PWQASEEP

Password Quality
Security Event Exit Process
Version 402

Reference Manual

15. August 2006

greenHouse
Software & Consulting
Karl-Heinz Weber
Heinrichstraße 12
D-45711 Datteln/Horneburg

Document History 9th Edition 15Aug2006 Release PWQASEEP 402

Trademarks or Service Marks

The following are trademarks or service marks of Tandem Computers Incorporated:
Atalla, Challenge/Response, Enform, Expand, Guardian, Guardian90, Inspect, Multilan, NonStop, TACL, Tandem.
All brand names and product names are trademarks or registered trademarks of their respective companies.

The following are trademarks or service marks of *GreenHouse Software & Consulting:*
$ARROW, $AS, CRYSTAL, CURIOUS, FTPSERV-E, FUNCTRAC, MPWD, MPWD-L, PASSYNC, SECMAN, SECOM, GSTK, SSTK.

The following are trademarks or service marks of Jelinek EDV:
SECMAIN

Please Comment

If you have questions or problems concerning the content of this document, please let me know! Send your comments to:

GreenHouse Software & Consulting
Karl-Heinz Weber
Heinrichstraße 12
D-45711 Datteln/Horneburg
Germany
Phone +49 (0)2363 72566
Fax +49 (0)2363 66106
Mobile +49 (0)172 23 18248
E-Mail: Info@GreenHouse.de
Internet: www.GreenHouse.de
PGP fingerprint: 3A32 D90A D125 5418
1150 2484 6629 2DD2

Index

Introduction

The standard authentication method on Tandem systems is based on static passwords.
A users credentials are:

- The unique ID
  The ID is defined by the users manager when the user is introduced to the system. It can be changed by the users manager.
  The ID is used to decide about access rights.
- The corresponding password
  The password is set when the user is introduced, but can be changed by the user, depending on the password life time criteria, defined in SAFEGUARD.

When SAFEGUARD is active, the password lifetime is controlled according to NCSC's[1] C2[2] requirements, but there is no quality filter delivered by Tandem.
The solution from Tandem is a so-called Security Event Exits (SEE), which interfaces to user written SEE Processes (SEEP). One SEE type interface is defined for password quality filtering.

The Password Quality Security Event Exit Process (PWQASEEP) is a process, that interfaces exactly to this SEE.

The password quality SEEP is used when:

- a user is introduced by SAFECOM and the users initial password is set
- the password of an existing user is changed by his manager
- the user logs on through a TACL and changes his password
- the user uses the PASSWORD program.
- an application process uses the GUARDIAN procedure call User_Authenticate_ and its ability to change the password during the authentication cycle

[1] NCSC = National Computer Security Center, USA
[2] C" = evaluation level according to the TCSEC

Installation

The PWQASEEP product is delivered on a CD and comes as a self extracting, self installing PAK type file, named **QUALITY.**

1. Transfer **QUALITY.100** in BINARY mode onto the Tandem system e.g. into location **$SYSTEM.PWQASEEP** (the location does not matter, but it is strongly recommended to put this software on $SYSTEM!).

2. Change the file code of QUALITY to 100 when necessary:
   ```
   FUP ALTER CODE QUALITY,100
   ```

3. Execute the QUALITY program file:
   ```
   RUN QUALITY
   ```

   - This installs the password quality SEEP product in the current location
   - It adjusts all location dependent entries in the TACL macros
   - It sets the security attributes to the most stringent values

   When the installation is finished, the following files exist in the PWQASEEP location:

```
$SYSTEM.PWQASEEP

              CODE                    Meaning

DICTTXT         101    password template text file
DINSTALL        101    de-installation TACL macro
GUESSES       18248    collection of passwords with some 2,000,000 entries
INFOSEEP        101    TACL Macro to display SEEP attributes
INSTALL         101    installation TACL macro
LOADDICT        100    program to load the DICTTXT file into PWQADICT, and to
                       add new entries into the GUESSES file
PWQADDL         101    Log file DDL
PWQADICT      18248    password template dictionary
PWQAHUB         101    Password rule hub file
PWQARULE        101    PWQASEEP configuration and rules file
PWQASEEP        100    password quality security event exit program
[PWQASTOK       101    LicenseToken   (see point [8]( below)]
QUALITY         100    self extracting, self installing PAK archive
RESTART         101    PWQASEEP restart TACL macro
SEEPSYNC      18248    data file, used with PASSYNC
```

4. Edit the file **DICTTXT**: It contains some template example entries of not allowed passwords. The number of entries is limited to 32767.
   Extended wildcards are supported.

