Index

Reasons for EXCHANGE

Are you responsible for introducing new files to the system?
Do you have to upgrade existing software once in a while?
Do you struggle with all the rename, purge, FUP DUP etc. commands and possibilities to do the job?
And when finished, you have to reverse it all, because one of the files was the wrong one?

EXCHANGE addresses the specific problem of adequately protecting subvolumes for multiple uses. The security access required to INSTALL programs and files is different from the security access most appropriate for USAGE of such programs and files. A Security Administrator must either enable the broad INSTALL access at all times, or stand ready to change the access from USAGE to INSTALL (and back again) whenever a new version of the software is delivered.

EXCHANGE solves this problem by concentrating the INSTALL authority into a software module. The Security Administrator sets up appropriate USAGE access, and then delegates the authority overlay the objects involved through execute access to the program EXCHANGE. The person authorized to INSTALL new versions only needs the EXCHANGE program, not the broader authority to LICENSE, PURGE, or SECURE other files on the system.

Program Logic

EXCHANGE performs all steps necessary to exchange production files (already in place) with installation files (the ones that have to replace the production files).
It:

- saves the old/existing files (to allow a fall back)
- puts the new files in place (it finds out by itself, if a rename is OK, or if FUP is needed)
- keeps the security settings in place (GUARDIAN as well as SAFEGUARD)

Depending on the
- files locations
  = old and new files reside on the same volume: a rename is sufficient
  = old or new files reside on different volumes or systems: FUP DUP have to be performed
- file situation
  = old and new reside on the same volume: files can be open
  = old or new files reside on different volumes or systems: files have to be closed
- disk file ACL
  = has to be saved, and re-introduced
- GUARDIAN security vectors
  = have to be saved, and re-introduced

different methods are necessary to 'do the trick'.

EXCHANGE actions

EXCHANGE does all this for you.
It:

- takes care of all GUARDIAN security attributes
- takes care of all SAFEGUARD disk file ACLs when necessary
- checks if the files are allowed to be open
  (files residing on the same \system.$vol can be exchanged while being open)
- cleans up temporary files when necessary
- uses FUP for file duplication
- uses SAFECOM for ACL saving and setting

Technically, it does the following for each file pair:

- It first checks, if an exchange is necessary.
  In case the attributes
  = file type
  = file code
  = EOF
  = last modification time stamp
  are identical, the exchange is skipped.
- It then checks, it an exchange is possible.
  The EXCHANGE user has to have owner ship rights on both files. He has to be:
  = the file owner, or
  = then manager of the file owner, or
  SUPER.SUPER
  to successfully execute a file exchange.
  When this is not the case for the file pair, the exchange is skipped.
- All SAFEGUARD ACLs become removed.
- All GUARDIAN security settings become replaced by "UUUU".
- The files are exchanged.
- The original GUARDIAN security settings are replaced.
- The original SAFEGUARD security settings are replaced.
- All temporary files become purged.

Restrictions

EXCHANGE follows the GUARDIAN and SAFEGUARD rules.
The following restrictions apply:

- to successfully EXCHANGE a file, the user has to have ownership rights on both files.
  He has to be:
  = the file owner, or
  = the manager of the file owner, or
  = SUPER.SUPER
- AK-file pointers in ENSCRIBE type files are NOT touched: This has to be done manually, or through the ALTFILE freeware utility that can be found at: www.GreenHouse.de.
- the LICENSE attribute can only be inherited, when EXCHANGE is executed by SUPER.SUPER. A secure way in doing so is the use of SECOM (Secure Command Manager), performing command level security; for more information about SECOM please visit www.GreenHouse.de.
- EXCHANGEing files,
  = having an ACL on file level, when SAFEGUARD is NOT running, or
  = where the file owner has no owner ship access rights
  is rejected:

```
$GHS1 EXCHANGE 146> fi

$GHS1.EXCHANGE

               CODE               EOF   LAST MODIFIED  OWNER  RWEP    PExt    SExt
A                 0                 0 19JAN2000 15:40 100,5   ****       2       2
AA                0                 0 19JAN2000 16:09 100,5   UUOO       2       2
BB                0                 0 19JAN2000 16:08 100,5   UUOO       2       2
COMPILE         101              2888 19JAN2000 15:58 100,5   UUOO       2       2
EXCHANGE        700             39880 20JAN2000 15:09 100,5   UUOO      32      32
EXSRC           101             84212 20JAN2000 15:08 100,5   OO--       6       6
GETFILES        101              2796 20JAN2000 12:32 100,5   UUOO       2       2
OBEY            101                94 18JAN2000 16:29 100,5   UUOO       2       2
README          101             11466 20JAN2000 12:46 100,5   UUOO       6      28
SAVE            101               550 16JAN2000 17:00 100,5   OO--       2       2
Z                 0                 0 18JAN2000 17:38 100,5   ****       2       2
$GHS1 EXCHANGE 147> exchange a with aa
EXCHANGE (101) - T7172G06 - (20Jan2000)   System \BEECH
Copyright (c) GreenHouse Software & Consulting 2000
$GHS1.EXCHANGE.A  with  $GHS1.EXCHANGE.AA
^^^^^ - can't retrieve DISK FILE ACL
No files processed
$GHS1 EXCHANGE 148>
```

