

Network Working Group	J. Sermersheim	
Internet-Draft	Novell, Inc	
Intended status: Standards Track	L. Poitou	
Expires: February 10, 2010	Sun Microsystems	
	H. Chu, Ed.	
	Symas Corp.	
	August 09, 2009	

[TOC](#)

Password Policy for LDAP Directories **draft-behera-ldap-password-policy-10.txt**

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

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

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

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

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

This Internet-Draft will expire on February 10, 2010.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (<http://trustee.ietf.org/license-info>). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

Password policy as described in this document is a set of rules that controls how passwords are used and administered in Lightweight Directory Access Protocol (LDAP) based 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 to deter password guessing attacks.

Table of Contents

- [1. Overview](#)
- [2. Conventions](#)
- [3. Application of Password Policy](#)
- [4. Articles of Password Policy](#)
 - [4.1. Password Usage Policy](#)
 - [4.2. Password Modification Policy](#)
 - [4.3. Restriction of the Password Policy](#)
- [5. Schema used for Password Policy](#)
 - [5.1. The pwdPolicy Object Class](#)
 - [5.2. Attribute Types used in the pwdPolicy ObjectClass](#)
 - [5.3. Attribute Types for Password Policy State Information](#)
- [6. Controls used for Password Policy](#)
 - [6.1. Request Control](#)
 - [6.2. Response Control](#)
- [7. Policy Decision Points](#)
 - [7.1. Locked Account Check](#)
 - [7.2. Password Must be Changed Now Check](#)
 - [7.3. Password Expiration Check](#)
 - [7.4. Remaining Grace AuthN Check](#)
 - [7.5. Time Before Expiration Check](#)
 - [7.6. Intruder Lockout Check](#)
 - [7.7. Intruder Delay Check](#)
 - [7.8. Password Too Young Check](#)
- [8. Server Policy Enforcement Points](#)
 - [8.1. Password-based Authentication](#)
 - [8.2. Password Update Operations](#)
 - [8.3. Other Operations](#)
- [9. Client Policy Enforcement Points](#)
 - [9.1. Bind Operation](#)
 - [9.2. Modify Operations](#)
 - [9.3. Add Operation](#)
 - [9.4. Compare Operation](#)
 - [9.5. Other Operations](#)
- [10. Administration of the Password Policy](#)
- [11. Password Policy and Replication](#)
- [12. Security Considerations](#)
- [13. IANA Considerations](#)
- [14. Acknowledgement](#)

[15.](#) Normative References

[§](#) Authors' Addresses

1. Overview

[TOC](#)

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. Conventions

[TOC](#)

Imperative keywords defined in [\[RFC2119\]](#) (Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.) are used in this document, and carry the meanings described there.

All ASN.1 [\[X.680\]](#) (International Telecommunications Union, "Abstract Syntax Notation One (ASN.1): Specification of basic notation," July 2002.) Basic Encoding Rules (BER) [\[X.690\]](#) (International Telecommunications Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)," July 2002.) encodings follow the conventions found in Section 5.1 of [\[RFC4511\]](#) (Sermersheim, J., "Lightweight Directory Access Protocol (LDAP): The Protocol," June 2006.).

The term "password administrator" refers to a user that has sufficient access control privileges to modify users' passwords. The term "password policy administrator" refers to a user that has sufficient

access control privileges to modify the pwdPolicy object defined in this document. The access control that is used to determine whether an identity is a password administrator or password policy administrator is beyond the scope of this document, but typically implies that the password administrator has 'write' privileges to the password attribute.

3. Application of Password Policy

[TOC](#)

The password policy defined in this document can be applied to any attribute holding a user's password used for an authenticated LDAP bind operation. In this document, the term "user" represents any LDAP client application that has an identity in the directory.

This policy is typically applied to the userPassword attribute in the case of the LDAP simple authentication method [\[RFC4511\] \(Sermersheim, J., "Lightweight Directory Access Protocol \(LDAP\): The Protocol," June 2006.\)](#) or the case of password based SASL [\[RFC4422\] \(Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer \(SASL\)," June 2006.\)](#) authentication such as CRAM-MD5 [\[RFC2195\] \(Klensin, J., Catoe, R., and P. Krumviede, "IMAP/POP AUTHorize Extension for Simple Challenge/Response," September 1997.\)](#) and DIGEST-MD5 [\[RFC2831\] \(Leach, P. and C. Newman, "Using Digest Authentication as a SASL Mechanism," May 2000.\)](#).

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 multi-valued password attributes.

Server implementations MAY institute internal policy whereby certain 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.

4. Articles of Password Policy

[TOC](#)

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 [Section 8 \(Server Policy Enforcement Points\)](#) and [Section 9 \(Client Policy Enforcement Points\)](#).

[TOC](#)

4.1. Password Usage Policy

This section describes policy enforced when a password is used to authenticate. The general focus of this policy is to minimize the threat of intruders once a password is in use.

4.1.1. Password Validity Policy

[TOC](#)

These mechanisms allow account usage to be controlled independent of any password expiration policies. The policy defines the absolute period of time for which an account may be used. This allows an administrator to define an absolute starting time after which a password becomes valid, and an absolute ending time after which the password is disabled.

A mechanism is also provided to define the period of time for which an account may remain unused before being disabled.

4.1.2. Password Guessing Limit

[TOC](#)

In order to prevent intruders from guessing a user's password, a mechanism exists to track the number of consecutive failed authentication attempts, and take action when a limit is reached. This policy consists of several parts:

- *A counter to track the number of failed authentication attempts.
- *The amount of time to delay on the first authentication failure.
- *The maximum amount of time to delay on subsequent failures.
- *A timeframe in which the limit of consecutive failed authentication attempts must happen before action is taken.
- *A configurable limit on failed authentication attempts.
- *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.