5. Execute the **LOADDICT** program with the following command to load the **DICTTXT** file into the **PWQADICT** dictionary file:
   ```
   run PWQADICT/IN DICTTXT,OUT PWQADICT/ DICT
   ```
   Changes in this file are automatically taken into account by **PWQASEEP** WITHOUT the need to re-start it!

6. Edit the **PWQAHUB** file to configure user specific rule files.
   PWQAHUB allows user specific password quality rules, e.g. user of group BASE24 have to follow different rules than all other users.
   Users templates are OK.
   Up to 1,000 entries are supported.
   All entries exceeding the maximum number of 1,000 are ignored.

7. Edit the **PWQARULE** file to configure the quality attributes.
   Changes in this file are automatically taken into account by **PWQASEEP** WITHOUT the need to re-start it!
   In case you defined multiple rule files in PWQAHUB make sure, all these files exist!

8. In case there is no PWQASTOK file, you need to get it from the product CD as described on the CD cover:
   - Upload your LicenseToken file in binary mode from the CD onto the Tandem system, and name it e.g. TOKEN
   - Change the file code of TOKEN to 1729:
   ```
   FUP ALTER TOKEN,CODE 1729
   ```
   - Unpack TOKEN:
   ```
   UNPAK -PASSWORD <password> TOKEN,*.*.PWQASTOK,VOL $vol.subvol,MYID,LISTALL
   ```
   where <password> is the password GreenHouse communicated to you.

Managing PWQASEEP

To activate the PWQASEEP process, SAFEGUARD has to be made aware of it.
Four TACL Macros are supplied along with the product to manage the PWQASEEP process:

1. **INSTALL**
   Activates PWQASEEP in SAFEGUARD.
2. **RESTART**
   Restarts the PWQASEEP.
3. **DINSTALL**
   Removes PWQASEEP from SAFEGUARD.
4. **INFOSEEP**
   Displays SAFEGUARD information of the Password Quality Event Exit Process.

To successfully manage a SEEP process, the manager must be configured in SAFEGUARD as SECURITY-ADMINISTRATOR with EXECUTION access rights.

Installation

To add the Password Quality SEEP to SAFEGUARD, execute the **INSTALL** TACL Macro, which is part of the product delivery.

Before you do so, please check the two entries

- **response-timeout**
- **prog**

in the INSTALL TACL Macro and adjust them to your requirements.

```
?TACL Macro
#FRAME
==
== Install the Password Quality SEEP PWQASEEP from GreenHouse
==
#PUSH #INLINEPROCESS

INLPREFIX +
SAFECOM/INLINE/

+ add   event-exit-process password_quality
+ alter event-exit-process password_quality response-timeout 10
+ alter event-exit-process password_quality enable-password-event on
+ alter event-exit-process password_quality prog $system.pwqaseep.pwqaseep
+ alter event-exit-process password_quality pri 199
+ alter event-exit-process password_quality enabled on

INLEOF

#POP #INLINEPROCESS
#UNFRAME
```

Restart

The SEEP is controlled by the $ZSMP process and kept alive all the time. In case the PWQASEEP process had to become restarted, e.g. when a new program file has to be activated, the execution of the **RESTART** TACL Macro can be used to perform this.

```
?TACL Macro
#FRAME
#PUSH #INLINEPROCESS

INLPREFIX +
SAFECOM/INLINE/
+ alter event-exit-process password_quality enabled off
+ alter event-exit-process password_quality enabled on
INLEOF

#POP #INLINEPROCESS
#UNFRAME
```

Delete

There is no reason to de-install the PWQASEEP. However, in case it has to become de-installed, execute the TACL Macro **DINSTALL**:

```
?TACL Macro
#FRAME
==
== De-Install the Password Quality SEEP PWQASEEP from GreenHouse
==
#PUSH #INLINEPROCESS

INLPREFIX +
SAFECOM/INLINE/

+ alter  event-exit-process password_quality enabled off
+ delete event-exit-process password_quality

INLEOF

#POP #INLINEPROCESS
#UNFRAME
```

InfoSEEP

To display SAFEGUARD SEEP settings run the INFOSEEP TACL Macro:

```
?TACL Macro
==
==  This TACL Macro displays all SEEP relevant attributes.
==  Not high sophisticated - but helpful...
==
#FRAME
SAFECOM info event-exit-process password_quality
#UNFRAME
```

