EMu Documentation

Password Management

Document Version 1.2

EMu Version 4.1

Contents

S ECTION 1

Overview

The first step for users using the Windows client to access EMu is to log in to the server. The familiar log in dialogue box allows a user to specify their:

- username
- password
- service to which to connect
- group to use (optional)
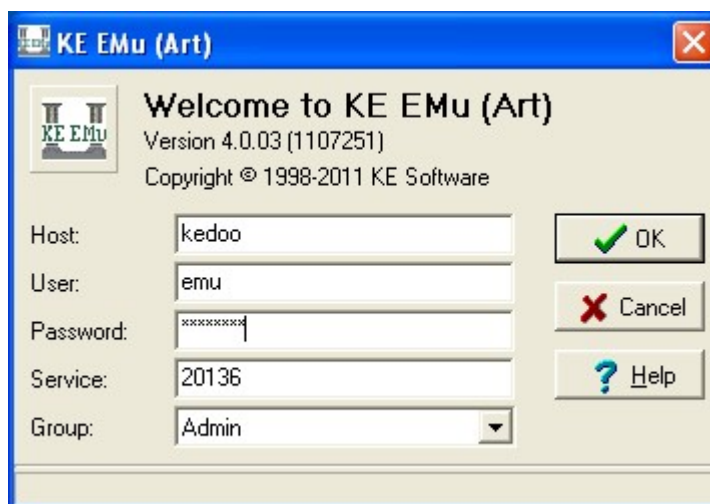
The information entered into the login dialogue box is transmitted to the EMu server for authentication. In particular, the username and password are checked against a database containing user name and password combinations for all persons who are allowed to access the system. EMu itself does not store any user name / password pairs but relies on external databases to authenticate users. These external databases are available through a number of sources:

Source	Description
Unix	The traditional Unix user's database consists of a file containing a list of all users who may access the system. The file is located at `/etc/passwd`. A typical entry looks like: `boris:QB2vbP7yzuNzQ:708:400:Boris Badenov:/home/boris:/bin/bash` Each of the fields in the entry is separated by a colon (:). The list of fields is unimportant for this document, except that the first field contains the user name and the second field the password. The password is stored in a one way encrypted format, that is, the password cannot be decrypted (so if you forget your password, your System Administrator cannot tell you what it is and a new password must be set). When a user logs in to EMu, their password is encrypted and checked against the encrypted version stored with their user name. If there is a match, the user can then access the system.
Shadow	The Shadow password file is an extension of the base UNIX password file described above. The Shadow file is located in `/etc/shadow`. If a Shadow password file is used, the user's password in `/etc/passwd` is replaced with an `x`. The encrypted password is then stored in the Shadow file. The permissions on the Shadow file only allow the System Administrator account (`root`) to read the contents, thus protecting the encrypted password from general access. The Shadow file contains extra fields used to implement password ageing. A typical entry looks like: `boris:QB2vbP7yzuNzQ:15215:1:30::::` Like the `/etc/passwd` file, the fields are separated by a colon (:) and the first two fields store the user name and the associated encrypted password. The third field stores the number of days between 1 January 1970 and the last time the user changed his password. The next two fields contain the minimum number and maximum number of days between password changes respectively. In the example above, user `boris` must change his password at least every thirty days. If the password is not changed within thirty days, it will expire and a new password must be set next time he logs in to EMu. The Shadow password file is available with most versions of Unix, including Solaris and Linux (but not FreeBSD).
NIS	One problem with the Unix and Shadow databases is that they are stored locally. Each machine has its own version of the database, so if a user wants to access more than one machine, an entry needs to exist on each machine for which access is required. A nice solution would be to have a *master* version of the password / Shadow database kept on one machine, and have all other machines contact the master machine when looking up password entries. NIS (Network Information name Service), or YP (Yellow Pages) as it was previously known, provides this functionality. One machine stores the NIS master password / Shadow file and all other machines communicate with the master machine when checking a user name / password combination. NIS can also be used in conjunction with the Unix or Shadow files, providing support for both local users (via Unix and Shadow) and global users (via NIS). NIS not only provides password facilities but may also be used to provide a master version of a wide range of other database files. NIS is available for

Source	Description
	Solaris, Linux and FreeBSD.
	Due to shortcomings in the original NIS design (e.g. password ageing is not possible), a new version of NIS, known as NIS+, was released. Its purpose is the same as NIS except it is more secure and provides extended user attributes (like password ageing).
LDAP	LDAP (Lightweight Directory Access Protocol) is an extension of the NIS idea. One issue with NIS is that it requires a Unix server. It does not provide a general purpose interface that may be used by non-Unix systems. LDAP addresses this problem by implementing a general purpose database (directory) facility that may be used to store any sort of information (including Unix password information). It then defines a system independent way of looking up this information, thus allowing any type of system to retrieve (and possibly update) data. An explanation of how LDAP is structured is beyond the scope of this document. LDAP is available via the OpenLDAP project for Solaris, Linux and FreeBSD. A number of other implementations are also available.
	LDAP allows password and Shadow information to be stored in its database, via the `posixAccount` and `shadowAccount` object classes respectively. When a user logs in to EMu, the LDAP database is consulted to determine whether access should be granted and determine whether the password has expired. As with NIS, the local password / Shadow files may still be used to store local accounts, while LDAP is used for global accounts.
Windows	Windows provides a local database used to contain user password information (amongst other things). The information is for local accounts only and provides authentication for users accessing the local machine. In this sense it is very similar in functionality to the Unix / Shadow databases (but implemented differently). Windows authentication is available on all versions of Windows.
AD	AD (Active Directory) is the Windows implementation of a general purpose information database (directory). It uses LDAP as one of its access protocols. This means that LDAP may be used to consult the Active Directory database. Active Directory allows password / Shadow information to be stored and retrieved via the `posixAccount` and `shadowAccount` object classes respectively. As with LDAP the information stored is for global accounts. AD is provided with Windows Server systems and can be queried by Solaris, Linux, FreeBSD and Windows. The local Windows accounts may still be used to register local users, while AD is used for global accounts.

As you can see, there are a number of alternatives available for registering EMu users and their passwords. In general, each institution will have a policy dictating which of the above sources should be used for user authentication.

EMu implements three mechanisms for determining whether a user name / password combination is correct. The mechanisms are:

Mechanism	Description
PAM	The Pluggable Authentication Module, or PAM for short, is available on all version of Unix, including Solaris, Linux and FreeBSD. It is not available on Windows servers. PAM is a flexible mechanism that uses a configuration file to determine what password sources should be consulted to retrieve password / Shadow information. It provides support for the following database sources: • Unix • Shadow • NIS/NIS+ • LDAP • AD PAM is the most common look-up mechanism used by EMu where the EMu server is installed on a Unix system.
SFU / SUA	Services For Unix (SFU), or Subsystem for UNIX-based Applications (SUA) as it is now known, provides password authentication on Windows based EMu servers. SFU / SUA is not configurable. It provides support for the following sources: • Windows • AD SFU / SUA is the look-up mechanism used by EMu on Windows based EMu servers.
Traditional	If a Unix system does not provide support for PAM, then the Traditional look-up mechanism is used. The Traditional system provides support for the following sources: • Unix • Shadow Traditional support is only provided where institutions elect to not use PAM support. In other words, very rarely!

KE EMu
ELECTRONIC MUSEUM

New Features

Now we have all the theory out of the way, let's look at the new features added to EMu to provide support for password management:

New Feature	Description
Change password	A user may change their password from within the EMu Windows client. The new password is checked to see if it conforms to minimum standards (e.g. length, character mix, etc.) before being set.
Password expiry	A user's password may expire. An expired password occurs when a user has not changed their password within a prescribed number of days. Once the password expires, the user must set a new password the next time they log in to EMu. They will be prompted for the new password.
Expire account	A user's account may expire. Once the account has expired the user will no longer be able to access EMu. There are two ways to expire an account: • The user has not logged in to EMu for a given number of days. • An absolute date after which access to the system will be denied.
Reset password	If a user forgets their password, the System Administrator can clear the old password (or set a new one) and force the user to enter a new password the next time they log in to EMu.
Lock account	The System Administrator may lock a user's account. While the account is locked the user will not be able to access EMu. Once the account is unlocked the user may use EMu once again. Account locking is useful if someone is leaving for an extended period of time, but may return some time in the future.