EXCHANGE usage

EXCHANGE is a non PRIV NSK based program of file type 700.
It can be used in:

- Batch Mode
- Obey Mode

Batch Mode

When all command relevant parameters are supplied through the start-up message, EXCHANGE runs in BATCH mode.

Command syntax is:

```
[run] EXCHANGE [/OUT [<file>]/] <old-files> WITH|, <new-files> [KEEPEX]
```

where

OUT <file>	File to which EXCHANGE has to send its messages. In case OUT does not exist, it becomes created as EDIT type file. When <file> is empty, no message is displayed.
<old-files>	File name template, defining the 'original' files to be replaced by <new-files>.
WITH or **,** (comma)	Required key word to make the command string readable.
<new-files>	File name template, defining the 'new' files, replacing <old-files>.
KEEPEX	Optional key word. When present prevents EXCHANGE from copying itself from <new-files> to <old-files>.

Obey Mode

Supplying the files to be exchanged through an EDIT type file, that is presented to EXCHNAGE as IN start-up parameters, causes EXCHANGE to run in OBEY mode.

Command syntax is:

`[run] EXCHANGE /OUT [<file>],IN <obey-file>/`

where

`OUT <file>`	File to which EXCHANGE has to send its messages. In case OUT does not exist, it becomes created as EDIT type file. When <file> is empty, no message is displayed.

`<obey-files>`	EDIT type file, holding a set of EXCHANGE commands.
	A typical obey-file looks like this:
	`!`
	`! Lines beginning with an exclamation mark, (!)`
	`!  a  double minus sign (--) or`
	`!  a  double equation sign (==)`
	`! are treated as comment lines, and NOT taken into account.`
	`!`
	`$dsmscm.secom.*  with $ghs1.secom.*  KEEPEX`
	`$dsmscm.secom600.SE??? with $ghs1.secom600.*`
	`! ---------- end of list ----`

HELP

When EXCHANGE is started with the -H or -HELP command string, it displays a help screen like the following one:

```
$GHS1 EXCHANGE 20> exchange -h
Command syntax is:
  EXCHANGE [/OUT [<file>]/] {<old> WITH|, <new>} [KEEPEX] | {-H[ELP]}
or
  EXCHANGE /IN <obey-file>[,OUT [<file>]]/

These wild cards for <old> and <new> are supported:
 $a.b.c with $d.e.f  =  $a.b.c <-> $d.e.f
 $a.b.c with $d.e.*  =  $a.b.c <-> $d.e.c
 $a.b.* with $d.e.f  =  $a.b.f <-> $d.e.f
 $a.b.* with $d.e.*  =  $a.b.* <-  $d.e.*
e.g.
  EXCHANGE $ghs1.secom.* with $ghs1.newsecom.*
  EXCHANGE $dsmscm.secom600.* with \sequoia.$dsv.secom600.*
  EXCHANGE/in doit,out/
$GHS1 EXCHANGE 21>
```

File pair constellations

The following combinations of old- and new-files, as well as wild card settings, are allowed.
An asterisk (*) defines any wild card pattern.
Valid wild card characters are:

- * (asterisk) defines zero or more characters
- ? (question mark) defines one character

EXCHANGE $a.b.c WITH $d.e.f

Both files are fully qualified.
When $d.e.f does not exists, an error is reported to the user, and this file specific exchange is aborted.
When $a.b.c does not exists, $d.e.f is put in place of $a.b.c with all its attributes.
When $a.b.f and $d.e.f exist: $a.b.c is replaced by $d.e.f, and $a.b.c is saved as $d.e.f