It displays:

```
$GHS1 QUALITY 86> infoseep

  EVENT-EXIT-PROCESS  PASSWORD_QUALITY

    ENABLED = ON
    RESPONSE-TIMEOUT =   10 SECONDS
    ENABLE-AUTHENTICATION-EVENT = OFF
    ENABLE-AUTHORIZATION-EVENT  = OFF
    ENABLE-PASSWORD-EVENT       = ON
    PROG  = $GHS1.QUALITY.PWQASEEP
    LIB   = * NONE *
    PNAME = * NONE *
    SWAP  = * NONE *
    CPU   = ANY
    PRI   =      199

    PARAM-TEXT =
$GHS1 QUALITY 87>
```

Testing PWQASEEP

The PWQASEEP program has an interactive interface that allows the security manger, as well as the user, to test the quality rules, and to test a password.
In case a password is rejected, an explanation is given why it is rejected.

To run the PWQASEEP program interactively, execute the command

```
[run] PWQASEEP [/CPU n, …/]
```

and the following messages are displayed:

```
$GHS1 QUALITY 46> pwqaseep
PWQASEEP (401) - T7172G06 - (21Feb2006) System \BEECH, running NSK G06
Copyright (c) GreenHouse Software & Consulting 2002-2006
You are user:                SA.CARL (GUARDIAN user)
SEEP rules in use:           $GHS1.QUALITY.PWQACARL
Obey files used:             $GHS1.QUALITY.PWQAGLOB
Password dictionary in use: $GHS1.QUALITY.PWQADICT
Number of entries in dict:  35
Password guesses file:       $GHS1.QUALITY.GUESSES
Passwords are checked as:    NOT CASE SENSITIVE
SEEPSYNC is:                 NOT defined
PASSYNC is:                  NOT defined
EMS collector process is:    $0
Minimum password length:     7
Num Special is:              0 - 8
Num Capital is:              0 - 8
Num LowerCase is:            1 - 8
Num Numeric is:              0 - 8
Num Alpha is  :              4 - 8
Num Multiple is:             2
Max num Multiple is:         8
Max num user name in PW:     3
Template:                    ********
Password:
```

The displayed information reflects the PWQASEEP configuration as it is read from the PWQARULE file, and the PWQADICT file.

The PASSWORD prompt then enables the user to enter a password, e.g.:

```
Password: CarlWeber
Password to long

Password: CarlWeb
Password to short

Password: Carl1234
*** Password rejected by: contains user name fragments

Password: 123carl5
*** Password rejected by: contains user name fragments

Password: We34we56
!!! Password passed quality filter!!!

Password:
```

The FC feature (fix command) allows the user to edit the password:

```
Password: 123carl5
*** Password rejected by: contains user name fragments

Password: fc
Password: 123carl5
..........     e
Password: 123cael5

!!! Password passed quality filter!!!

Password:
```

The keyword **EXIT**, or the character combination **CTRL-Y**, terminates the interactive PWQASEEP process.

```
Password: exit
$GHS1 QUALITY 50>
```

The PWQASEEP program can be executed from the same program file, from which the SEE process (SEEP) is started.

Generating Passwords

The PWQASEEP-Program can be forced to generate passwords, which match the quality rules. This enables the user to get an idea, which type of passwords are accepted by the rules. To enter the GENERATE mode, the PWQASEEP-program has to be started with the keyword **GENERATE**, and it displays passwords, which have passed the quality check.:

```
$GHS1 QUALITY 41> pwqaseep generate
PWQASEEP (401) - T7172G06 - (21Feb2006) System \BEECH, running NSK G06
Copyright (c) GreenHouse Software & Consulting 2002-2006
Q-kh$IrQ
rvJ`e^Ma
$rmVeQ9X
auE(O#-@
9oACwOGr
&S(Nu;Hb
ZCeBe6lz
E*.QECVK
'oF3fulO
Q.a-<^t"
)p!Qj&X\
jXb:5vxF
LA|nm|rG
;vhfLW@C
pp`\fDaK
gL^#OL+n
GO4+}5I)
U*C8i}I{
|jT4Wmn/
w#<QdUax
zx%Q{T@;
More [Y/n]?
```

When the first 20 passwords are displayed, the user is prompted to either continue generating passwords, or to abort the generation process:

- Typing RETURN, or the letter "Y" and RETURN, causes PWQASEEP to generate the next set of passwords.
- Typing the letter "n" and RETURN, or the CTRL-Y key combination, terminates the password generation.