Note that using the account lock feature provides an easy avenue for Denial-of-Service (DoS) attacks on user accounts. While some sites' policies require accounts to be locked, this feature is discouraged in favor of delaying each failed login attempt.

The delay time will be doubled on each subsequent failure, until it reaches the maximum time configured.

[TBD: we could also provide a syntax for configuring a backoff algorithm. E.g. "+<int>" for linearly incrementing delay, "x<int>" for constant multiplier, "^<int>" for geometric. But it's probably overkill to add a calculator language to the server.]

4.2. Password Modification Policy

[TOC](#)

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. In this document, the term "modify password operation" refers to any operation that is used to add or modify a password attribute. Often this is done by updating the password attribute during an add or modify operation, but MAY be done by other means such as an extended operation.

4.2.1. Password Expiration, Expiration Warning, and Grace Authentications

[TOC](#)

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.

Password policy 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:

- *A warning may be returned to the user 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' authentications, her account will be locked.

[TOC](#)

4.2.2. Password History

When the Password Expiration policy is used, an additional mechanism may be employed 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 password policy 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.

4.2.3. Password Minimum Age

[TOC](#)

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.

4.2.4. Password Quality and Minimum length

[TOC](#)

In order to prevent users from creating or updating passwords that are easy to guess, a password quality policy may be employed. This policy consists of two general mechanisms - ensuring that passwords conform to a defined quality criterion and ensuring that they are of a minimum length.

Forcing a password to comply with the quality policy 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 'quality checking' means. This can lead to unexpected behavior in a heterogeneous environment.

4.2.5. User Defined Passwords

[TOC](#)

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

4.2.6. Password Change after Reset

[TOC](#)

This policy forces the user to update her password after it has been set for the first time, or has been reset by a password administrator. This is needed in scenarios where a password administrator has set or reset the password to a well-known value.

4.2.7. Safe Modification

[TOC](#)

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 during a password modify operation.

{TODO: This allows a dictionary attack unless we specify that this is also subject to intruder detection. One solution is to require users to authN prior to changing password. Another solution is to perform intruder detection checks when the password for a non-authenticated identity is being updated}

4.3. Restriction of the Password Policy

[TOC](#)

The password policy defined in this document can apply to any attribute containing a password. Password policy state information is held in the user's entry, and applies to a password attribute, not a particular password attribute value. Thus the server SHOULD enforce that the

password attribute subject to password policy, contains one and only one password value.

5. Schema used for Password Policy

[TOC](#)

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 that hold general password policy state information for each user.

5.1. The pwdPolicy Object Class

[TOC](#)

This object class contains the attributes defining a password policy in effect for a set of users. [Section 10 \(Administration of the Password Policy\)](#) describes the administration of this object, and the relationship between it and particular objects.

```
( 1.3.6.1.4.1.42.2.27.8.2.1
NAME 'pwdPolicy'
SUP top
AUXILIARY
MUST ( pwdAttribute )
MAY ( pwdMinAge $ pwdMaxAge $ pwdInHistory $ pwdCheckQuality $
pwdMinLength $ pwdMaxLength $ pwdExpireWarning $
pwdGraceAuthNLimit $ pwdGraceExpiry $ pwdLockout $
pwdLockoutDuration $ pwdMaxFailure $ pwdFailureCountInterval $
pwdMustChange $ pwdAllowUserChange $ pwdSafeModify $
pwdMinDelay $ pwdMaxDelay $ pwdMaxIdle ) )
```

5.2. Attribute Types used in the pwdPolicy ObjectClass

[TOC](#)

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

[TOC](#)

5.2.1. 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 objectIdentifierMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
```

5.2.2. pwdMinAge

[TOC](#)

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 )
```

5.2.3. pwdMaxAge

[TOC](#)

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. If not 0, the value must be greater than or equal to the value of the pwdMinAge.

```
( 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 )
```

[TOC](#)

5.2.4. 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 )
```

5.2.5. pwdCheckQuality

[TOC](#)

{TODO: Consider changing the syntax to OID. Each OID will list a quality rule (like min len, # of special characters, etc). These rules can be specified outside this document.}

{TODO: Note that even though this is meant to be a check that happens during password modification, it may also be allowed to happen during authN. This is useful for situations where the password is encrypted when modified, but decrypted when used to authN.}

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

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

5.2.6. pwdMinLength

[TOC](#)

When quality checking is enabled, this attribute holds the minimum number of characters that must be used in a password. 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 `pwdCheckQuality` 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 )
```

5.2.7. `pwdMaxLength`

[TOC](#)

When quality checking is enabled, this attribute holds the maximum number of characters that may be used in a password. If this attribute is not present, no maximum 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 `pwdCheckQuality` attribute, either accept the password without checking it ('0' or '1') or refuse it ('2').

```
( 1.3.6.1.4.1.42.2.27.8.1.31
  NAME 'pwdMaxLength'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )
```

5.2.8. `pwdExpireWarning`

[TOC](#)

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 returned. If not 0, the value must be smaller than the value of the `pwdMaxAge` attribute.

```
( 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 )
```

5.2.9. pwdGraceAuthNLimit

[TOC](#)

This attribute specifies the number of times an expired password can be used to authenticate. If this attribute is not present or if the value is 0, authentication will fail.

