing her password. The `invalidPasswordSyntax` error is set when a password doesn’t pass syntax checking. The `passwordTooYoung` is set if the age of the password to be modified is not yet old enough.

Typically, only either a warning or an error will be encoded though there may be exceptions. For example, if the user is required to change a password after the administrator set it, and the password will expire in a short amount of time, the control may include the `timeBeforeExpiration` warning and the `changeAfterReset` error.

6. Server Implementation by LDAP operation

The following sections contain detailed instructions that refer to attributes of the `pwdPolicy` object class. When doing so, the attribute of the `pwdPolicy` object that governs the entry being discussed is implied.

The server SHOULD enforce that the password attribute subject to a password policy as defined in this document, contains one and only one password value.

The scenarios in the following operations assume that the client has attached a `passwordPolicyRequest` control to the request message of
the operation. In the event that the passwordPolicyRequest control was not sent, no passwordPolicyRequest control is returned. All other instructions remain the same.

6.1 Bind Operation

When processing a bind request, the server MUST perform the following steps:

1. Check for a locked account:

   If the value of the pwdAccountLockedTime attribute is 0, or if the current time is less than the value of the pwdAccountLockedTime attribute added to the value of the pwdLockoutDuration, the account is locked.

   If the account is locked, the server MUST send a bindResponse to the client with the resultCode: unwillingToPerform (53), and MUST include the passwordPolicyResponse in the controls field of the bindResponse message with the error: accountLocked (1).

   If the account is not locked, the server MUST proceed with the bind operation.

2. Check the result of the bind operation:

   If the bind operation succeeds with authentication, the server MUST do the following:

   A. Delete the pwdFailureTime attribute.

   B. Check whether the password must be changed now.

      If the pwdMustChange attribute is set to TRUE, and if the pwdReset attribute is set to TRUE, the password must be changed now.

      If the password must be changed now, the server MUST send a bindResponse to the client with the resultCode: success (0), and MUST include the passwordPolicyResponse in the controls field of the bindResponse message with the warning: changeAfterReset specified.

      The server MUST then disallow all operations issued by this user except modify password, bind, unbind, abandon and StartTLS extended operation.

      If the password does not need to be changed now, the operation
proceeds.

C. Check for password expiration

The password has expired when either of the following conditions is met:

- If the value of the pwdExpireWarning attribute is 0, the server subtracts the current time from the time stored in pwdChangedTime to arrive at the password's age. If the age is greater than the value in the pwdMaxAge attribute, the password has expired.

- If the value of the pwdExpireWarning attribute is non-zero, and the pwdExpirationWarned attribute is present and has a time value, the server subtracts the current time from the time stored in the pwdExpirationWarned to arrive at the first warning age. If the age is greater than the value in the pwdExpireWarning attribute, the password has expired.

If the password has expired, the server MUST check for remaining grace logins.

If the pwdGraceUseTime attribute is present, the server MUST count the number of values in that attribute and subtract it from the pwdGraceLoginLimit. A positive result specifies the number of remaining grace logins.

If there are remaining grace logins, the server MUST add a new value with the current time in pwdGraceUseTime. Then it MUST send a bindResponse with the resultCode: success (0), and MUST include the passwordPolicyResponse in the controls field of the bindResponse message with the warning: graceLoginsRemaining choice set to the number of grace logins left.

If there are no remaining grace logins, the server MUST send a bindResponse with the resultCode: invalidCredentials (49), and MUST include the passwordPolicyResponse in the controls field of the bindResponse message with the error: passwordExpired (0) set.

If the password has not expired, execution continues.

D. Calculates whether the time before expiration warning should be sent.
If the pwdExpireWarning attribute is present and contains a value, the server MUST perform the following steps.

If the pwdExpirationWarning attribute is present and has a time value, the warning time is the value of the pwdExpirationWarning attribute plus the value of the pwdExpireWarning attribute minus the current time.