When the generation process is terminated, PWQASEEP displays some statistical information such as:

```
.
.
.
5VLsbaL@
y"(bD=6u
67tbx`Tj
=rIY9l)V
More [Y/n]? EOF!
Successful generated: 205
Failed generations:   43
$GHS1 QUALITY 20>
```

Successful generated:	Is the number of displayed passwords
Failed generations:	Is the number of generated passwords, which failed to pass the quality check.

Security Settings

The PWQASEEP program is started from the SAFEGUARD subsystem. Because SAFEGFUARD, as a system process, runs with the SUPER.SUPER-ID, all processes, started by SAFEGUARD, get this ID inherited.
This allows the following security settings of the PWQASEEP files:

- Owner = SUPER.SUPER
- GUARDIAN security (RWEP) for all files, except PWQASEEP = "OOOO"
- PWQASEEP security (RWEP) = "OOAO"

These security settings are set during installation time by executing the XSECURE TACL Macro, which is part of the PWQASEEP delivery:

```
?TACL Routine
============================================================================
==
==  PWQASEEP installation Routine XSECURE
==
==  This TACL routine is called from the XINSTALL TACL Routine.
==  It secures all PWQASEEO files to the best GUARDIAN security settings.
==
==  To successfully run this Routine
==   > it has to be executed in the volume where PWQASEEP resides
==   > the user has to be logged on as SUPER.SUPER, because PWQASEEP
==     needs to become licensed.
==
==  Copyright (c) GreenHouse Software & Consulting, 26Apr2002
============================================================================

#FRAME
#PUSH #INLINEPROCESS

INLPREFIX -
FUP/INLINE/

- ALLOW 1000 ERRORS
- ALLOW 1000 WARNINGS

#OUTPUT >>> XSECURE:   Securing the PWQASEEP environment

==
==  Give everything to SUPER.SUPER, and secure it to OOOO
==  PWQASEEP is secure for local access everybody: This allows
==  all local users to check a password before it is used.
==
- SECURE  *       ,OOOO
- SECURE  PWQASEEP,OOAO

- GIVE    *       ,SUPER.SUPER

INLEOF

#POP #INLINEPROCESS
#UNFRAME

========================================================= End >>> XSECURE ==
```

Log file

All PWQASEEP actions are recorded in a log file. This file resides in the same location as the PWQASEEP object file, and is named: PWQALOG0. In case it does not exists, it is automatically created.
When it runs full, it is automatically renamed from PWQALOG0 to PWQALOG1, and a new PWQALOG0 file is created.
In case PWQALOG1 runs full, it is renamed to PWQALOG2, PWQALOG0 is renamed to PWQALOG1 and a new PWQALOG0 is created.
This works OK until a file named PWQALOG9 exists. When its name is needed, it is purged, before the other files (PWQALOG0 .. PWQALOG8) are renamed.

The log file can easily be evaluated by using the delivered ENFORM query source file LISTLOG. To make it work, a dictionary is needed. To create a dictionary, follow these steps:

1. Volume over to the PWQASEEP location.
2. Create a dictionary by running the DDL compiler with this command:
   ```
   DDL/IN PWQADDL/NOLIST,DICT !
   ```
3. Run ENFORM with this command:
   ```
   ENFORM/IN LISTLOG/
   ```