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

5.2.10. pwdGraceExpiry

[TOC](#)

This attribute specifies the number of seconds the grace authentications are valid. If this attribute is not present or if the value is 0, there is no time limit on the grace authentications.

```
( 1.3.6.1.4.1.42.2.27.8.1.30
NAME 'pwdGraceExpire'
EQUALITY integerMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
SINGLE-VALUE )
```

5.2.11. pwdLockout

[TOC](#)

This attribute indicates, when its value is "TRUE", that the password may not be used to authenticate 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 password may be used to authenticate 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 )
```

5.2.12. pwdLockoutDuration

[TOC](#)

This attribute holds the number of seconds that the password cannot be used to authenticate due to too many failed bind attempts. If this attribute is not present, or if the value is 0 the password cannot be used to authenticate until reset by a password 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 )
```

5.2.13. pwdMaxFailure

[TOC](#)

This attribute specifies the number of consecutive failed bind attempts after which the password may not be used to authenticate. If this attribute is not present, or if the value is 0, this policy is not checked, and the value of pwdLockout will be ignored.

```
( 1.3.6.1.4.1.42.2.27.8.1.11
NAME 'pwdMaxFailure'
EQUALITY integerMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
SINGLE-VALUE )
```

5.2.14. pwdFailureCountInterval

[TOC](#)

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.

```
( 1.3.6.1.4.1.42.2.27.8.1.12
NAME 'pwdFailureCountInterval'
EQUALITY integerMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
SINGLE-VALUE )
```

5.2.15. pwdMustChange

[TOC](#)

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 a password 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 password administrator sets or resets the password. This attribute is not set due to any actions specified by this document, it is typically set by a password administrator after resetting a user's password.

```
( 1.3.6.1.4.1.42.2.27.8.1.13
NAME 'pwdMustChange'
EQUALITY booleanMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
SINGLE-VALUE )
```

5.2.16. pwdAllowUserChange

[TOC](#)

This attribute indicates whether users can change their own passwords, although the change operation is still subject to access control. If this attribute is not present, a value of "TRUE" is assumed. This attribute is intended to be used in the absence of an access control mechanism.

```
( 1.3.6.1.4.1.42.2.27.8.1.14
NAME 'pwdAllowUserChange'
EQUALITY booleanMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
SINGLE-VALUE )
```

5.2.17. pwdSafeModify

[TOC](#)

This attribute specifies whether or not the existing password must be sent along with the new password when being changed. If this attribute is not present, a "FALSE" value is assumed.

```
( 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 )
```

5.2.18. pwdMinDelay

[TOC](#)

This attribute specifies the number of seconds to delay responding to the first failed authentication attempt. If this attribute is not set or is 0, no delays will be used. pwdMaxDelay must also be specified if pwdMinDelay is set.

```
( 1.3.6.1.4.1.42.2.27.8.1.24
  NAME 'pwdMinDelay'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )
```

5.2.19. pwdMaxDelay

[TOC](#)

This attribute specifies the maximum number of seconds to delay when responding to a failed authentication attempt. The time specified in pwdMinDelay is used as the starting time and is then doubled on each failure until the delay time is greater than or equal to pwdMaxDelay (or a successful authentication occurs, which resets the failure counter). pwdMinDelay must be specified if pwdMaxDelay is set.

```
( 1.3.6.1.4.1.42.2.27.8.1.25
  NAME 'pwdMaxDelay'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )
```

5.2.20. pwdMaxIdle

[TOC](#)

This attribute specifies the number of seconds an account may remain unused before it becomes locked. If this attribute is not set or is 0, no check is performed.

```
( 1.3.6.1.4.1.42.2.27.8.1.26
  NAME 'pwdMaxIdle'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )
```

5.3. Attribute Types for Password Policy State Information

[TOC](#)

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, pwdFailureTime, pwdHistory, pwdGraceUseTime, pwdReset, pwdPolicySubEntry, pwdStartTime, pwdEndTime, pwdLastSuccess.

5.3.1. Password Policy State Attribute Option

[TOC](#)

Since the password policy could apply to several attributes used to store passwords, each of the above operational attributes must have an option to specify which pwdAttribute it applies to. The password policy option is defined as the following:

pwd-<passwordAttribute>

where passwordAttribute a string following the OID syntax (1.3.6.1.4.1.1466.115.121.1.38). The attribute type descriptor (short name) MUST be used.

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

pwdChangedTime;pwd-userPassword: 20000103121520Z

This attribute option follows sub-typing semantics. If a client requests a password policy state attribute to be returned in a search operation, and does not specify an option, all subtypes of that policy state attribute are returned.

[TOC](#)

5.3.2. pwdChangedTime

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

```
( 1.3.6.1.4.1.42.2.27.8.1.16
  NAME 'pwdChangedTime'
  DESC 'The time the password was last changed'
  EQUALITY generalizedTimeMatch
  ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
  SINGLE-VALUE
  NO-USER-MODIFICATION
  USAGE directoryOperation )
```

5.3.3. pwdAccountLockedTime

[TOC](#)

This attribute holds the time that the user's account was locked. A locked account means that the password may no longer be used to authenticate. A 000001010000Z value means that the account has been locked permanently, and that only a password 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'
  EQUALITY generalizedTimeMatch
  ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
  SINGLE-VALUE
  NO-USER-MODIFICATION
  USAGE directoryOperation )
```

5.3.4. pwdFailureTime

[TOC](#)

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'
EQUALITY generalizedTimeMatch
ORDERING generalizedTimeOrderingMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
NO-USER-MODIFICATION
USAGE directoryOperation )
```

5.3.5. pwdHistory

[TOC](#)

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          = GeneralizedTime

syntaxOID     = numericoid      ; the string representation of the
                                ; dotted-decimal OID that defines the
                                ; syntax used to store the password.

length        = number          ; the number of octets in data.