If the pwdExpirationWarning attribute is not present, the server MUST subtract the current time from the time stored in pwdChangedTime to arrive at the password’s age. If the age is greater than the value of the pwdMaxAge attribute minus the value of the pwdExpireWarning attribute, the server MUST set the current time as the value of the pwdExpirationWarning attribute, and the warning time is the value of pwdMaxAge minus the password’s age.

If the warning time is a positive number, the server MUST send a bindResponse with the resultCode: success (0), and MUST include the passwordPolicyResponse in the controls field of the bindResponse message with the warning: timeBeforeExiration set to the value as described above.

If the warning time is zero, or wasn’t calculated, the server MUST send a bindResponse with the resultCode: success (0), and MUST include the passwordPolicyResponse with nothing in the SEQUENCE.

If the pwdExpireWarning attribute is not present, the server MUST send a bindResponse with the resultCode: success (0), and MUST include the passwordPolicyResponse with nothing in the SEQUENCE.

If the bind operation fails authentication due to invalid credentials, the server MUST do the following:

A. Add the current time as a value of the pwdFailureTime attribute.

B. If the pwdLockout attribute is TRUE, the server MUST also do the following:

Count the number of values in the pwdFailureTime attribute that are younger than pwdFailureCountInterval.

If the number of these failures is greater or equal to the pwdMaxFailure attribute, the server MUST lock the account by setting the value of the pwdAccountLockedTime attribute to the current time. After locking the account, the server...
MUST send a bindResponse to the client with the resultCode: unwillingToPerform (53), and MUST include the passwordPolicyResponse in the controls field of the bindResponse message with the error: accountLocked (1).

If the number of failures is less than the pwdMaxFailure attribute, operation proceeds.

If the pwdLockout attribute is FALSE, operation continues.

C. Failure times that are old by more than pwdFailureCountInterval, are purged from the pwdFailureTime attribute.

D. If no errors were returned, the server MUST send a bindResponse with the resultCode: success (0), and MUST include the passwordPolicyResponse with nothing in the SEQUENCE.

6.2. Modify Operation

Because the password is stored in an attribute, the modify operation may be used to create or update a password. But some alternate mechanisms have been defined or may be defined, such as the LDAP Password Modify Extended Operation [RFC-3062].

The following instructions are to be followed when processing any request that includes changes to a subject to a policy as defined in this document. The instructions assume that the request is a Modify operation, they SHOULD be adapted to the actual request (for example replace modifyResponse by extendedResponse and PasswdModifyResponseValue in the case of the Password Modify Extended Request).

While processing a password modification, the server MUST perform the following steps:

1. Check the pwdSafeModify attribute. If set to TRUE, the server MUST ensure that the modify operation contains a delete action before the add or replace action, and that the delete action specifies the existing password. If the delete action is not specified, the server MUST abort the modification and then MUST send a modifyResponse to the client with the resultCode: unwillingToPerform (53), and MUST include the passwordPolicyResponse in the controls field of the modifyResponse message with the error: mustSupplyOldPassword (4).
If the Password Modify Extended Request is used, the server MUST ensure that the old password was specified in the PasswdModifyRequestValue. If the old password is absent, the server MUST abort the modification and MUST return an error as described above.

2. Check the value of the pwdMustChange attribute. If TRUE, the server MUST check the pwdReset attribute in the user’s entry, to see if a Directory Administrator has reset the password. If so, it MUST ensure that the modify operation contains no modifications other than the modification of the password attribute. If other modifications exist, the server MUST send a modifyResponse to the client with the resultCode: unwillingToPerform (53), and MUST include the passwordPolicyResponse in the controls field of the modifyResponse message with the error: changeAfterReset (2).

3. Check to see whether the bound identity has sufficient rights to modify the password. If the bound identity is a user changing her own password, this MAY be done by either the use of an access control mechanism, or by checking the pwdAllowUserChange attribute. Otherwise an access control mechanism MUST be used. The determination of this is implementation specific. If the user is not allowed to change her password, the server MUST send a modifyResponse to the client with the resultCode: unwillingToPerform (53), and MUST include the passwordPolicyResponse in the controls field of the modifyResponse message with the error: passwordModNotAllowed (3).