The ENFORM LISTLOG source files looks like this:

```
!
! Example query to list all password change events, filtered
! through the PWQASEEP process.
!
! Adjust the 'AS' clauses below according to your needs.
! For details, please refer to the ENFORM Reference Manual.
!
! A password, that passed all checks, does not necessarily be the
! new password, because it can be rejected because
! - it is in the history buffer
! - it can not in the change period
! The reason for this is, that the SEEP process is contacted BEFORE
! the other attributes are checked.
!
! 28Nov2005, CW
!
open PWQALOG;

list EventTime           heading  "Password Check/Time"
     User       as a15   heading  "Password changed/for User"
     Manager    as a15   heading  "Password change/initiated by"
     Terminal   as a25   heading  "Terminal"
     IPAddress  as a15   heading  "Terminal/IP-Address"
     ProgFile   as a30   heading  "Change initiating/Resource"

     (If Outcome =  0 then "PW passed all checks"
     else
     (If Outcome =  1 then "Num special failed"
     else
     (If Outcome =  2 then "Num capital failed"
     else
     (If Outcome =  3 then "Num numeric failed"
     else
     (If Outcome =  4 then "Num alpha failed"
```

```
     else
     (If Outcome =  5 then "Max num multiple failed"
     else
     (If Outcome =  6 then "Template matc failed"
     else
     (If Outcome =  7 then "Dictionary match failed"
     else
     (If Outcome =  8 then "Password len to short"
     else
     (If Outcome =  9 then "Num multiple failed"
     else
     (If Outcome = 10 then "Lading or trailing blanks"
     else
     (If Outcome = 11 then "Password is part of users name"
     else
     (If Outcome = 12 then "Password is in GUESSES"
     else
     (If Outcome = 13 then "Num lower case failed"
     else "Unknown check outcome"))))))))))))
       heading "Password Change/Outcome"
;
```

A samle ENFORM run shows this result:

```
  Password Check    Password changed  Password change
       Time             for User       initiated by           Terminal
     Terminal            Change initiating               Password Change

    IP-Address               Resource                        Outcome


-----------------  ---------------  --------------  -------------------------
  ---------------  ------------------------------  ------------------------------
-

14Mar2006 14:55:54  SUPER.SUPER       SA.CARL          \BEECH.$ZTN00.#PTXMALC
  192.231.36.1     \BEECH.$SYSTEM.SYS27.SAFECOM    Dictionary match failed

05Apr2006 16:37:19  SA.CARL           SA.CARL          \BEECH.$ZTN01.#PTS2DAC
  192.231.36.1     \BEECH.$GHS1.PWMGMT.PWMGMT      PW passed all checks

06Apr2006 18:20:17  SA.CARL           SA.CARL          \BEECH.$ZTN01.#PT9WLPC
  192.231.36.1     \BEECH.$GHS1.SECOM600.SECOM     Password is in GUESSES

06Apr2006 18:25:09  SA.CARL           SA.CARL          \BEECH.$ZTN01.#PT9WLPC
  192.231.36.1     \BEECH.$GHS1.SECOM600.SECOM     PW passed all checks
```

Please change the query according to your needs.

PWQAHUB

PWQAHUB allows the system manager to define up to 1,000 different password quality rule files related to users and/or user groups.

```
!
!                       Password Quality Hub - PWQAHUB
!                       ==============================
!
! Defines the PWQARULE files in relation to the current user.
! This HUB file allows the password quality SEEP process to
! use different password quality rules based on a 'by user' basis.
! The number of entries is limited to 1,000.
!
! User
! GUARDIAN user names are NOT case sensitive.
! Alias user names are case sensitive.
! Both name formats support wildcards.
!
! PWQARule file
! Defines the password quality rule file that has to be used for the
! defined user.
! Always use fully qualified file names.
! File names are NOT case sensitive.
! File names do NOT support wildcards!
!
! This file is read by PWQASEEP at start-up time, and later on every
! time it is changed. The entries are sorted according to the MCO
! (most complete) rules.
!
!  User            PWQARule File
! ---------------------------------------
*                  $ghs1.quality.PWQARULE
sa.carl            $ghs1.quality.pwqacarl
Carlito            $ghs1.quality.pwqarule
sa.*               $ghs1.quality.pwqarule
```

Edit this file accoring to your needs.