The implementation of password management in EMu 4.1 provides System Administrators with a range of options in terms of managing user authentication and ageing passwords. It also allows users to change their passwords from within the EMu Windows client.

Section 2

Using Password Management

The majority of the new password management functionality is available in EMu 4.1 without the need to have it enabled. In fact, most of the client side functionality is invoked by requests from the EMu server. For example, if a user's password has expired, the EMu server will inform the EMu Windows client next time the user logs in to EMu. At this time the user will be prompted for a new password.

The only feature that is instigated by the user is the ability to change their password. In order to change a user's password, the EMu server must provide the required support. Both PAM and SFU / SUA provide mechanisms for updating a password, whereas Traditional does not. It is also possible to configure PAM to not support password updates, or to add in other authentication mechanisms (e.g. dongles) that do not provide password updates.

The EMu client cannot determine what level of support is provided by the EMu server for password changing. As such, a new Registry entry has been added to indicate whether password changing is supported. The format of the entry is:

```
User|user|Setting|Change Password|value
Group|group|Setting|Change Password|value
System|Setting|Change Password|value
```

where:

> *value*   is either `true` or `false`.
>
> A `true` value indicates that password changing is supported, while a `false` value removes the Change Password command from the Tools menu in the Windows client. The default value is `true`. Thus, if you are using the Traditional password mechanism, you will need to disable password changing explicitly.

1. Changing your Password

A user may change their password provided the Change Password Registry entry is not set to `false`.

Server support required:

- PAM
- SFU / SUA

In EMu:

1. Open any module.
2. Select **Tools>Change Password...** in the Menu bar
   -OR-
   Use the keyboard shortcut, `ALT+T+H`.
   The Change Password dialogue displays:

3. Enter your existing password and your new password twice.
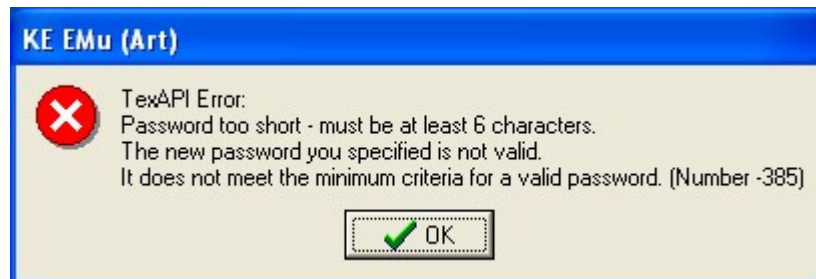   Both instances of the new password must be the same.

4. Click ![OK](OK button).
   The password is updated on the server.
   If the two instances of the new password do not match, an error message displays:

If the new password does not pass the validation criteria, an error message displays:

In both cases the Change Password dialogue displays allowing the error to be corrected.

5. Click [✔ OK] once the password is updated:

2. Updating an Expired Password

A user's password may expire for one of two reasons:

- The System Administrator has expired the password.
- The user has not updated their password within a given time frame. The time frame for password updates is set on the EMu server and can vary from institution to institution.
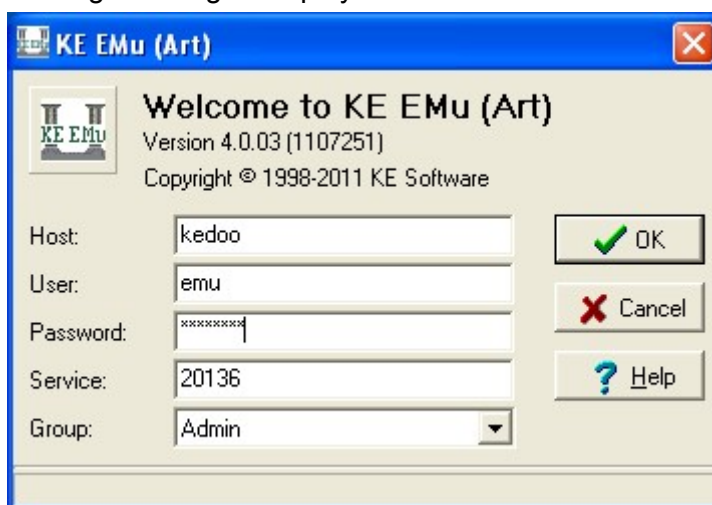
Server support required:
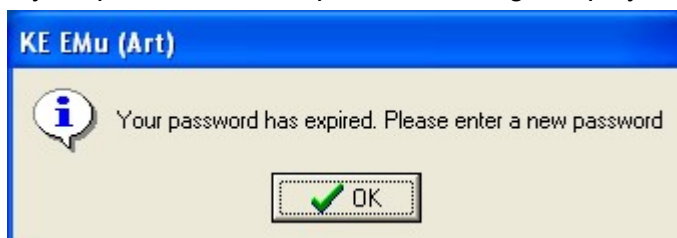
- PAM

In EMu:

1. Start up EMu.

The log in dialogue displays:

6. Enter your log in details and click .

If your password has expired, a message displays:

7. Click .

A new log in dialogue displays:
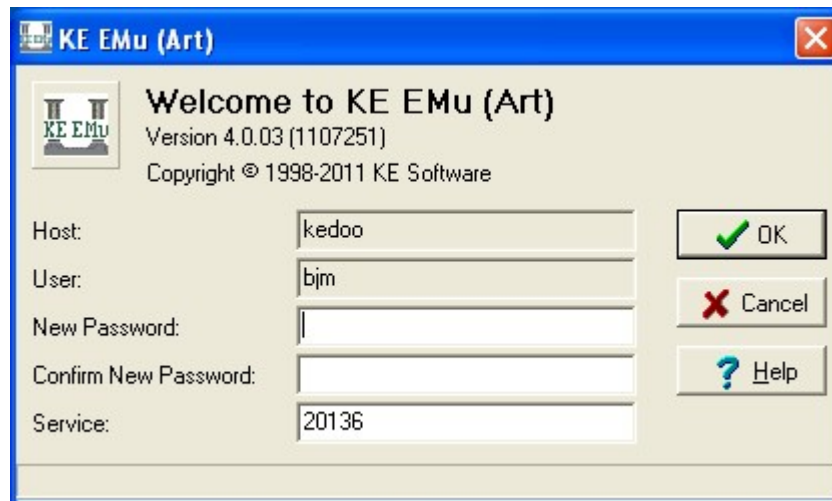
8. Enter a new password and confirm it by re-entering it.

9. Click [✔ OK].

The password is updated on the server.

If the two instances of the new password do not match, an error message displays:

If the new password does not pass the validation criteria, an error message displays:
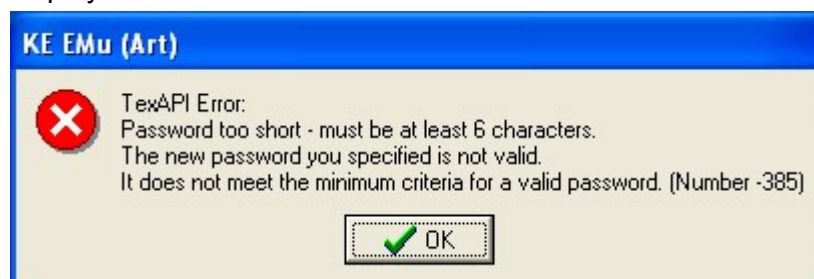
In both cases the log in dialogue is displayed allowing the error to be corrected.

10. Click [✔ OK] once the password is updated:

3. Password Admin Task

A user may also change their password via an Admin task. The task is provided for systems that do not support password changes via PAM or SFU/SUA. The following Registry entry is required to provide the Change Password Admin task:

```
Group|Default|Table|eadmin|Admin Task|Change Password|password
'[Password:Old Password]' '[Password:New Password]'
'[Password:Confirm New Password]'
```

Once the Registry entry is specified, the task becomes available.