4. Check the value of the pwdMinAge attribute. If it is set to a non-zero value, the server MUST subtract the current time from the value of the pwdChangedTime attribute to arrive at the password’s age. If the password’s age is less than the value of the pwdMinAge attribute, the password is too young to modify. In this case, the server MUST send a modifyResponse to the client with the resultCode: constraintViolation (19), and MUST include the passwordPolicyResponse in the controls field of the modifyResponse message with the error: passwordTooYoung (7).

5. Check the value of the pwdCheckSyntax attribute.

    If the value is non-zero, The server:

    A. MUST ensure that the password meets the syntax restrictions enforced by the server. This enforcement is implementation specific.
If the server is unable to check the syntax (due to a hashed password or otherwise), the value of pwdCheckSyntax is evaluated. If the value is 1, operation MUST continue. If the value is 2, the server MUST send a modifyResponse to the client with the resultCode: constraintViolation (19), and MUST include the passwordPolicyResponse in the controls field of the modifyResponse message with the error: invalidPasswordSyntax (5).

If the server is able to check the password syntax, and the check fails, the server MUST send a modifyResponse to the client with the resultCode: constraintViolation (19), and MUST include the passwordPolicyResponse in the controls field of the modifyResponse message with the error: invalidPasswordSyntax (5).

B. MUST Check the value of the pwdMinLength attribute. If the value is non-zero, it MUST ensure that the new password is of at least the minimum length.

If the server is unable to check the length (due to a hashed password or otherwise), the value of pwdCheckSyntax is evaluated. If the value is 1, operation MUST continue. If the value is 2, the server MUST send a modifyResponse to the client with the resultCode: constraintViolation (19), and MUST include the passwordPolicyResponse in the controls field of the modifyResponse message with the error: passwordTooShort (5).

If the server is able to check the password length, and the check fails, the server MUST send a modifyResponse to the client with the resultCode: constraintViolation (19), and MUST include the passwordPolicyResponse in the controls field of the modifyResponse message with the error: passwordTooShort (5).

6. Check the value of the pwdInHistory attribute. If the value is non-zero, the server MUST check whether this password exists in the entry’s pwdHistory attribute or in the current password attribute. If the password does exist in the pwdHistory attribute or in the current password attribute, the server MUST send a modifyResponse to the client with the resultCode: constraintViolation (19), and MUST include the passwordPolicyResponse in the controls field of the modifyResponse message with the error: passwordInHistory (8).

If the steps have completed without causing an error condition, the server MUST follow the following steps in order to update the
necessary password policy state attributes:

7. Check the value of the pwdMaxAge attribute. If the value is non-zero, or if the value of the pwdMinAge attribute is non-zero, the server MUST update the pwdChangedTime attribute on the entry to the current time.

8. If the value of the pwdInHistory attribute is non-zero, the server MUST add the previous password to the pwdHistory attribute. If the number of attributes held in the pwdHistory attribute exceeds the value of pwdInHistory, the server MUST remove the oldest excess passwords.

9. Remove the pwdFailureTime, pwdReset, pwdGraceUseTime and pwdExpirationWarned attributes from the user’s entry if they exist.

The server MUST then apply the modify operation.

6.3 Add Operation

The password MAY be set during an Add operation. If it is, the server MUST perform the following steps while processing the add operation. Note that these are essentially duplicates of steps 3, 5 and 7 from Section 6.2 with the exception that pwdAllowUserChange is not checked.

1. Check to see whether the bound identity has sufficient rights to modify the password. This MAY be done by the use of an access control mechanism. If the user is not allowed to add this password, the server MUST send an addResponse to the client with the resultCode: unwillingToPerform (53), and MUST include the passwordPolicyResponse in the controls field of the addResponse message with the error: passwordModNotAllowed (3).