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

GeneralizedTime is specified in 3.3.13 of [\[RFC4517\] \(Legg, S., "Lightweight Directory Access Protocol \(LDAP\): Syntaxes and Matching Rules," June 2006.\)](#). numericoid and number are specified in 1.4 of [\[RFC4512\] \(Zeilenga, K., "Lightweight Directory Access Protocol \(LDAP\): Directory Information Models," June 2006.\)](#).

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'
EQUALITY octetStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
NO-USER-MODIFICATION
USAGE directoryOperation )
```

5.3.6. pwdGraceUseTime

[TOC](#)

This attribute holds the timestamps of grace authentications after a password has expired.

```
( 1.3.6.1.4.1.42.2.27.8.1.21
NAME 'pwdGraceUseTime'
DESC 'The timestamps of the grace authentication after the
password has expired'
EQUALITY generalizedTimeMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
NO-USER-MODIFICATION
USAGE directoryOperation )
```

5.3.7. pwdReset

[TOC](#)

This attribute holds a flag to indicate (when TRUE) that the password has been updated by the password administrator and must be changed by the user.

```
( 1.3.6.1.4.1.42.2.27.8.1.22
NAME 'pwdReset'
DESC 'The indication that the password has been reset'
EQUALITY booleanMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
SINGLE-VALUE
USAGE directoryOperation )
```

[TOC](#)

5.3.8. pwdPolicySubentry

This attribute points to the pwdPolicy subentry in effect for this object.

```
( 1.3.6.1.4.1.42.2.27.8.1.23
NAME 'pwdPolicySubentry'
DESC 'The pwdPolicy subentry in effect for this object'
EQUALITY distinguishedNameMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
SINGLE-VALUE
NO-USER-MODIFICATION
USAGE directoryOperation )
```

5.3.9. pwdStartTime

[TOC](#)

This attribute specifies the time the entry's password becomes valid for authentication. Authentication attempts made before this time will fail. If this attribute does not exist, then no restriction applies.

```
( 1.3.6.1.4.1.42.2.27.8.1.27
NAME 'pwdStartTime'
DESC 'The time the password becomes enabled'
EQUALITY generalizedTimeMatch
ORDERING generalizedTimeOrderingMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
SINGLE-VALUE
NO-USER-MODIFICATION
USAGE directoryOperation )
```

5.3.10. pwdEndTime

[TOC](#)

This attribute specifies the time the entry's password becomes invalid for authentication. Authentication attempts made after this time will fail, regardless of expiration or grace settings. If this attribute does not exist, then this restriction does not apply.

```
( 1.3.6.1.4.1.42.2.27.8.1.28
NAME 'pwdEndTime'
DESC 'The time the password becomes disabled'
EQUALITY generalizedTimeMatch
ORDERING generalizedTimeOrderingMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
SINGLE-VALUE
NO-USER-MODIFICATION
USAGE directoryOperation )
```

Note that pwdStartTime may be set to a time greater than or equal to pwdEndTime; this simply disables the account.

5.3.11. pwdLastSuccess

[TOC](#)

This attribute holds the timestamp of the last successful authentication.

```
( 1.3.6.1.4.1.42.2.27.8.1.29
NAME 'pwdLastSuccess'
DESC 'The timestamp of the last successful authentication'
EQUALITY generalizedTimeMatch
ORDERING generalizedTimeOrderingMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
SINGLE-VALUE
NO-USER-MODIFICATION
USAGE directoryOperation )
```

6. Controls used for Password Policy

[TOC](#)

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. {TODO: add a note about advertisement and discovery}

6.1. Request Control

[TOC](#)

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 may be TRUE or FALSE. There is no controlValue.

6.2. Response Control

[TOC](#)

If the client has sent a passwordPolicyRequest control, the server (when solicited by the inclusion of the request control) 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). The controlType is 1.3.6.1.4.1.42.2.27.8.5.1 and the controlValue is the BER encoding of the following type:

```
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 }
```

The timeBeforeExpiration warning specifies the number of seconds before a password will expire. The graceAuthNsRemaining warning specifies the remaining number of times 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 error is set when a user is restricted from changing her password. The insufficientPasswordQuality error is set when a password doesn't pass quality checking. The passwordTooYoung error 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 password 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.

7. Policy Decision Points

[TOC](#)

Following are a number of procedures used to make policy decisions. These procedures are typically performed by the server while processing an 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.

7.1. Locked Account Check

[TOC](#)

A status of `true` is returned to indicate that the account is locked if any of these conditions are met:

- *The value of the `pwdAccountLockedTime` attribute is `000001010000Z`.

- *The current time is less than the value of the `pwdStartTime` attribute.

- *The current time is greater than or equal to the value of the `pwdEndTime` attribute.

- *The current time is greater than or equal to the value of the `pwdLastSuccess` attribute added to the value of the `pwdMaxIdle` attribute.

- *The current time is less than the value of the `pwdAccountLockedTime` attribute added to the value of the `pwdLockoutDuration`.

Otherwise a status of `false` is returned.

[TOC](#)

7.2. Password Must be Changed Now Check

A status of true is returned to indicate that the password must be changed if all of these conditions are met:

- *The pwdMustChange attribute is set to TRUE.

- *The pwdReset attribute is set to TRUE.

Otherwise a status of false is returned.

7.3. Password Expiration Check

[TOC](#)

A status of true is returned indicating that the password has expired if the current time minus the value of pwdChangedTime is greater than the value of the pwdMaxAge.

Otherwise, a status of false is returned.

7.4. Remaining Grace AuthN Check

[TOC](#)

If the pwdGraceUseTime attribute is present, the number of values in that attribute subtracted from the value of pwdGraceAuthNLimit is returned. Otherwise zero is returned. A positive result specifies the number of remaining grace authentications.

7.5. Time Before Expiration Check

[TOC](#)

If the pwdExpireWarning attribute is not present a zero status is returned. Otherwise the following steps are followed:

Subtract the time stored in pwdChangedTime from the current time to arrive at the password's age. If the password's age is greater than the value of the pwdMaxAge attribute, a zero status is returned. Subtract the value of the pwdExpireWarning attribute from the value of the pwdMaxAge attribute to arrive at the warning age. If the password's age is equal to or greater than the warning age, the value of pwdMaxAge minus the password's age is returned.

[TOC](#)

7.6. Intruder Lockout Check

A status of true indicating that an intruder has been detected is returned if the following conditions are met:

- *The pwdLockout attribute is TRUE.

- *The number of values in the pwdFailureTime attribute that are younger than pwdFailureCountInterval is greater or equal to the pwdMaxFailure attribute.

Otherwise a status of false is returned.

While performing this check, values of pwdFailureTime that are old by more than pwdFailureCountInterval are purged and not counted.

7.7. Intruder Delay Check

[TOC](#)

If the pwdMinDelay attribute is 0 or not set, zero is returned.

Otherwise, a delay time is computed based on the number of values in the pwdFailureTime attribute. If the computed value is greater than the pwdMaxDelay attribute, the pwdMaxDelay value is returned.

While performing this check, values of pwdFailureTime that are old by more than pwdFailureCountInterval are purged and not counted.

7.8. Password Too Young Check

[TOC](#)

If the [Section 7.2 \(Password Must be Changed Now Check\)](#) check returned true then this check will return false, to allow the password to be changed.

A status of true indicating that not enough time has passed since the password was last updated is returned if:

- *The value of pwdMinAge is non-zero and pwdChangedTime is present.

- *The value of pwdMinAge is greater than the current time minus the value of pwdChangedTime.

Otherwise a false status is returned.

[TOC](#)

8. Server Policy Enforcement Points

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.

Note: The case where a single password value is stored in multiple formats simultaneously is still considered to be 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 passwordPolicyResponse control is returned. All other instructions remain the same.

For successfully completed operations, unless otherwise stated, no passwordPolicyResponse control is returned.

8.1. Password-based Authentication

[TOC](#)

This section contains the policy enforcement rules and policy data updates used while validating a password. Operations that validate passwords include, but are not limited to, the Bind operation where the simple choice specifies a password, and the Compare operation where the attribute being compared holds a password. Note that while the Compare operation does not authenticate a user to the LDAP server, it may be used by an external application for purposes of authentication.

8.1.1. Fail if the account is locked

[TOC](#)

If the account is locked as specified in [Section 7.1 \(Locked Account Check\)](#), the server fails the operation with an appropriate resultCode (i.e. invalidCredentials (49) in the case of a bind operation, compareFalse (5) in the case of a compare operation, etc.). The server MAY set the error: accountLocked (1) in the passwordPolicyResponse in the controls field of the message.

8.1.2. Validated Password Procedures

[TOC](#)

If the validation operation indicates that the password validated, these procedures are followed in order:

8.1.2.1. Policy state updates

[TOC](#)

Delete the `pwdFailureTime` and `pwdAccountLockedTime` attributes.

Set the value of the `pwdLastSuccess` attribute to the current time.

Note: setting `pwdLastSuccess` is optional, but it is required if the policy has `pwdMaxIdle` defined.

8.1.2.2. Password must be changed now

[TOC](#)

If the decision in [Section 7.2 \(Password Must be Changed Now Check\)](#) returns true, the server sends to the client a response with an appropriate successful `resultCode` (i.e. `success (0)`, `compareTrue (6)`, etc.), and includes the `passwordPolicyResponse` in the `controls` field of the `bindResponse` message with the `warning: changeAfterReset` specified. For `bind`, the server MUST then disallow all operations issued by this user except `modify password`, `bind`, `unbind`, `abandon` and `StartTLS` extended operation.

8.1.2.3. Expired password

[TOC](#)

If the password has expired as per [Section 7.3 \(Password Expiration Check\)](#), the server either returns a success or failure based on the state of grace authentications.

8.1.2.3.1. Remaining Grace Authentications

[TOC](#)

If there are remaining grace authentications as per [Section 7.4 \(Remaining Grace AuthN Check\)](#), the server adds a new value with the current time in `pwdGraceUseTime`. Then it sends to the client a response with an appropriate successful `resultCode` (i.e. `success (0)`, `compareTrue (6)`, etc.), and includes the `passwordPolicyResponse` in the `controls` field of the response message with the `warning: graceAuthNsRemaining` choice set to the number of grace authentications left.

Implementor's note: The system time of the host machine may be more granular than is needed to ensure unique values of this attribute. It is recommended that a mechanism is used to ensure unique generalized time values. The `fractional seconds` field may be used for this purpose.

8.1.2.3.2. No Remaining Grace Authentications

[TOC](#)

If there are no remaining grace authentications, the server fails the operation with an appropriate resultCode (invalidCredentials (49), compareFalse (5), etc.), and includes the passwordPolicyResponse in the controls field of the bindResponse message with the error: passwordExpired (0) set.