Server support required:

- PAM
- Traditional

In EMu:

1.   Select Admin from the Command Centre.
The Admin Tasks dialogue displays.
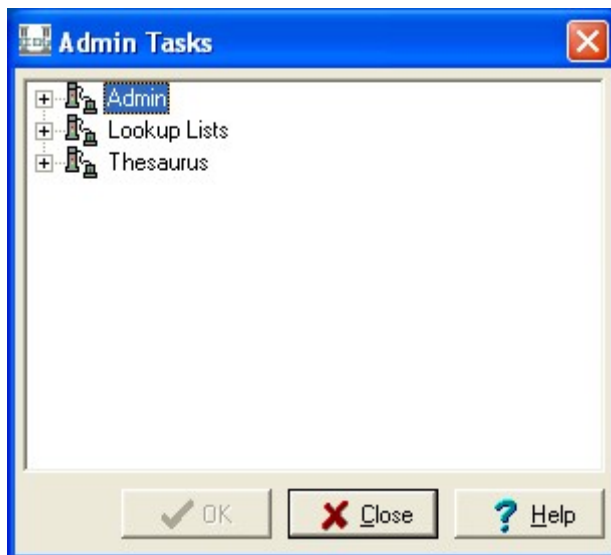
11.   Expand the Admin node:

12. Select **Change Password** and click [✓ OK].

The Admin Parameters dialogue displays:
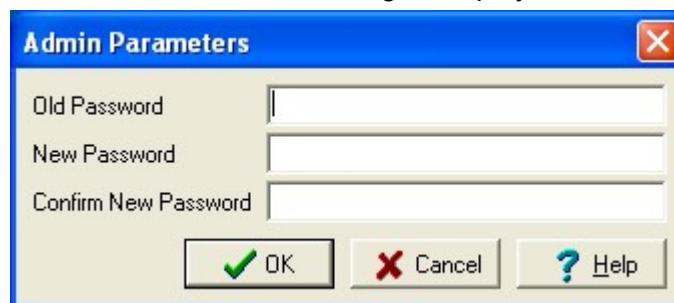
13. Enter your existing password.

14. Enter a new password and confirm it by re-entering it.

15. Click [✓ OK].

The password is updated on the server.

If the two instances of the new password do not match, an error message displays:

Re-enter the new password.

If the new password does not pass the validation criteria, an error message displays:
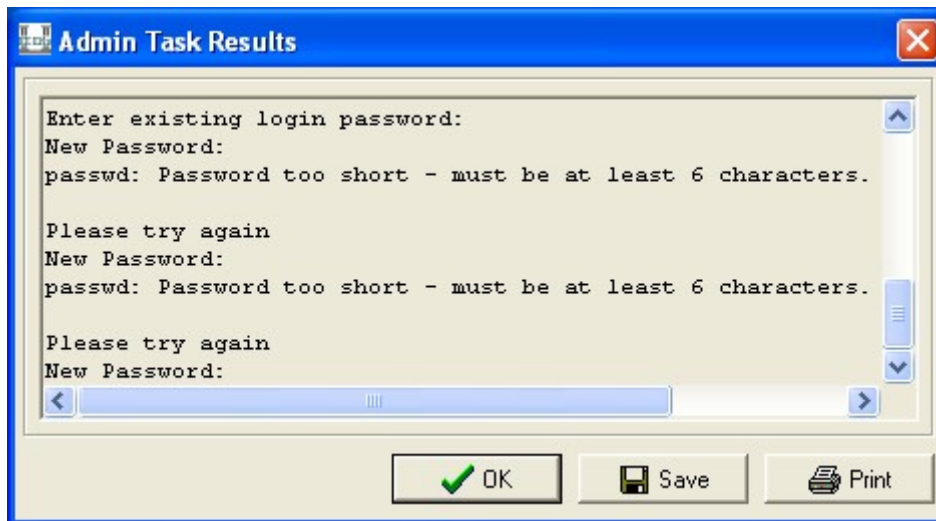
Enter a new password that passes validation.

16. Click [✓ OK] once the password is updated:

SECTION 3

Managing Passwords

In this section we look at the commands used by a System Administrator to manage user passwords. Unfortunately there is no utility common to all systems on which EMu runs (Unix and Windows) that provides a common interface to password management. In fact, even within the Unix family of systems no such utility exists. As such, we will look at the support provided by the four most common platforms on which the EMu server is installed:

- Solaris 10
- Linux
- FreeBSD
- Windows

1. Password Ageing

Password ageing allows a System Administrator to force users to change their password within a given number of days. For example, your institution may have a policy that users must change their passwords at least once a quarter.

Solaris 10

The `passwd` command is used to set up password ageing on Solaris 10. The format of the command is:

```
passwd -n min -x max user
```

where:

> *min*   is the minimum number of days required between password changes.
>
> Not supported by EMu due to limitations in Solaris.
>
> *max*   is the maximum number of days for which the password is valid. Once *max* days have elapsed without a password change, the user will be prompted for a new password at the next successful log in. A *max* value of `-1` is used to disable password ageing.
>
> *user*   is the name of the user account to which the restrictions are to apply.

The default values for *min* and *max* are defined in the file `/etc/default/passwd`. There are two variables used to set the minimum and maximum ageing values:

- `MINWEEKS`

  The minimum number of weeks between password changes. The default value is empty, implying no minimum is set.

  Not supported by EMu due to limitations in Solaris.

- `MAXWEEKS`

  The maximum number of weeks between password changes. The default value is empty, implying no maximum is set.

Setting the minimum and / or maximum default values to non-empty in `/etc/default/passwd` will result in users without ageing having it enabled the next time their password is modified.

Password ageing is supported by the Shadow and NIS+ password databases. LDAP and AD also support password ageing via the `shadowAccount` class object.

Linux

The `chage` command is used to set up password ageing on Linux. The format of the command is:

```
chage -m min -M max user
```

where:

> *min*   is the minimum number of days required between password changes. A value of zero indicate there is no minimum.
>
> *max*   is the maximum number of days for which the password is valid. Once *max* days have passed without a password change, the user will be prompted for a new password at the next successful log in. A `max` value of `99999` is used to disable password ageing.
>
> *user*   is the name of the user account to which the restrictions are to apply.

The default values for *min* and *max* are defined in the file `/etc/login.defs`. There are two variables used to set the minimum and maximum ageing values:

- `PASS_MIN_DAYS`

  The minimum number of days between password changes. The default value is zero, implying no minimum is set.

- `PASS_MAX_DAYS`

  The maximum number of days between password changes. The default value is `99999`, implying no maximum is set.

Setting the minimum and / or maximum default values to non-empty in `/etc/login.defs` will result in users without ageing having it enabled the next time their password is modified.

Password ageing is supported by the Shadow password database only. LDAP and AD also support password ageing via the `shadowAccount` class object, however changes to a user's settings must be made via `ldapmodify` or an ldap client (e.g. Active Directory Explorer for AD).

FreeBSD

FreeBSD provides a limited form of password ageing. Rather than setting a minimum and maximum number of days between password changes, it allows you to set the date on which a password should expire. Once the date arrives the user will be prompted for a new password. The `pw` command is used to set the date on which a password expires:

```
pw usermod user -p date
```

where:

> *user* is the name of the user account to which the restrictions are to apply.
>
> *date* is the date on which the password will expire. The date format is `dd-mmm-yyyy` (e.g. `23-Oct-2011`). An empty value is used to remove an expiry date.

When the new password is set the expiry date field is cleared.

If you want to have a new expiry date calculated automatically when a password is set, you need to specify the `passwordtime` attribute in the log in class file located at `/etc/login.conf`. The login class file allows a set of system attributes (resource usage, etc.) to be set on a login class basis. A user is then assigned to a login class using the `pw` command:

```
pw usermod user -L class
```

where:

> *user* is the name of the user account to be added to the login class.
>
> *class* is the class name as specified in the file `/etc/login.conf`.

To set the system wide password expiry date, the *default* login class should be modified to:

```
default:\
        :passwordtime=time:\
        ...
```

where:

> *time* is the time interval to set for a password to expire. A large number of formats are available for the value with `nnnd` being the most common. For example, `90d` would indicate the password will expire ninety days after it was last set.