2. Check the value of the pwdCheckSyntax attribute.

   If the value is non-zero, The server:

   A. MUST ensure that the password meets the syntax restrictions enforced by the server. This enforcement is implementation specific.

      If the server is unable to check the syntax (due to a hashed password or otherwise), the value of pwdCheckSyntax MUST be evaluated. If the value is 1, operation MUST continue. If the value is 2, the server MUST send an addResponse to the client with the resultCode: constraintViolation (19), and MUST
include the passwordPolicyResponse in the controls field of the addResponse message with the error: invalidPasswordSyntax (5).

If the server is able to check the password syntax, and the check fails, the server MUST send an addResponse to the client with the resultCode: constraintViolation (19), and MUST include the passwordPolicyResponse in the controls field of the addResponse message with the error: invalidPasswordSyntax (5).

B. MUST check the value of the pwdMinLength attribute. If the value is non-zero, it MUST ensure that the new password is of at least the minimum length.

If the server is unable to check the length (due to a hashed password or otherwise), the value of pwdCheckSyntax MUST be evaluated. If the value is 1, operation MUST continue. If the value is 2, the server MUST send an addResponse to the client with the resultCode: constraintViolation (19), and MUST include the passwordPolicyResponse in the controls field of the addResponse message with the error: passwordTooShort (5).

If the server is able to check the password length, and the check fails, the server MUST send an addResponse to the client with the resultCode: constraintViolation (19), and MUST include the passwordPolicyResponse in the controls field of the addResponse message with the error: passwordTooShort (5).

If the steps above have completed without causing an error condition, the server MUST follow the steps below in order to update the necessary password policy state attributes.

3. Check the value of the pwdMaxAge attribute. If the value is non-zero, or if the value of the pwdMinAge attribute is non-zero, the server MUST update the pwdChangedTime attribute on the entry to the current time.

6.4 Compare Operation

The compare operation MAY be used to compare a password. This might be performed when a client wishes to verify that user’s supplied password is correct. An example of this is an LDAP HTTP authentication redirector. It may be desirable to use this rather than performing a bind operation in order to reduce possible overhead involved in performing a bind. Access Controls SHOULD be used to restrict this comparison from being made.

If a server supports this behavior, it MUST comply with the
following. Otherwise the password policy described in this document may be circumvented.

While comparing password attributes, the server MUST perform the following steps:

1. Check for a locked account:

   If the value of the pwdAccountLockedTime attribute is 0, or if the current time is less than the value of the pwdAccountLockedTime attribute added to the value of the pwdLockoutDuration, the account is locked.

   If the account is locked, the server MUST send a compareResponse to the client with the resultCode: compareFalse (5), and MUST include the passwordPolicyResponse in the controls field of the compareResponse message with the error: accountLocked (1).

   If the account is not locked, the server MUST proceed with the compare operation.

2. If Access Controls permit, the server MUST proceed with compare operation and MUST check the result.

   If the result of the compare operation is true, the server MUST do the following:

   A. Delete the pwdFailureTime attribute.

   B. Check for password expiration

       The password has expired when either of the following conditions is met:

       - If the value of the pwdExpireWarning attribute is 0, the server MUST subtract the current time from the time stored in pwdChangedTime to arrive at the password’s age. If the age is greater than the value in the pwdMaxAge attribute, the password has expired.

       - If the value of the pwdExpireWarning attribute is non-zero, and the pwdExpirationWarned attribute is present and has a time value, the server MUST subtract the current time from the time stored in the pwdExpirationWarned to arrive at the first warning age. If the age is greater than the value in the pwdExpireWarning attribute, the password has expired.
If the password has expired, the server MUST check for remaining grace logins.

If the pwdGraceUseTime attribute is present, the server MUST count the number of values in that attribute and MUST subtract it from the pwdGraceLoginLimit. A positive result specifies the number of remaining grace logins.