8.1.2.4. Expiration Warning

[TOC](#)

If the result of [Section 7.5 \(Time Before Expiration Check\)](#) is a positive number, the server sends to the client a response with an appropriate successful resultCode (i.e. success (0), compareTrue (6), etc.), and includes the passwordPolicyResponse in the controls field of the bindResponse message with the warning: timeBeforeExpiration set to the value as described above. Otherwise, the server sends a successful response, and omits the passwordPolicyResponse.

8.1.3. AuthN Failed Procedures

[TOC](#)

If the authentication process indicates that the password failed validation due to invalid credentials, these procedures are followed:

8.1.3.1. Policy state update

[TOC](#)

Add the current time as a value of the pwdFailureTime attribute. Implementor's note: The system time of the host machine may be more granular than is needed to ensure unique values of this attribute. It is recommended that a mechanism is used to ensure unique generalized time values. The fractional seconds field may be used for this purpose.

8.1.3.2. Handle Intruder Detection

[TOC](#)

If the check in [Section 7.6 \(Intruder Lockout Check\)](#) returns a true state, the server locks the account by setting the value of the pwdAccountLockedTime attribute to the current time. After locking the account, the server fails the operation with an appropriate resultCode (invalidCredentials (49), compareFalse (5), etc.), and includes the

passwordPolicyResponse in the controls field of the message with the error: accountLocked (1).

If the check in [Section 7.7 \(Intruder Delay Check\)](#) returns a non-zero value, the server waits that number of seconds before sending the authentication response back to the client.

8.2. Password Update Operations

[TOC](#)

Because the password is stored in an attribute, various operations (like add and modify) 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 [\[RFC3062\] \(Zeilenga, K., "LDAP Password Modify Extended Operation," February 2001.\)](#).

While processing a password update, the server performs the following steps:

8.2.1. Safe Modification

[TOC](#)

If pwdSafeModify is set to TRUE and if there is an existing password value, the server ensures that the password update operation includes the user's existing password.

When the LDAP modify operation is used to modify a password, this is done by specifying both a delete action and an add or replace action, where the delete action specifies the existing password, and the add or replace action specifies the new password. Other password update operations SHOULD employ a similar mechanism. Otherwise this policy will fail.

If the existing password is not specified, the server does not process the operation and sends the appropriate response message to the client with the resultCode: insufficientAccessRights (50), and includes the passwordPolicyResponse in the controls field of the response message with the error: mustSupplyOldPassword (4).

8.2.2. Change After Reset

[TOC](#)

If the decision in [Section 7.2 \(Password Must be Changed Now Check\)](#) returns true, the server ensures that the password update operation contains no modifications other than the modification of the password attribute. If other modifications exist, the server sends a response message to the client with the resultCode: insufficientAccessRights

(50), and includes the passwordPolicyResponse in the controls field of the response message with the error: changeAfterReset (2).

8.2.3. Rights Check

[TOC](#)

Check to see whether the bound identity has sufficient rights to update the password. If the bound identity is a user changing its own password, this MAY be done by checking the pwdAllowUserChange attribute or using an access control mechanism. The determination of this is implementation specific. If the user is not allowed to update her password, the server sends a response message to the client with the resultCode: insufficientAccessRights (50), and includes the passwordPolicyResponse in the controls field of the response message with the error: passwordModNotAllowed (3).

8.2.4. Too Early to Update

[TOC](#)

If the check in [Section 7.8 \(Password Too Young Check\)](#) results in a true status The server sends a response message to the client with the resultCode: constraintViolation (19), and includes the passwordPolicyResponse in the controls field of the response message with the error: passwordTooYoung (7).

8.2.5. Password Quality

[TOC](#)

Check the value of the pwdCheckQuality attribute. If the value is non-zero, the server:

*Ensure that the password meets the quality criteria enforced by the server. This enforcement is implementation specific. If the server is unable to check the quality (due to a hashed password or otherwise), the value of pwdCheckQuality is evaluated. If the value is 1, operation continues. If the value is 2, the server sends a response message to the client with the resultCode: constraintViolation (19), and includes the passwordPolicyResponse in the controls field of the response message with the error: insufficientPasswordQuality (5). If the server is able to check the password quality, and the check fails, the server sends a response message to the client with the resultCode: constraintViolation (19), and includes the passwordPolicyResponse

in the controls field of the response message with the error: insufficientPasswordQuality (5).

*checks the value of the pwdMinLength attribute. If the value is non-zero, it ensures 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 pwdCheckQuality is evaluated. If the value is 1, operation continues. If the value is 2, the server sends a response message to the client with the resultCode: constraintViolation (19), and includes the passwordPolicyResponse in the controls field of the response message with the error: passwordTooShort (6). If the server is able to check the password length, and the check fails, the server sends a response message to the client with the resultCode: constraintViolation (19), and includes the passwordPolicyResponse in the controls field of the response message with the error: passwordTooShort (6).

8.2.6. Invalid Reuse

[TOC](#)

If pwdInHistory is present and its value is non-zero, the server checks 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 sends a response message to the client with the resultCode: constraintViolation (19), and includes the passwordPolicyResponse in the controls field of the response message with the error: passwordInHistory (8).

8.2.7. Policy State Updates

[TOC](#)

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

If the value of either pwdMaxAge or pwdMinAge is non-zero, the server updates the pwdChangedTime attribute on the entry to the current time.

If the value of pwdInHistory is non-zero, the server adds the previous password (if one existed) to the pwdHistory attribute. If the number of attributes held in the pwdHistory attribute exceeds the value of pwdInHistory, the server removes the oldest excess passwords.

If the value the pwdMustChange is TRUE and the modification is performed by a password administrator, then the pwdReset attribute is

set to TRUE. Otherwise, the pwdReset is removed from the user's entry if it exists.

The pwdFailureTime and pwdGraceUseTime attributes is removed from the user's entry if they exist.

8.3. Other Operations

[TOC](#)

For operations other than bind, password update, unbind, abandon or StartTLS, if the decision in [Section 7.2 \(Password Must be Changed Now Check\)](#) returns true, the server sends a response message to the client with the resultCode: insufficientAccessRights (50), and includes the passwordPolicyResponse in the controls field of the response message with the error: changeAfterReset (2).

9. Client Policy Enforcement Points

[TOC](#)

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 passwordPolicyResponse control is returned. All other instructions remain the same.

9.1. Bind Operation

[TOC](#)

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