If you change values in `/etc/login.conf`, you need to rebuild the internal database by executing:

```
cap_mkdb /etc/login.conf
```

Password ageing is supported by Unix and NIS+ password databases only. LDAP and AD also support password ageing via the `shadowAccount` class object, however changes to a user's settings must be made via `ldapmodify` or an ldap client (e.g. Active Directory Explorer for AD).

Windows

Password ageing is set on a Windows server running EMu via either a Local or Global Security Policy. The Local Security Policy editor is invoked by running `secpol.msc`. The Global Security Policy Editor is started by running `gpedit.msc`. The policy paths are:

- Local Security Policy

  `Security Settings/Account Policies/Password Policy`

- Global Security Policy

  [computer name] `Policy/Computer Configuration/Windows Settings/Security Settings/Account Policies/Password Policy`

There are two attributes used to set the minimum and maximum ageing values:

- *Minimum password age*

  The minimum number of days between password changes. A value of zero implies there is no minimum number of days.

- *Maximum password age*

  The maximum number of days between password changes. A value of zero implies there is no maximum number of days.

When a password expires, the user will be prompted to enter a new password when they next log in to Windows. EMu will not prompt for a new password when a Windows password expires, rather the user will not be able to access the system.

Examples

Example 1

Our institution has a policy that passwords must be changed at least once a quarter. There is no minimum time between changes.

Solaris 10

As the policy is a default setting the best solution is to edit `/etc/default/passwd` and set the following entries:

```
MINWEEKS=
MAXWEEKS=13
```

Linux

As the policy is a default setting the best solution is to edit `/etc/login.defs` and set the following entries:

```
PASS_MIN_DAYS=0
PASS_MAX_DAYS=90
```

FreeBSD

As the policy is a default setting the best solution is to edit `/etc/login.conf` and update the default login class to contain the following entry:

```
default:\
        :passwordtime=90d:\
        ...
```

Windows

If the setting is to be domain wide, then the Global Security Policy should be updated, otherwise if it is machine specific, the Local Security Policy should be used. The following attributes should be set:

`Minimum password age` set to `0`.
`Maximum password age` set to `90`.

Example 2

We have a default policy of forcing password changes every quarter, however we have a certain user (`boris`) for which we would like to disable password ageing.

Solaris 10

The following command may be used to disable password ageing for user `boris`:

```
passwd -x -1 boris
```

Linux

The following command may be used to disable password ageing for user `boris`:

```
chage -M 99999 boris
```

FreeBSD
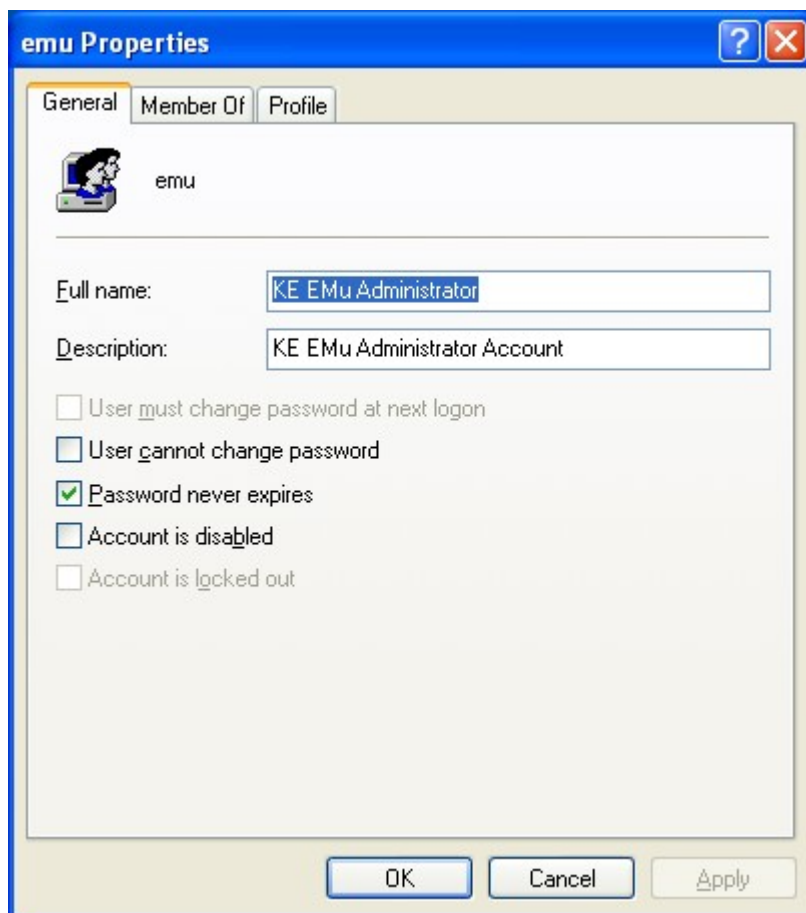
The following command may be used to disable password ageing for user `boris`:

```
pw usermod boris -p ''
```

Windows

View the properties for the user account. If Active Directory is enabled, run `dsa.msc` to view Active Directory users, otherwise run `lusrmgr.msc` to list local users. Right-click on user `boris` and select **Properties**. Turn on the *Password never expires* checkbox:

2. Password Reset

The password reset facility allows a System Administrator to force a user to change their password the next time they log in successfully. It is generally used when a user forgets their password. In this case the System Administrator sets a new password and then forces the user to change it the next time they access the system.

Solaris 10

The `passwd` command is used to force a user to change their password the next time they log in to EMu. The format of the command is:

```
passwd -f user
```

where:

>   *user*   is the name of the user account to reset.

Password resetting is supported by the Shadow and NIS+ password databases. LDAP and AD also support password resetting via the `shadowAccount` class object.

Linux

The `chage` command is used to force a user to change their password the next time they log in to EMu. The format of the command is:

```
chage -d 0 user
```

where:

>   *user*   is the name of the user account to reset.

Password resetting is supported by the Shadow and NIS+ password databases. LDAP and AD also support password resetting via the `shadowAccount` class object.

FreeBSD

The `pw` command is used to force a user to change their password the next time they log in to EMu. The format of the command is:

```
pw usermod user -p 01-jan-2000
```

where:

>   *user*   is the name of the user account to reset.

Password resetting is supported by the Shadow and NIS+ password databases. LDAP and AD also support password resetting via the `shadowAccount` class object.

Windows

View the properties for the user account. If Active Directory is enabled, run `dsa.msc` to

view the Active Directory users, otherwise run `lusrmgr.msc` to list local users. Turn off the *Password never expires* checkbox, then turn on the *User must change password at next logon* checkbox:
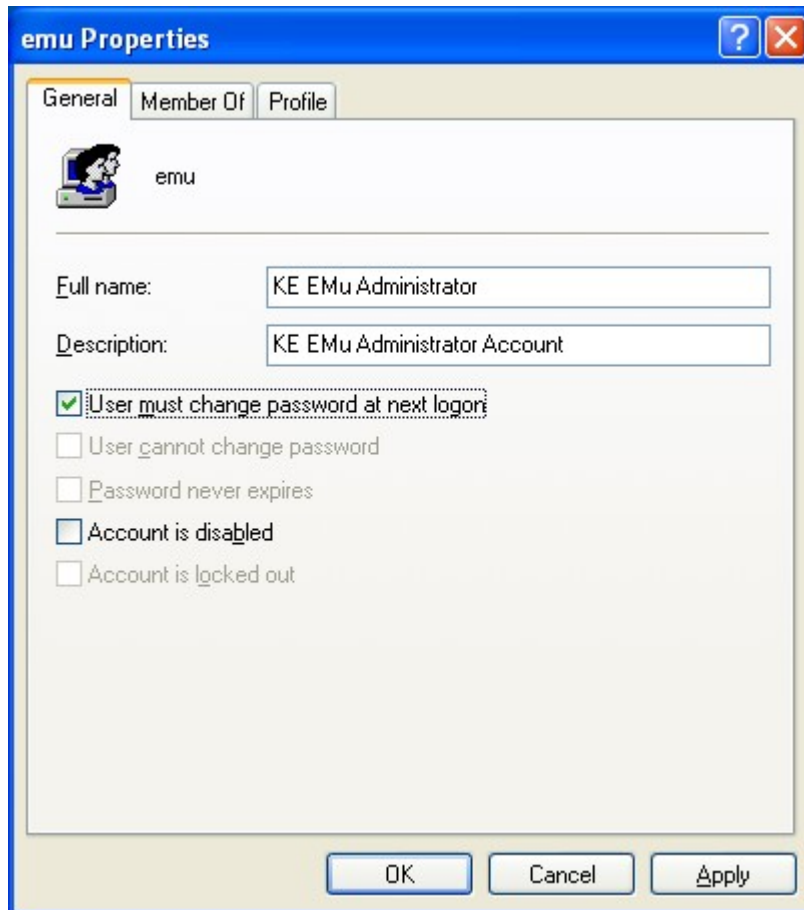
Example

User `boris` has forgotten his password. We would like to reset it to his user name and force him to change it the next time he logs in to EMu.