If there are remaining grace logins, the server MUST add a new value with the current time in pwdGraceUseTime. Then it MUST send a compareResponse with the resultCode: compareTrue (6), and MUST include the passwordPolicyResponse in the controls field of the compareResponse message with the warning: graceLoginsRemaining choice set to the number of grace logins left.

If there are no remaining grace logins, the server MUST send a compareResponse with the resultCode: compareFalse (5), and MUST include the passwordPolicyResponse in the controls field of the compareResponse message with the error: passwordExpired (0) set.

If the password has not expired, execution MUST continue.

C. Calculate whether the time before expiration warning should be sent.

If the pwdExpireWarning attribute is present and contains a value, the server MUST perform the following steps.

If the pwdExpirationWarned attribute is present and has a time value, the warning time is the value of the pwdExpirationWarned attribute plus the value of the pwdExpireWarning attribute minus the current time.

If the pwdExpirationWarned attribute is not present, the server MUST subtract the current time from the time stored in pwdChangedTime to arrive at the password’s age. If the age is greater than the value of the pwdMaxAge attribute minus the value of the pwdExpireWarning attribute, the server MUST set the current time as the value of the pwdExpirationWarned attribute, and the warning time is the value of pwdMaxAge minus the password’s age.

If the warning time is a positive number, the server MUST send a compareResponse with the resultCode: compareTrue (6), and MUST include the passwordPolicyResponse in the controls field of the compareResponse message with the
warning: timeBeforeExiration set to the value as described above.

If the warning time is zero, or wasn’t calculated, the server MUST send a compareResponse with the resultCode: compareTrue (6), and MUST include the passwordPolicyResponse with nothing in the SEQUENCE.

If the pwdExpireWarning attribute is not present, the server MUST send a compareResponse with the resultCode: compareTrue (6), and MUST include the passwordPolicyResponse with nothing in the SEQUENCE.

If the result of the compare operation is false, the server MUST do the following:

A. Add the current time as a value of the pwdFailureTime attribute.

B. If the pwdLockout attribute is TRUE, the server MUST do the following:

   Count the number of values in the pwdFailureTime attribute that are younger than pwdFailureCountInterval.

   If the number of these failures is greater or equal to the pwdMaxFailure attribute, the server MUST lock the account by setting the value of the pwdAccountLockedTime attribute to the current time. After locking the account, the server MUST send a compareResponse to the client with the resultCode: compareFalse (5), and MUST include the passwordPolicyResponse in the controls field of the compareResponse message with the error: accountLocked (1).

   If the number of failures is less than the pwdMaxFailure attribute, operation MUST proceed.

   If the pwdLockout attribute is FALSE, operation MUST continue.

C. Failure times that are old by more than pwdFailureCountInterval, MUST be purged from the pwdFailureTime attribute.
D. If no errors were returned, the server MUST send a compareResponse with the resultCode: compareTrue (6), and MUST include the passwordPolicyResponse with nothing in the SEQUENCE.

7. Client Implementation by LDAP operation

These sections illustrate possible scenarios for each LDAP operation and define the types of responses that identify those scenarios.

The scenarios in the following operations assume that the client attached a passwordPolicyRequest control to the request message of the operation, and thus MAY receive a passwordPolicyResponse control in the response message. In the event that the passwordPolicyRequest control was not sent, no passwordPolicyRequest control is returned. All other instructions remain the same.

7.1. Bind Operation

For every bind response received, the client MUST check the resultCode of the bindResponse and MUST check for a passwordPolicyResponse to determine if any of the following conditions are true and MAY prompt the user accordingly.