```
*bindResponse.resultCode = insufficientAccessRights (50),
passwordPolicyResponse.error = accountLocked (1): The password
failure limit has been reached and the account is locked. The
user needs to retry later or contact the password administrator
to reset the password.
```

```
*bindResponse.resultCode = success (0),
passwordPolicyResponse.error = changeAfterReset (2): The user is
binding for the first time after the password administrator set
```

the password. In this scenario, the client SHOULD prompt the user to change his password immediately.

```
*bindResponse.resultCode = success (0),  
passwordPolicyResponse.warning = graceAuthNsRemaining: The  
password has expired but there are remaining grace  
authentications. The user needs to change it.
```

```
*bindResponse.resultCode = invalidCredentials (49),  
passwordPolicyResponse.error = passwordExpired (0): The password  
has expired and there are no more grace authentications. The user  
contacts the password administrator in order to have its password  
reset.
```

```
*bindResponse.resultCode = success (0),  
passwordPolicyResponse.warning = timeBeforeExpiration: The user's  
password will expire in n number of seconds.
```

9.2. Modify Operations

[TOC](#)

9.2.1. Modify Request

[TOC](#)

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 `pwdCheckQuality` attributes and SHOULD enforce these policies.

9.2.2. Modify Response

[TOC](#)

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 checks the `resultCode` of the response and checks for a `passwordPolicyResponse` control to determine if any of the following conditions are true and optionally notify the user of the condition.

```
*pwdModResponse.resultCode = insufficientAccessRights (50),  
passwordPolicyResponse.error = mustSupplyOldPassword (4): The
```

user attempted to change her password without specifying the old password but the password policy requires this.

```
*pwdModResponse.resultCode = insufficientAccessRights (50),
passwordPolicyResponse.error = changeAfterReset (2): The user
must change her password before submitting any other LDAP
requests.
```

```
*pwdModResponse.resultCode = insufficientAccessRights (50),
passwordPolicyResponse.error = passwordModNotAllowed (3): The
user doesn't have sufficient rights to change his password.
```

```
*pwdModResponse.resultCode = constraintViolation (19),
passwordPolicyResponse.error = passwordTooYoung (7): It is too
soon after the last password modification to change the password.
```

```
*pwdModResponse.resultCode = constraintViolation (19),
passwordPolicyResponse.error = insufficientPasswordQuality (5):
The password failed quality checking.
```

```
*pwdModResponse.resultCode = constraintViolation (19),
passwordPolicyResponse.error = passwordTooShort (6): The length
of the password is too short.
```

```
*pwdModResponse.resultCode = constraintViolation (19),
passwordPolicyResponse.error = passwordInHistory (8): The
password has already been used; the user must choose a different
one.
```

9.3. Add Operation

[TOC](#)

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

```
*addResponse.resultCode = insufficientAccessRights (50),
passwordPolicyResponse.error = passwordModNotAllowed (3): The
user doesn't have sufficient rights to add this password.
```

```
*addResponse.resultCode = constraintViolation (19),
passwordPolicyResponse.error = insufficientPasswordQuality (5):
The password failed quality checking.
```

```
*addResponse.resultCode = constraintViolation (19),
passwordPolicyResponse.error = passwordTooShort (6): The length
of the password is too short.
```

9.4. Compare Operation

[TOC](#)

When a compare operation is used to compare a password, the client checks the resultCode of the compareResponse and checks 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.

```
*compareResponse.resultCode = compareFalse (5),
passwordPolicyResponse.error = accountLocked (1): The password
failure limit has been reached and the account is locked. The
user needs to retry later or contact the password administrator
to reset the password.
```

```
*compareResponse.resultCode = compareTrue (6),
passwordPolicyResponse.warning = graceAuthNsRemaining: The
password has expired but there are remaining grace
authentications. The user needs to change it.
```

```
*compareResponse.resultCode = compareFalse (5),
passwordPolicyResponse.error = passwordExpired (0): The password
has expired and there are no more grace authentications. The user
must contact the password administrator to reset the password.
```

```
*compareResponse.resultCode = compareTrue (6),
passwordPolicyResponse.warning = timeBeforeExpiration: The user's
password will expire in n number of seconds.
```

9.5. Other Operations

[TOC](#)

For operations other than bind, unbind, abandon or StartTLS, the client checks the result code and control to determine if any other actions are needed.

```
*<Response>.resultCode = insufficientAccessRights (50),
passwordPolicyResponse.error = accountLocked (1) : The password
failure limit has been reached and the account is locked. The
```

user needs to retry later or contact the password administrator to reset the password.