Solaris 10

The commands required to reset `boris`'s password are:

```
passwd boris
passwd -f boris
```

Linux

The commands required to reset `boris`'s password are:

```
passwd boris
chage -d 0 boris
```

FreeBSD

The commands required to reset `boris`'s password are:

```
passwd boris
pw usermod boris -p 01-jan-2000
```

Windows

If Active Directory is enabled, run `dsa.msc` to view the Active Directory users, otherwise run `lusrmgr.msc` to list local users. Right-click on user `boris` and select **Set Password....** Change the password as required. View the properties for user `boris` and turn off the *Password never expires* checkbox, then turn on the *User must change password at next logon* checkbox.

3. Account Ageing

The account ageing facility allows a System Administrator to set a date after which the user account becomes inactive. If a user tries to access EMu after the set date, a message indicating that the account has expired is displayed and access is denied.

Solaris 10

The `usermod` command is used to set / remove an expiry date for a user account. The format of the command is:

```
usermod -e date user
```

where:

|	
date	is the date after which the account is no longer valid. The date format used is `mm/dd/yy`. An empty date value is used to remove an expiry date.
user	is the name of the user account to expire.

Account ageing is supported by the Shadow and NIS+ password databases. LDAP and AD also support account expiry via the `shadowAccount` class object, however setting the expiry date must be done via an LDAP client, rather than the `usermod` command.

Linux

The `chage` command is used to set / remove an expiry date for a user account. The format of the command is:

```
chage -E date user
```

where:

date	is the date after which the account is no longer valid. The date format used is `yyyy-mm-dd`. An empty date value is used to remove an expiry date.
user	is the name of the user account to expire.

Account ageing is supported by the Shadow and NIS+ password databases. LDAP and AD also support account expiry via the `shadowAccount` class object, however setting the expiry date must be done via an LDAP client, rather than the `chage` command.

KE EMu
ELECTRONIC MUSEUM

FreeBSD

The `pw` command is used to set / remove an expiry date for a user account. The format of the command is:

```
pw usermod user -e date
```

where:

date	is the date after which the account is no longer valid. The date format used is `dd-mm-yyyy`. An empty date value is used to remove an expiry date.
user	is the name of the user account to expire.

Account ageing is supported by the Unix and NIS+ password databases. LDAP and AD also support account expiry via the `shadowAccount` class object, however setting the expiry date must be done via an LDAP client, rather than the `pw` command.

Windows

Account ageing is only supported for Active Directory user accounts. Local accounts cannot have an expiry date. Run `dsa.msc` and view the properties for the user account. On the *Account* tab turn on the *End of* radio button in the *Account expires* group box. Set the date after which the account will not be active:

Examples

Example 1

We have a number of students working for our institution over the summer break. We would like to expire their accounts once they return to university on 1st March 2012.

In each case below, it is not possible to set the expiry date for all students with the one command. Each user account needs to be set individually.

Solaris 10

The command required to set the expiry date is:

```
usermod -e '03/01/12' student1
```

Linux

The command required to set the expiry date is:

```
chage -E '2012-03-01' student1
```

FreeBSD

The command required to set the expiry date is:

```
pw usermod student1 -e '01-03-2012'
```

Windows

Run `dsa.msc` and view the properties for the `student1` user account. On the Account tab turn on the *End of* radio button in the *Account expires* group box. Set the expiry date to `1 March 2012`.

Example 2

One of the students (`student2`) is not returning to university and will continue to work for us for the foreseeable future. We need to remove their expiry date.

Solaris 10

The command required to remove the expiry date is:

```
usermod -e '' student2
```

Linux

The command required to remove the expiry date is:

```
chage -E '' student2
```

FreeBSD

The command required to remove the expiry date is:

```
pw usermod student2 -e ''
```

Windows

Run `dsa.msc` and view the properties for the `student2` user account. On the Account tab turn on the *Never* radio button in the *Account expires* group box.

4. Account Locking

The account locking facility allows a System Administrator to turn off access for a user account. While the account is locked, the user will not be able to log in to EMu. The error message will indicate an incorrect user name / password combination has been supplied. When an account is unlocked the user may access EMu again using their old password. Account locking is useful if a user is leaving for an extended period of time, but does plan to return in the future.

Solaris 10

The `passwd` command is used to lock / unlock a user's account. The format of the commands used to lock and unlock an account respectively is:

```
passwd -l user
passwd -u user
```

where:

> *user*   is the name of the user account to lock / unlock.

Account locking is supported by the Shadow and NIS+ password databases. LDAP and AD also support account locking via the `posixAccount` class object.

Linux

The `passwd` command is used to lock / unlock a user's account. The format of the commands used to lock and unlock an account respectively is:

```
passwd -l user
passwd -u user
```

where:

> *user*   is the name of the user account to lock / unlock.

Account locking is supported by the Shadow and NIS+ password databases. LDAP and AD also support account locking via the `posixAccount` class object.

FreeBSD

The `pw` command is used to lock / unlock a user's account. The format of the commands used to lock and unlock an account respectively is:

```
pw lock user
pw unlock user
```

where:

> *user*   is the name of the user account to lock / unlock.

Account locking is supported by the Unix and NIS+ password databases. LDAP and AD also support account locking via the `posixAccount` class object.

Windows

View the properties for the user account. If Active Directory is enabled, run `dsa.msc` to view the Active Directory users, otherwise run `lusrmgr.msc` to list local users. Turn on the *Account is disabled* checkbox:
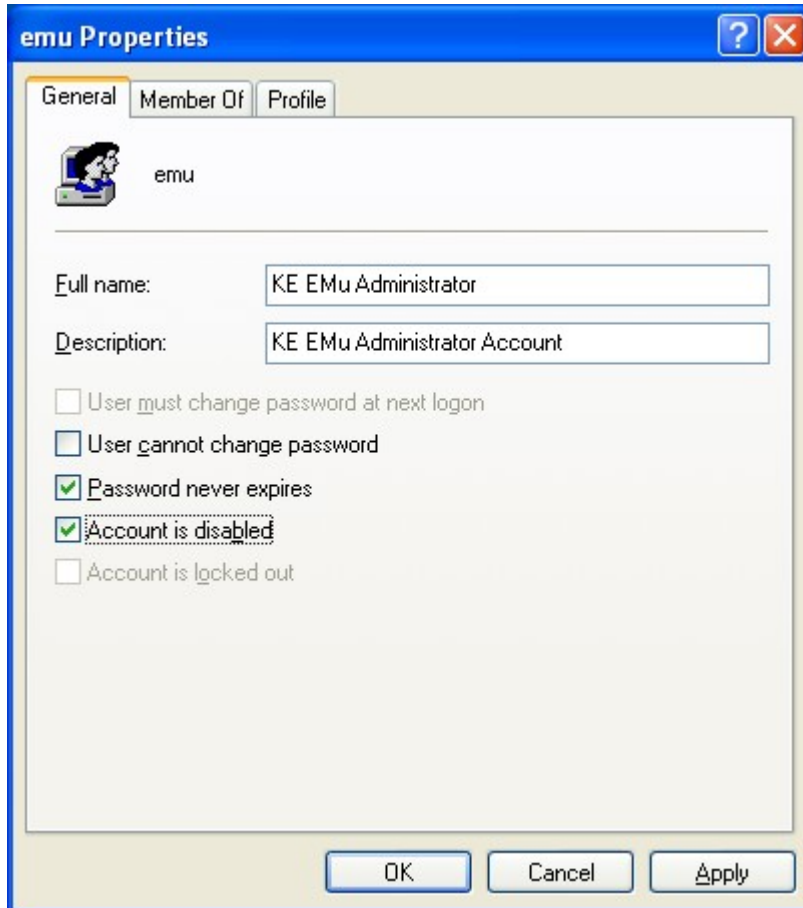
Examples

Example 1

User `boris` has taken leave for a year. We would like to lock his account while he is away.