1. The password failure limit has been reached and the account is locked. The user needs to retry later or contact the directory administrator to reset the password.

   resultCode: unwillingToPerform (53)
   passwordPolicyResponse: error: accountLocked (1)

2. The user is binding for the first time after the directory administrator set the password. In this scenario, the client SHOULD prompt the user to change his password immediately.

   resultCode: success (0)
   passwordPolicyResponse: error: changeAfterReset (2)

3. The password has expired but there are remaining grace logins. The user needs to change it.

   resultCode: success (0)
   passwordPolicyResponse: warning: graceLoginsRemaining

4. The password has expired and there are no more grace logins. The user MUST contact the directory administrator in order to have its password reset.
resultCode: invalidCredentials (49)  
passwordPolicyResponse: error: passwordExpired (0)

5. The user’s password will expire in n number of seconds.

resultCode: success (0)  
passwordPolicyResponse: warning: timeBeforeExpiration

7.2 Modify Operations

7.2.1 Modify Request

If the application or client encrypts the password prior to sending it in a password modification operation (whether done through modifyRequest or another password modification mechanism), it SHOULD check the values of the pwdMinLength, and pwdCheckSyntax attributes and SHOULD enforce these policies.

7.2.2 Modify Response

If the modifyRequest operation was used to change the password, or if another mechanism is used --such as an extendedRequest-- the modifyResponse or other appropriate response MAY contain information pertinent to password policy. The client MUST check the resultCode of the response and MUST check for a passwordPolicyResponse to determine if any of the following conditions are true and optionally notify the user of the condition.

1. The user attempted to change her password without specifying the old password but the password policy requires this.
   
   resultCode: unwillingToPerform (53)  
   passwordPolicyResponse: error: mustSupplyOldPassword (4)

2. The user MUST change her password before submitting any other LDAP requests.

   resultCode: unwillingToPerform (53)  
   passwordPolicyResponse: error: changeAfterReset (2)

3. The user doesn’t have sufficient rights to change his password.

   resultCode: unwillingToPerform (53)  
   passwordPolicyResponse: error: passwordModNotAllowed (3)
4. It is too soon after the last password modification to change the password.

   resultCode:              constraintViolation (19)
   passwordPolicyResponse:  error: passwordTooYoung (7)

5. The password failed syntax checking.

   resultCode:              constraintViolation (19)
   passwordPolicyResponse:  error: invalidPasswordSyntax (5)

6. The length of the password is too short.

  resultCode:              constraintViolation (19)
   passwordPolicyResponse:  error: passwordTooShort (6)

7. The password has already been used; the user MUST choose a different one.

   resultCode:              constraintViolation (19)
   passwordPolicyResponse:  error: passwordInHistory (8)

7.3 Add Operation

   If a password is specified in an addRequest, the client MUST check the resultCode of the addResponse and MUST check for a passwordPolicyResponse to determine if any of the following conditions are true and may prompt the user accordingly.

1. The user doesn't have sufficient rights to add this password.

   resultCode:              unwillingToPerform (53)
   passwordPolicyResponse:  error: passwordModNotAllowed (3)

2. The password failed syntax checking.

   resultCode:              constraintViolation (19)
   passwordPolicyResponse:  error: invalidPasswordSyntax (5)

3. The length of the password is too short.

   resultCode:              constraintViolation (19)
   passwordPolicyResponse:  error: passwordTooShort (6)

7.4. Compare Operation
When a compare operation is used to compare a password, the client MUST check the resultCode of the compareResponse and MUST check for a passwordPolicyResponse to determine if any of the following conditions are true and MAY prompt the user accordingly. These conditions assume that the result of the comparison was true.

1. The password failure limit has been reached and the account is locked. The user needs to retry later or contact the directory administrator to reset the password.

   resultCode: compareFalse (5)
   passwordPolicyResponse: error: accountLocked (1)

2. The password has expired but there are remaining grace logins. The user needs to change it.

   resultCode: compareTrue (6)
   passwordPolicyResponse: warning: graceLoginsRemaining

3. The password has expired and there are no more grace logins. The user MUST contact the directory administrator to reset the password.

   resultCode: compareFalse (5)
   passwordPolicyResponse: error: passwordExpired (0)

4. The user’s password will expire in n number of seconds.

   resultCode: compareTrue (6)
   passwordPolicyResponse: warning: timeBeforeExpiration

7.4 Other Operations

For operations other than bind, unbind, abandon, search or StartTLS, the client MUST check the following result code and control to determine if the user needs to change the password immediately.