EXCHANGE $a.b.c WITH $d.e.*

The file to be replaced is fully qualified, the new file has to be taken from a given subvol.
The file that has to be EXCHANGEd is: $a.b.c with $d.e.c
When $d.e.c is missing, an error is reported to the user, and this file specific exchange is aborted.
When $a.b.c is missing, $d.e.c is put in place of $a.b.c with all its attributes.
When $a.b.c and $d.e.c exist: $a.b.c is replaced by $d.e.c, and $a.b.c is saved as $d.e.c

EXCHANGE $a.b.* WITH $d.e.f

The file to be replaced is defined by a template, the new file is fully qualified.
The file that has to be EXCHANGEd is: $a.b.f with $d.e.f
When $d.e.f is missing, an error is reported to the user, and this file specific exchange is aborted.
When $a.b.f is missing, $d.e.f is put in place of $a.b.f with all its attributes.
When $a.b.f and $d.e.f exist: $a.b.f is replaced by $d.e.f, and $a.b.f is saved as $d.e.f

EXCHANGE $a.b.* WITH $d.e.*

The file to be replaced is defined by a template, the new files are defined by a template.
The file that has to be EXCHANGEd is: $a.b.* with $d.e.*:

$a.b.a with $d.e.a
$a.b.b with $d.e.b
etc.

When $a.b.* is missing, $d.e.* is put in place of $a.b.* with all its attributes.
When $a.b.* and $d.e.* exist: $a.b.* is replaced by $d.e.*, and $a.b.* is saved as $d.e.*

Run time actions

- A temporary location is used as 'parking lot' for the files that have to be replaced.
  This location resides on the volume, on which the files to be replaced reside (old files).
  The location name is composed of the three hard coded letters: GHS, and 5 characters from a system generated process name.
- When the files to be exchanged reside on the same volume, EXCHANGE uses a RENAME mechanism for performance reasons.
- When the files to be exchanged reside on different volumes, FUP is started to perform the necessary DUP operations.
- In case the file to be replaced has a file level ACL, EXCHANGE starts a SAFECOM process.
- In case the new file, that replaces the old one has a file level ACL, EXCHANGE starts a second SAFECOM process.
- In summary, an EXCHANGE process may have three processes created:
  One FUP, max. two SAFECOM

Installation

EXCHANGE is a stand alone pTAL program. It does NOT require any subsystem to be up and running.

To install EXCHANGE, perform these steps:

- transfer the file EXCHANGE.700 in binary mode onto the Tandem system, and name the file e.g. EXCHANGE
- change the file code of EXCHANGE to 700 (**`FUP ALTER EXCHANGE,CODE 700`**)
- set the owner, e.g. to SEC.ADMIN (**`FUP GIVE EXCHANGE,SEC.ADMIN`**)
- secure EXCHANGE to "OONO" (**`FUP SECURE EXCHANGE,OONO)`**
- optionally add a SAFEGUARD ACL (**`SAFECOM ADD DISKFILE <file>...)`**

Preferred security settings

The most stringent security settings are:

Owner	GUARDIAN Security	License
Security Adminsitrator	"OONO"	N/A

Run time specialty

EXCHANGE needs to keep the file, that has to be replaced, in a temporary location until it can be moved to the place, that was held by the new file.
This is done by using a subvol on the volume, on which the files, that have to be replaced, exist.

e.g. the command:

```
EXCHAGE $ghs1.secom600.se*  with  $ghs2.newsecom.*
```

causes EXCHANGE to use a subvol on **`$ghs1`**. The subvol name is generated at run time, and the naming convention is:

```
$vol.GHSxxxxxx
```

where

`$vol`	is the location of the 'old' files
`GHS`	is a hard coded part of the name
`xxxxx`	is a system generated name (generated process name without the leading $ sign)

Example

```
EXCHANGE $system.secom603.* with $ghs1.secomupd.*
```

This command updates an existing SECOM environment in **$system.secom603** with new files from **$ghs1.secomupd**.

- Files in **$ghs1.secomupd** matching files in **$system.secom603** are exchanged.
- Files in **$ghs1.secomupd**, NOT matching any file in **$system.secom603**, are moved to **$system.secom603**.

When EXCHANGE is done, the old files from **$sytem.secom603** reside in **$ghs1.secomupd**.

Hints

EXCHANGEing open files

To EXCHANGE open files, the new files have to reside on the same **\system.$vol** as the open files that have to be replaced!

PROGIDing EXCHANGE

Giving EXCHANGE to SUPER.SUPER, and setting the PROGID flag, is a security breach: The user of EXCHANGE can replace licensed programs with his own ones, and introduce dangerous code to the system!

In case ...

... you run into problems, please let me know (Info@GreenHouse.de), and I'll fix it!