Solaris 10

The command required to lock the user account is:

```
passwd -l boris
```

Linux

The command required to lock the user account is:

```
passwd -l boris
```

FreeBSD

The command required to lock the user account is:

```
pw lock boris
```

Windows

If Active Directory is enabled, run `dsa.msc` to view the Active Directory users, otherwise run `lusrmgr.msc` to list local users. View the properties for user account `boris`. Turn on the *Account is disabled* checkbox.

Example 2

User `boris` has now returned from leave and we would like to unlock his account.

Solaris 10

The command required to unlock the user account is:

```
passwd -u boris
```

Linux

The command required to unlock the user account is:

```
passwd -u boris
```

FreeBSD

The command required to unlock the user account is:

```
pw unlock boris
```

Windows

If Active Directory is enabled, run `dsa.msc` to view the Active Directory users, otherwise run `lusrmgr.msc` to list local users. View the properties for user account `boris`. Turn off the *Account is disabled* checkbox.

5. Maximum Retries

The maximum retries setting allows a System Administrator to set the maximum number of incorrect passwords that may be entered consecutively before the user's account is locked. Implementing a maximum retries limit is useful if you suspect people are trying to break into user accounts by guessing their password.

Solaris 10

Solaris 10 provides a variable called `RETRIES` in the file `/etc/default/login` for setting the maximum number of password retries before an account may be locked. The default value is empty, implying no limit.

Solaris 10, by default, does not lock an account when the limit is reached. In order to lock the account two mechanisms are provided. The first mechanism is system wide, while the second is on a per user basis. The system wide setting is found in the file `/etc/security/policy.conf`. The `LOCK_AFTER_RETRIES` variable must be set to `YES` to force accounts to be locked once the maximum number of retries is triggered. The setting may also be enabled on a per-user basis in the file `/etc/user_attr`. The `usermod` command is used to set the per-user value:

```
usermod -K lock_after_retries=value user
```

where:

value	is `no` (account locking is disabled once the retries limit is reached) or `yes` (account locking is enabled once the retries limit is reached).
user	is the name of the user account for which to set `lock_after_retries`.

Once an account has been locked due to exceeding the maximum number of tries, the user cannot log in to EMu until the account is unlocked (via `passwd -u` *user*).

Account locking after a maximum number of incorrect passwords is supported by the Shadow and NIS+ password databases. LDAP and AD also support account locking via the `posixAccount` and `shadowAccount` class objects.

Linux

Linux uses the `pam_tally` module in the PAM sub-system to implement account locking after a set number of failed passwords. In order to enable the module you need to include it in the Texpress PAM configuration file located at `/etc/pam.d/texpress`. The following two entries need to be added:

```
auth        required       pam_tally.so onerr=fail deny=5
account     required       pam_tally.so reset
```

The order of these two lines in the PAM configuration file is important. The first line must appear **before** the entry for:

```
auth ... pam_unix.so
```

and the second line must appear **after** the entry for:

```
account ... pam_unix.so
```

The `deny` argument determines the number of failed password attempts before the user can no longer log in to the system.

Once a user has reached the `deny` limit, they cannot log in until the number of failed passwords is reset to zero. The `faillog` command is used to reset the count (and hence allow the user to retry logging in):

```
faillog -r -u user
```

where:

> *user*   is the name of the user account to unlock.

It is also possible to set the maximum number of password attempts on a per user basis, rather than system wide. If per-user limits are to be used, the PAM entries in the configuration file should be altered to:

```
auth        required       pam_tally.so onerr=fail deny=5 per_user
account     required       pam_tally.so reset
```

The `faillog` command is then used to set the per-user limit:

```
faillog -u user -m max
```

where:

> *max*   is the number of password failures before the account is disabled. A value of zero turns off checking.
>
> *user*   is the name of the user account for which to set the limit.

FreeBSD

Like Linux, FreeBSD also uses PAM to provide support for setting a maximum number of login attempts before a user is locked out. FreeBSD uses the `pam_abl` module to provide the required PAM support. The `pam_abl` package is not installed on a FreeBSD system by default. You may need to install the package before enabling it.

To enable the module you need to include it in the Texpress PAM configuration file located at `/etc/pam.d/texpress`. The following entry needs to be added:

```
auth    required    /usr/local/lib/pam_abl.so
config=/usr/local/etc/pam_abl.conf
```

The position of this line in the PAM configuration file is important. The line must appear **before** the entry for:

```
auth ... pam_unix.so
```

The `config` argument defines the location of the `pam_abl` configuration file. The configuration file allows system wide and per-user properties to be set.

To set a system wide entry, edit the configuration file located at `/usr/local/etc/pam_abl.conf` and alter the `host_rule` property. The format of the setting required to set the maximum login retries is:

```
*:retries/period
```

where:

   retries is the number of login attempts before locking out the user.

   period is the period in which the retries have to occur for the user to be locked out. The period is generally a number of days, for example `30d`. After the period has expired, the account may be accessed again.

To set a user specific entry, the `user_rule` should be used. The format of a user specific entry is:

```
user:retries/period
```

where:

   user is the name of the user account to which the setting applies.

The remaining values are the same as for the `host_rule` property. More sophisticated rules may be set. Please see the manual entry for `pam_abl` for complete details.

The `pam_abl` command is used to clear an account once it has been disabled. The format of the command is:

```
pam_abl --okuser user
```

where:

   user is the name of the user account to be re-enabled.

KE EMu
ELECTRONIC MUSEUM

Windows

Disabling an account after a maximum number of login attempts is set on a Windows server running EMu via either a Local or Global Security Policy. The Local Security Policy editor is invoked by running `secpol.msc`. The Global Security Policy Editor is invoked by running `gpedit.msc`. The policy paths are:

- Local Security Policy

  `Security Settings/Account Policies/Account Lockout Policy`

- Global Security Policy

  [computer name] `Policy/Computer Configuration/Windows Settings/Security Settings/Account Policies/Account Lockout Policy`

The property used to control the maximum number of login attempts before disabling an account is:

- `Account lockout threshold` [the number of password attempts after which the account will be locked]

Once an account has been disabled, it must be re-enabled before the user can access EMu.

Examples

Example 1

Our institution has a policy of allowing five failed log in attempts before locking a user account. To unlock the account the user must contact the System Administrator.

Solaris 10

The setting required in the file `/etc/default/login` is:

```
RETRIES=5
```

and the setting required in the file `/etc/security/policy.conf` is:

```
LOCK_AFTER_RETRIES=YES
```

Linux

Add the following line to `/etc/pam.d/texpress`:

```
auth required pam_tally.so onerr=fail deny=5
```

The line should be added before the line:

```
auth ... pam_unix.so
```

FreeBSD

Install `pam_abl` and add the following line to the PAM configuration file located at `/etc/pam.d/texpress` (note that this should be entered on a single line):

```
auth required /usr/local/lib/pam_abl.so
config=/usr/local/etc/pam_abl.conf
```

The line should be added before the line:

```
auth ... pam_unix.so
```

Edit the `pam_abl` configuration file located at `/usr/local/etc/pam_abl.conf` and add the line:

```
host_rule=*:5/1h
```

The rule will allow up to five incorrect log in attempts per hour.