```
*<Response>.resultCode = insufficientAccessRights (50),  
passwordPolicyResponse.error = changeAfterReset (2) : The user  
needs to change the password immediately.
```

10. Administration of the Password Policy

[TOC](#)

{TODO: Need to define an administrativeRole (need OID). Need to describe whether pwdPolicy admin areas can overlap}

A password policy is 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 defined by the SubtreeSpecification attribute of the LDAP subentry as specified in [\[RFC3672\] \(Zeilenga, K., "Subentries in the Lightweight Directory Access Protocol \(LDAP\)," December 2003.\)](#).

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.

Only one policy may be in effect for a given password attribute in any entry. If multiple policies exist which overlap in the range of entries affected, the resulting behavior is undefined.

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.

Each object that is controlled by password policy advertises the subentry that is being used to control its policy in its pwdPolicySubentry attribute. Clients wishing to examine or manage password policy for an object may interrogate the pwdPolicySubentry for that object in order to arrive at the proper pwdPolicy subentry.

11. Password Policy and Replication

[TOC](#)

{TODO: This section needs to be changed to highlight the pitfalls of replication, suggest some implementation choices to overcome those pitfalls, but remove prescriptive language relating to the update of state information}

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`, `pwdFailureTime` and `pwdGraceUseTime` attributes **SHOULD** 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 authentications and the locking will take place on each replicated server. For example, the effective number of failed attempts on a user password will be $N \times M$ (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 inconsistencies in the way the password policy is applied.

Note: there are some situations where global replication of these state attributes may not be desired. For example, if two clusters of replicas are geographically remote and joined by a slow network link, and their users only login from one of the two locations, it may be unnecessary to propagate all of the state changes from one cluster to the other. Servers **SHOULD** allow administrators to control which attributes are replicated on a case-by-case basis.

Servers participating in a loosely consistent multi-master replication agreement **SHOULD** employ a mechanism which ensures uniqueness of values when populating the attributes `pwdFailureTime` and `pwdGraceUseTime`. The method of achieving this is a local matter and may consist of using a single authoritative source for the generation of unique time values, or may consist of the use of the fractional seconds part to hold a replica identifier.

12. Security Considerations

[TOC](#)

This document defines a set of rules to implement in an LDAP server, in order to mitigate some of the security risks associated with the use of passwords and to make it difficult for password cracking programs to break into directories.

Authentication with a password MUST follow the recommendations made in [\[RFC4513\] \(Harrison, R., "Lightweight Directory Access Protocol \(LDAP\): Authentication Methods and Security Mechanisms," June 2006.\)](#).

Modifications of passwords SHOULD only occur when the connection is protected with confidentiality and secure authentication.

Access controls SHOULD be used to restrict access to the password policy attributes. The attributes defined to maintain the password policy state information SHOULD only be modifiable by the password administrator or higher authority. The pwdHistory attribute MUST be subject to the same level of access control as the attribute holding the password.

As it is possible to define a password policy for one specific user by adding a 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 the intruder detection password policy is enforced, 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 attack. Using the login delay instead of the lockout mechanism will also help avoid this denial of service.

Returning certain status codes (such as passwordPolicyResponse.error = accountLocked) allows a denial of service attacker to know that it has successfully denied service to an account. Servers SHOULD implement additional checks which return the same status when it is sensed that some number of failed authentication requests has occurred on a single connection, or from a client address. Server implementors are encouraged to invent other checks similar to this in order to thwart this type of DoS attack.

13. IANA Considerations

[TOC](#)

<<<TBD>>>

14. Acknowledgement

[TOC](#)

This document is based in part on prior work done by Valerie Chu from Netscape Communications Corp, published as draft-vchu-ldap-pwd-policy-00.txt (December 1998). Prasanta Behera participated in early revisions of this document.

15. Normative References

[TOC](#)

[RFC2119]	Bradner, S. , " Key words for use in RFCs to Indicate Requirement Levels ," BCP 14, RFC 2119, March 1997 (TXT , HTML , XML).
[RFC2195]	Klensin, J. , Catoe, R. , and P. Krumviede , " IMAP/POP AUTHorize Extension for Simple Challenge/Response ," RFC 2195, September 1997 (TXT , HTML , XML).
[RFC2831]	Leach, P. and C. Newman , " Using Digest Authentication as a SASL Mechanism ," RFC 2831, May 2000 (TXT).
[RFC3062]	Zeilenga, K. , " LDAP Password Modify Extended Operation ," RFC 3062, February 2001 (TXT).
[RFC3383]	Zeilenga, K. , " Internet Assigned Numbers Authority (IANA) Considerations for the Lightweight Directory Access Protocol (LDAP) ," RFC 3383, September 2002 (TXT).
[RFC3672]	Zeilenga, K. , " Subentries in the Lightweight Directory Access Protocol (LDAP) ," RFC 3672, December 2003 (TXT).
[RFC4422]	Melnikov, A. and K. Zeilenga , " Simple Authentication and Security Layer (SASL) ," RFC 4422, June 2006 (TXT).
[RFC4511]	Sermersheim, J. , " Lightweight Directory Access Protocol (LDAP): The Protocol ," RFC 4511, June 2006 (TXT).
[RFC4512]	Zeilenga, K. , " Lightweight Directory Access Protocol (LDAP): Directory Information Models ," RFC 4512, June 2006 (TXT).
[RFC4513]	Harrison, R. , " Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms ," RFC 4513, June 2006 (TXT).
[RFC4517]	Legg, S. , " Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules ," RFC 4517, June 2006 (TXT).
[X.680]	International Telecommunications Union, "Abstract Syntax Notation One (ASN.1): Specification of basic notation," ITU-T Recommendation X.680, July 2002.
[X.690]	International Telecommunications Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)," ITU-T Recommendation X.690, July 2002.

Authors' Addresses

[TOC](#)

	Jim Sermersheim
	Novell, Inc
	1800 South Novell Place

	Provo, Utah 84606
	US
Phone:	+1 801 861-3088
Email:	jimse@novell.com
	Ludovic Poitou
	Sun Microsystems
	180, Avenue de l'Europe
	Zirst de Montbonnot, Saint Ismier cedex 38334
	FR
Phone:	+33 476 188 212
Email:	ludovic.poitou@sun.com
	Howard Chu (editor)
	Symas Corp.
	18740 Oxnard Street, Suite 313A
	Tarzana, California 91356
	US
Phone:	+1 818 757-7087
Email:	hyc@symas.com