PWQARULE

The behavior of the password quality SEEP program is configurable: Password quality features as well as data base settings can be configured in an EDIT type file named PWQARULE.
Edit this file and adjust it to your needs.

```
!
! This is a simple example of a RULES file, read by the
! Password Quality SEE Program PWQASEEP from GreenHouse.
! PWQASEEP Version 201 from 13Jul2004
!-------------------------------------------------------------------
!
! The password quality program supports two dictionaries for a
! collection of passwords, NOT allowed to be used:
! - DICTIONARY  defines all passwords by templates, supporting
!               the extended template characters. For more details,
!               please consult the documentation.
! - GUESSES     is a collection of some 2 million passwords, used to
!               run crack programs.
! Both files can be loaded with the LOADDICT utility, that comes along with
! the PWQASEEP product.
!
DICTIONARY   $dsmscm.pwqaseep.pwqadict
GUESSES      $dsmscm.pwqaseep.guesses

!
! The entries in the dictionary as well as guesses file can be treaded
! as case sensitive ON or OFF.
! Default is:  ON
!
DICTIONARYISCASESENSITIVE  OFF

!
! In case the PWQASEEP is running, it can kick off a password
! synchronization.
! To successfully control this process, a disk file for sync flags
! is needed by:
! - PASSYNC
! - PWQASEEP
! When this file is not defined, PWQASEEP does not initiate any
! synchronization.
!
SEEPSYNC  $dsmscm.pwqaseep.seepsync

!
! The SEEP can start a password synchronization, using the
! PASSYNC program from GreenHouse.
! To make SEEP aware of this, the file name of PASSYNC has to
! be defined here.
! When the entry is missing, no synchronization is started.
!
!**  PASSYNC  $dsmscm.newsync.passync

!
! When PWQASEEP is running in SEEP mode, it directs error messages
! to the EMS system.
! The system specific EMS collector process is addressed here.
```

```
! Default is: $0
!
EMSCOLLECTORPROCESS $0

!
! Num Special Characters
! This entry defines the minimum and maximum number of special
! characters to be used in the password. If this number is not
! reached, or exceeded, the password is rejected.
! Default MINNUMSPECIAL is 0
!         MAXNUMSPECIAL is 8
!
MINNUMSPECIAL 0
MAXNUMSPECIAL 8

!
! Num Capital Characters
! This entry defines the minimum and maximum number of capital
! characters to be used in the password. If this number is not
! reached, or exceeded, the password is rejected.
! Default MINNUMCAPITAL is 0
!         MAXNUMCAPITAL is 8
!
MINNUMCAPITAL 0
MAXNUMCAPITAL 8

!
! Num Lower Case Characters
! This entry defines the minimum and maximum number of lower case
! characters to be used in the password. If this number is not
! reached, or exceeded, the password is rejected.
! Default MINNUMLOWERCASE is 0
!         MAXNUMLOWERCASE is 8
!
MINNUMLOWERCASE 0
MAXNUMLOWERCASE 8

!
! Num Numeric Characters
! This entry defines the minimum and maximum number of numeric
! characters to be used in the password. If this number is not
! reached, or exceeded, the password is rejected.
! Default MINNUMNUMERIC is 0
!         MAXNUMNUMERIC is 8
!
MINNUMNUMERIC 0
MAXNUMNUMERIC 8

!
! Num Alpha Characters
! This entry defines the minimum and maximum number of numeric
! characters to be used in the password. If this number is not
! reached, or exceeded, the password is rejected.
! Default MINNUMALPHA is 0
!         MAXNUMALPHA is 8
!
MINNUMALPHA 0
MAXNUMALPHA 8

!
```

```
! NUMMULTIPLE defines the number of characters, which may occur
! 'side by side'.
! e.g. abba  = a = 1
!             b = 2
!
! Valid values are:
!      0 = no check
! 1 .. 7 = 1 to 7 characters side-by-side
!      8 = no check
!

! Default is: 0 = no check
!
NUMMULTIPLE 0

!
! Max number of duplicates
! This value defines the maximum number of duplicate occurrences of any
! character in a password.
! If it is set to 2 (this case), a password with a character that occurs
! more than 2 times, is rejected, e.g.
!  ABC   is fine
!  AAB   is OK
!  AAA   is rejected (3 A's)
!  AABA  is rejected (3 A's)
!
! Valid entries are 0 .. 8
! 0  = no check
! 1  = characters may only appear once in the password
! n  = number of max. occurrences of one character
! Default is 8
!
MAXNUMMULTIPLE

!
! The minimum length of a password still is controlled by SAFEGUARD.
! The smallest of both numbers defines the minimum password length.
! Default is 0
!
PASSWORDMINIMUMLEN  0

!
! Beside defining characters to be used, the use of a password
! structure can be enforced by defining templates.
!
! A template that is shorten than 8 bytes is filled up with asterisk
! characters.
! A template that is longer than 8 characters is truncated to 8 characters.
! The template structure is:
! TEMPLATE "nnnnnnnn"  = 8 byte template form of password, where n is:
!                        * = Any character
!                        1 = numeric character
!                        A = alphabetic Character
!                        S = Special character
!
! A password has to match ALL defined templates to qualify
!
TEMPLATE  ********

!
```

```
! PWQASEEP is able to check a password with the users name, and to
! reject it in case the users name is part of the password.
! The maximum allowed consecutive string, that might be similar
! to the users name, can be defined here.
!
! Valid values are:  0 = don't check the password with the users name
!                    1 = one   character  is  allowed to be identical
!                    2 = two   characters are allowed to be identical
!                    3 = three characters are allowed to be identical
!                    .
!                    .
!                    8 = eight characters are allowed to be identical
!
! Default is 0
!
USERNAMEINPASSWORD 0

! OBEY
! A rule file can call another rule file using the OEBY key word.
! This feature makes it easy to have e.g.
! – a system wide config file, where all general PWQASEEP attributes are
!   defined
! – a user specific password quality attribute file
! A stack of up to three OBEY recursions is supported.
! Any recursion that exceeds this number is rejected.
! The OBEY entry is optional.
!
OBEY  $vol.subvol.file

! NUMCASECHANGE
! A password may consist upper as well as lower case characters.
! The number of case changes can be configured with this attribute.
! Valid values are 0 .. 4
! 0      = no case change required
! 1 .. 4 = number of case changes
!
! Default is: 0
!
NUMCASECHANGE 0
```