Windows

If the setting is to be domain wide, then the Global Security Policy should be updated; otherwise, if it is machine specific, the Local Security Policy should be used. The following attribute should be set:

`Account lockout threshold` set to `5`.

KE EMu
ELECTRONIC MUSEUM

Example 2

We have one particular user (`boris`) who suffers from dyslexia, so we do not want to lock his account after five incorrect password submissions.

Solaris 10

The command required to disable account locking for user `boris` is:

```
usermod -K lock_after_retries=no boris
```

Linux

Edit the PAM configuration file located at `/etc/pam.d/texpress` and add the `per_user` property to the `pam_tally` auth entry:

```
auth required pam_tally.so onerr=fail deny=5 per_user
```

The command required to disable account locking for user `boris` is:

```
faillog -m 0 -u boris
```

FreeBSD

Edit the `pam_abl` configuration file located at `/usr/local/etc/pam_abl.conf` and add the following entry:

```
user_rule=boris:100/1s
```

While the above rule does not disable the checking, it will only lock out a user if there are more than 100 login attempts per second (which is very unlikely).

Windows

Windows does not provide a per-user setting for the number of login attempts before locking a user's account.

6. Valid Passwords

When a user submits a new password for updating it may be worthwhile checking that the password is sufficiently obscure as to not be easily guessed. The check may include looking for:

- known words
- a mixture of characters and digits
- a mixture of lower case and upper case characters
- a minimum length

The password validation facility provides a mechanism where new passwords can be checked against a number of criteria before being approved for updating.

Solaris 10

Solaris 10 provides support for checking that a new password supplied by a user meets a minimum set of conditions. The conditions are defined in the file `/etc/default/passwd` and are checked each time a user sets a new password. If the new password does not pass all checks, it is disallowed. The checks available are:

`DICTIONDBDIR`	The path of a directory containing a series of pre-compiled dictionary files. The files are checked to see if a new password appears in one of them. If so, the password is rejected. The `mkpwdict` command is used to build the pre-compiled dictionary files. The default value is empty, implying a dictionary check is not performed.
`DICTIONLIST`	A comma separated list of full file names to dictionary files. Each dictionary file should contain one word per line with a newline character used to end the line. A new password is checked against the contents of the dictionary files before being passed. The default value is an empty list, implying dictionary look ups are not performed.
`HISTORY`	The maximum number of prior passwords to maintain for each user. When a user submits a new password it is checked against the last `HISTORY` passwords the user has set and if a match occurs, the password is disallowed. If the value for `HISTORY` is empty or zero, password histories are disabled. The default value is empty, implying password histories are disabled. Password histories are only supported by the Shadow password database.
`MINALPHA`	The minimum number of alphabetic characters required. If `MINALPHA` is not set, the default is `2`.
`MINDIGIT`	The minimum number of digits required. If `MINDIGIT` is not set or is set to zero, the default is no checks. You cannot set `MINDIGIT` if `MINNONALPHA` is specified.

MINNONALPHA	The minimum number of non-alpha characters (including numeric and special) required. If MINNONALPHA is not set, the default is 1. You cannot set MINNONALPHA if either MINDIGIT or MINSPECIAL is specified.
MINSPECIAL	The minimum number of special characters (non-alphabetic and non-digit) required. If MINSPECIAL is not set or is zero, the default is no checks. You cannot set MINSPECIAL if you specify MINNONALPHA.
MINLOWER	The minimum number of lower case letters required. If not set or zero, the default is no checks.
MINUPPER	The minimum number of upper case letters required. If not set or zero, the default is no checks.
MINDIFF	The minimum character differences required between an old and a new password. If MINDIFF is not set, the default is 3.
NAMECHECK	Enable / disable checking for the user name as the password. The default is to perform the check. A value of NO disables this feature.
PASSLENGTH	The minimum acceptable length for a password in characters.
WHITESPACE	Whether whitespace characters (space, tab, etc.) are acceptable in a password. The default value is that whitespace is acceptable. A value of NO disables this feature.

As you can see Solaris 10 provides quite a bit of control over what constitutes a valid password. The settings above are on a system wide basis. It is not possible to apply any of the settings on a per-user basis.

Linux

Linux uses the `pam_cracklib` module in the PAM sub-system to implement password validation. In order to enable the module you need to include it in the Texpress PAM configuration file located at `/etc/pam.d/texpress`. The following two entries need to replace the existing `password` entry:

```
password       required       pam_cracklib.so minlen=6 difok=3
password       required       pam_unix.so use_authtok
```

The `pam_cracklib` module takes a number of parameters, including:

- `minlen`

  The minimum length for an acceptable password.

- `difok`

  The minimum number of characters difference between the previous password and the new password.

- `dictpath`

  Full file prefix for `cracklib` dictionaries. The default value is `/usr/lib/cracklib.dict`. The `cracklib` dictionary extensions are:

  - `.hwm`
  - `.pwd`
  - `.pwi`

`pam_cracklib` provides a dictionary containing over 50,000 words. It is also possible to configure `pam_cracklib` to enforce passwords to contain a mixture of lowercase, uppercase, digits and special characters. See the `pam_cracklib` manual page for complete details on how to configure these restrictions.

FreeBSD

FreeBSD uses the `pam_passwdqc` module in the PAM sub-system to implement password validation. In order to enable the module you need to include it in the Texpress PAM configuration file located at `/etc/pam.d/texpress`. The following two entries need to replace the existing `password` entry:

```
password    requisite     pam_passwdqc.so enforce=users
password    required      pam_unix.so no_warn try_first_pass
```

The `pam_passwdqc` module takes a number of parameters, including:

- `enforce`

   Indicates whether weak passwords should be allowed. The allowed values are:
   - `everyone` - strong password are enforced for all users.
   - `users` - strong passwords are enforced for all non-root users.
   - `none` - strong passwords are not enforced.

- `max`

   The maximum allowed password length. The default value is `40`.

- `match`

   The length of a common sequence between old and new passwords for the new password to be considered unsuitable. The default value is four, while a value of zero disables sub-string matching.

- `min`

   The minimum allowed password lengths for different combinations of character sequences. A password is broken down into a number of characters classes. The classes are:
   - digits
   - lower-case letters
   - upper-case letters
   - other characters

   The format of the setting for `min` is:

   `min=N0,N1,N2,N3,N4`

   where:

`N0`	is the minimum length where a password contains characters from one class only.
`N1`	is the minimum length where a password contains characters from two classes.
`N2`	is the minimum number of words for a passphrase (not used by EMu).
`N3`	is the minimum length where a password contains characters from three classes.
`N4`	is the minimum length where a password contains characters from all four classes.

   A value of disabled is used to disallow passwords of a given format. The default value is `min=disabled,24,12,8,7`.

`pam_passwdqc` does not provide a dictionary look-up facility, however judicious use of

the `min` property forces users to submit passwords with non-obvious sequences.

Windows

Enabling password complexity requirements is set on a Windows server running EMu via either a Local or Global Security Policy. The Local Security Policy editor is invoked by running `secpol.msc`. The Global Security Policy Editor is invoked by running `gpedit.msc`. The policy paths are:

- Local Security Policy
  `Security Settings/Account Policies/Password Policy`
- Global Security Policy
  [computer name] `Policy/Computer Configuration/Windows Settings/Security Settings/Account Policies/Password Policy`

The property used to control password complexity is:

- *Password must meet complexity requirements* - specifies whether password complexity checks are enabled or disabled. The default value for a domain controller is enabled, otherwise disabled. The password complexity requirements are:
  - must not contain the user's account name or parts of the user's full name that exceed two consecutive characters.
  - must be at least six characters in length.
  - must contain characters from three of the following four categories:
    - upper-case characters (A through Z)
    - lowercase characters (a through z)
    - digits (0 through 9)
    - non-alphabetic characters (for example, !, $, #, %)

The property used to control password histories is:

- *Enforce password history* - specifies the number of unique new passwords required before an old password may be re-used. The default value is `24` for a domain controller, otherwise a value of zero is used.

Windows does not provide a dictionary look-up facility.