1. The user needs to change password. The user SHOULD be prompted to change the password immediately.

   resultCode: unwillingToPerform (53)
   passwordPolicyResponse: error: changeAfterReset (2)

8. Administration of a Password Policy

A password policy MUST be defined for a particular subtree of the DIT by adding to an LDAP subentry whose immediate superior is the root of the subtree, the pwdPolicy auxiliary object class.
The scope of the password policy is the same as the default scope of an LDAP subentry as defined in section 5.1.2 of [SubEntry].

It is possible to define password policies for different password attributes within the same pwdPolicy entry, by specifying multiple values of the pwdAttribute. But password policies could also be in separate sub entries as long as they are contained under the same LDAP subentry.

Modifying the password policy MUST not result in any change in users’ entries to which the policy applies.

It SHOULD be possible to overwrite the password policy for one user by defining a new policy in a subentry of the user entry.

9. Password Policy and Replication

The pwdPolicy object defines the password policy for a portion of the DIT and MUST be replicated on all the replicas of this subtree, as any subentry would be, in order to have a consistent policy among all replicated servers.

The elements of the password policy that are related to the users are stored in the entry themselves as operational attributes. As these attributes are subject to modifications even on a read-only replica, replicating them must be carefully considered.

The pwdChangedTime attribute MUST be replicated on all replicas, to allow expiration of the password.

The pwdReset attribute MUST be replicated on all replicas, to deny access to operations other than bind and modify password.

The pwdHistory attribute MUST be replicated to writable replicas. It doesn’t have to be replicated to a read-only replica, since the password will never be directly modified on this server.

The pwdAccountLockedTime, pwdExpirationWarned, pwdFailureTime and pwdGraceUseTime attributes MUST be replicated to writable replicas, making the password policy global for all servers.

When the user entry is replicated to a read-only replica, these attributes SHOULD NOT be replicated. This means that the number of failures, of grace logins and the locking will take place on each replicated server. For example, the effective number of failed attempts on a user password will be N x M (where N is the number of servers and M the value of pwdMaxFailure attribute). Replicating these attributes to a read-only replica MAY reduce the number of tries globally but MAY also introduce some inconstancies in the way the password policy is applied.
10. Security Considerations

This document defines a set of rules to implement in an LDAP server, in order to improve the security of LDAP directories and make it difficult for password cracking programs to break into directories.

Access controls SHOULD be used to restrict access to the password policy attributes. Especially all the attributes defined to maintain the Password Policy state information SHOULD not be modifiable by anyone but the Administrator of the directory server.

As it is possible to define a password policy for one specific user by adding an subentry immediately under the user’s entry, Access Controls SHOULD be used to restrict the use of the pwdPolicy object class or the LDAP subentry object class.

When a password policy is put in place, the LDAP directory is subject to a denial of service attack. A malicious user could deliberately lock out one specific user’s account (or all of them) by sending bind requests with wrong passwords. There is no way to protect against this kind of attack. The LDAP directory server SHOULD log as much information as it can (such as client IP address) whenever an account is locked, in order to be able to identify the origin of the attack. Denying anonymous access to the LDAP directory is also a way to restrict this kind of attacks.

11. References


12. Copyright Notice

Copyright (C) The Internet Society (2001). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

13. Authors’ Addresses

Behera, et. al. Expires January 18, 2002 Page 34

INTERNET DRAFT LDAP Password Policy 18 July 2001

Prasanta Behera
Netscape Communications Corp.
4170 Network circle.
Santa Clara, CA 95054
USA
+1 408 276-3960
prasanta@netscape.com

Valerie Chu