Password Reject Messages

The software, which initiates a password change, creates its own reject message rather than using the one created and available by PWQASEEP.

Here is the list of different messages I could find:

PASSWORD program:

```
$GHS1 SECOM 56> run $system.sys00.password 12345678
*ERROR* Insufficient password quality
ABENDED: 1,140
$GHS1 SECOM 57>
```

SAFECOM:

```
$GHS1 SECOM 57>safecom alter user sa.carl,password 12345678
ERROR * Insufficient password quality
$GHS1 SECOM 58>
```

TACL at logon time:

```
$GHS1 SECOM 58> logon ghs.carl
Password:
Enter new password:
Reenter new password:
Unable to change password, access denied
$GHS1 SECOM 59>
```

Extended Wildcards

The commonly used wildcard characters on the Tandem|NSK system are:

*** (asterisk)**	defines any number (zero to n) of any type of characters
? (question mark)	defines exactly any ONE character

e.g.	$GHS?.CARL.TEST* **addresses the range of:**	
	$GHS?	$GHS ... $GHSZ
	CARL	**CARL is a complete name (no wildcards)**
	TEST*	**TEST … TESTZZZZ**

In summary: The wildcard above is everything between:

$GHS.CARL.TEST and **$GHSZ.CARL.TESTZZZZ**

The wildcard schema described above works quite well, but it does not differentiate between alphabetical and numerical characters. But this differentiation is sometimes very helpful!

To satisfy this requirement, GreenHouse extended the wildcard schema, and implemented it for all its tools and products.

The old wild card characters are still supported to allow the new schema to be upwards compatible:

* (asterisk)	**stands for any number of characters**
? (question mark)	**stands for one character**

These characters can appear anywhere in e.g. a string, or file name.

The new wildcard characters have to be surrounded by braces (opening brace = {, and closing brace = }). To make the usage as convenient as possible, the 'old' wildcard characters (* and **?) can be defined within the braces as well.**

These new wildcard template characters are available:

{*}	**(asterisk)**	or
{..*}	**(dot dot asterisk)**	stands for any number of characters (same as above)
{..A}	**(dot dot A)**	stands for any number of alphabetical characters
{..N}	**(dot dot N)**	stands for any number of numerical characters
{?}	**(question mark)**	stands for exactly one character
{A}	**(N)**	stands for exactly one alphabetic character
{N}	**(A)**	stands for exactly one numeric character

Alphabetical characters are defined as: a .. z and A .. Z
Numerical characters are defined as: 0 .. 9

Note: Template characters within the braces are NOT case sensitive!
The two 'old' wildcard characters (**? and** *) can appear inside, as well as outside, the braces, while
The new wildcard characters (**A, N, ..A, ..N**) MUST appear INSIDE the braces.

More than one extended wildcard character can be specified within the braces, e.g.

{..NAN..A} any number of numeric characters (**..n**), followed by one alphabetical character (**a**), followed by one numerical character (**n**) followed by any number of alphabetical characters (**..a**).

{AAA} three alphabetical characters (**aaa**)

Question marks and asterisks can be present within the braces as well:

{N?A..*}

To address all files, where at least one character is a number, use this wildcard:

{N}

To address all file names, beginning with the letter "G", followed by one or more numerical character, followed by the letter "A", use this wild card:

G{N..N}A

For more details, as well as a helpful text program, please consult the GreenHouse product CD.