SECTION 4

PAM Configuration

PAM (Pluggable Authentication Modules) is a very flexible authentication system. As the name implies, it allows modules to be plugged in to provide specific functionality. Each module looks after some part of the authentication process with the combination of the results of each module determining whether access is granted.

For example, there is a module that provides LDAP functionality and another that provides Unix / Shadow functionality and so on. In order to provide the password checks and updates required by a given institution it is necessary to adjust the PAM configuration to match the institution's policy. If an institution uses Active Directory to manage users, then the PAM LDAP module must be enabled; if an institution uses dongles, then the required PAM module (e.g. `pam_usbng`) needs to be enabled.

It is beyond the scope of this document to explain how to configure PAM (there are plenty of good sources available on the internet). Rather we will look at configurations required to support the functionality required for password management on:

- Solaris 10
- Linux
- FreeBSD

In order to provide general support for the password database used by your institution (LDAP, AD, Shadow, etc.) within EMu you need to not only configure PAM, but also NSS (Name Service Switch). The combination of PAM and NSS on Unix systems provide the integration required to communicate with the various user / password databases. An explanation of NSS is beyond the scope of this document, however sample NSS configurations will be provided. The NSS configuration file is located at `/etc/nsswitch.conf`.

The PAM configurations outlined in this section apply to the setup required by EMu only. The configurations do not provide general purpose account access to the server via PAM, rather they allow EMu to be configured to use the required user / password database. The configurations show the settings required within the listed file, not the complete contents of the file. Thus if you are configuring the PAM and NSS settings you will need to amend the contents of the existing file, rather than replace them.

Solaris 10

The PAM configuration file used by Solaris 10 is located at `/etc/pam.conf`. The file contains the configuration for all PAM services, rather than one service per file (as is used by Linux and FreeBSD).

Shadow

The PAM and NSS configuration file segments required to provide *Shadow* database support are:

```
/etc/pam.conf
#
# EMu Texpress service
#
  texpress          auth sufficient          pam_rhosts_auth.so.1
  texpress          auth requisite           pam_authtok_get.so.1
  texpress          auth required            pam_dhkeys.so.1
  texpress          auth required            pam_unix_cred.so.1
  texpress          auth required            pam_unix_auth.so.1
/etc/nsswitch.conf
  passwd:     files
```

NIS & Shadow

The PAM and NSS configuration file segments required to provide NIS and Shadow database support are:

```
/etc/pam.conf
#
# EMu Texpress service
#
  texpress          auth sufficient          pam_rhosts_auth.so.1
  texpress          auth requisite           pam_authtok_get.so.1
  texpress          auth required            pam_dhkeys.so.1
  texpress          auth required            pam_unix_cred.so.1
  texpress          auth required            pam_unix_auth.so.1
/etc/nsswitch.conf
  passwd:     files nis
```

LDAP / AD & Shadow

The PAM and NSS configuration file segments required to provide LDAP or AD and Shadow database support are:

```
/etc/pam.conf
#
# EMu Texpress service
#
  texpress        auth sufficient       pam_rhosts_auth.so.1
  texpress        auth requisite        pam_authtok_get.so.1
  texpress        auth required         pam_dhkeys.so.1
  texpress        auth required         pam_unix_cred.so.1
  texpress        auth binding          pam_unix_auth.so.1
server_policy
  texpress        auth required         pam_ldap.so.1

  #
  # Default Account service
  #
  other           account requisite     pam_roles.so.1
  other           account binding       pam_unix_account.so.1
server_policy
  other           account required      pam_ldap.so.1

  #
  # Password checking (used by Admin Task only)
  #
  passwd          auth binding          pam_passwd_auth.so.1
server_policy
  passwd          auth required         pam_ldap.so.1

  #
  # Default Password service
  #
  other      password required     pam_dhkeys.so.1
  other      password requisite     pam_authtok_get.so.1
  other      password requisite     pam_authtok_check.so.1
  other      password required     pam_authtok_store.so.1
server_policy
/etc/nsswitch.conf
  passwd:     files ldap
```

If you select LDAP support, you will need to configure how to bind to the LDAP server. Use the `ldapclient` command to specify these settings.

Linux

The PAM configuration file used by Linux is located at `/etc/pam.d/texpress`. The file contains the configuration for EMu services only.

Shadow

The PAM file and NSS file segments required to provide Shadow database support are:

```
/etc/pam.d/texpress
   #
   # EMu Texpress service
   #
   auth          required      pam_env.so
   auth          required      pam_unix.so nullok try_first_pass

   account       required      pam_unix.so

   password      requisite     pam_cracklib.so try_first_pass
   password      required      pam_unix.so md5 shadow nullok
try_first_pass use_authtok
/etc/nsswitch.conf
   passwd:       files
   shadow:       files
```

NIS & Shadow

The PAM file and NSS file segments required to provide NIS and Shadow database support are:

```
/etc/pam.d/texpress
   #
   # EMu Texpress service
   #
   auth          required      pam_env.so
   auth          required      pam_unix.so nullok try_first_pass

   account       required      pam_unix.so

   password      requisite     pam_cracklib.so try_first_pass
   password      required      pam_unix.so md5 shadow nullok
try_first_pass use_authtok
/etc/nsswitch.conf
   passwd:       files nis
   shadow:       files nis
```

LDAP / AD & Shadow

The PAM file and NSS file segments required to provide LDAP or AD and Shadow database support are:

```
/etc/pam.conf
   #
   # EMu Texpress service
   #
   auth        required      pam_env.so
   auth        sufficient    pam_ldap.so
   auth        required      pam_unix.so nullok try_first_pass

   account     sufficient    pam_ldap.so
   account     required      pam_unix.so

   password    requisite     pam_cracklib.so try_first_pass
   password    sufficient    pam_ldap.so
   password    required      pam_unix.so md5 shadow nullok
try_first_pass use_authtok
/etc/nsswitch.conf
   passwd:     files ldap
   shadow:     files ldap
```

If you select LDAP support, you will need to configure how to bind to the LDAP server. The LDAP configuration file is located at `/etc/ldap.conf`. See the manual entry for `ldap.conf` for details on how to bind to an LDAP/AD server.

FreeBSD

The PAM configuration file used by FreeBSD is located at `/etc/pam.d/texpress`. The file contains the configuration for EMu services only.

Unix

The PAM file and NSS file segments required to provide Unix database support are:

```
/etc/pam.d/texpress
   #
   # EMu Texpress service
   #
   auth         required      pam_unix.so try_first_pass

   account      required      pam_login_access.so
   account      required      pam_unix.so

   password     requisite     pam_passwdqc.so enforce=users
   password     required      pam_unix.so try_first_pass
/etc/nsswitch.conf
   passwd:          files
   passwd_compat:   nis
```

NIS & Unix

The PAM file and NSS file segments required to provide NIS and Unix database support are:

```
/etc/pam.d/texpress
   #
   # EMu Texpress service
   #
   auth         required      pam_unix.so try_first_pass

   account      required      pam_login_access.so
   account      required      pam_unix.so

   password     requisite     pam_passwdqc.so enforce=users
   password     required      pam_unix.so try_first_pass
/etc/nsswitch.conf
   passwd:          files nis
   passwd_compat:   nis
```

LDAP / AD & Unix

The PAM file and NSS file segments required to provide LDAP or AD and Unix database support are:

```
/etc/pam.conf
   #
   # EMu Texpress service
   #
   auth         sufficient    /usr/local/lib/pam_ldap.so
try_first_pass
   auth         required      pam_unix.so try_first_pass

   account      required      pam_login_access.so
   account      sufficient    /usr/local/lib/pam_ldap.so
   account      required      pam_unix.so

   password     requisite     pam_passwdqc.so enforce=users
   password     sufficient    /usr/local/lib/pam_ldap.so use_authtok
   password     required      pam_unix.so try_first_pass
/etc/nsswitch.conf
   passwd:             files ldap
```

If you select LDAP support, you will need to configure how to bind to the LDAP server. The PAM LDAP configuration file is located at:

```
/usr/local/etc/ldap.conf
```

A copy of the configuration file should be linked to:

```
/usr/local/etc/nss_ldap.conf
```

to provide the required NSS LDAP configuration.

Index