162 VM update

February 2000

In this issue

VM Update

Published by

Software Diversified Services (SDS)
5155 East River Road
Minneapolis, MN 55421-1025
USA
www.sdsusa.com
sales@sdsusa.com
support@sdsusa.com
voice 612-571-9000
fax 612-572-1721

SDS became the publisher of VM Update with the January 2000 issue. Prior to that, it was published by Xephon plc.

Editor

Rick Kothe
rickkothe@sdsusa.com
612-571-9000 ext. 121

File formats

VM Update is published in pdf format, to be read with an Adobe® Acrobat® Reader. The Reader is available free of charge at www.adobe.com. Once the Reader is installed, Netscape and Microsoft browsers can display pdf files in browser windows.

Beginning with January 2000, individual articles will also be available in html so that readers can more readily copy the codes.

Free subscription, back issues

VM Update is free of charge at www.sdsusa.com. At that site, SDS provides back issues through January 1997. Parts of older issues are available at www.xephon. com/archives/vmi.htm.

Contributions

SDS and *VM Update* welcome contributions. The rate of payment varies. For information, contact the editor.

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither SDS nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither SDS nor the contributing organizations or individuals accept any liability of any kind whatsoever arising out of the use of such material.

Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, EXECs, and other contents of this journal before making any use of it.

SFS Authorities

With this text we try to help you to better understand the sharing facilities of SFS. We know from experience that beginners have some trouble understanding the philosophy behind SFS authorities. Once this concept is clear, understanding the different ways one can share files will be a piece of cake.

Permissions in SFS

Access to a file stored in SFS is possible only if you have the authority to do so. Sharing files requires multiple users to get the needed authorities. Authorization can be implicit (for your own files). But for letting other users access your files, explicit authorizations have to be given using SFS commands. The authorization requirements may seem somewhat complex. This is mainly because SFS directories must remain compatible with minidisks as much as possible. Since its introduction in CMS Release 6 (1979), SFS got several enhancements, but backward compatibility with the first version is mandatory too, sometimes causing extra confusion. To help you understand some "reasons why", we'll start explaining the original CMS6 status, where appropriate we'll explain the enhancements. SFS has its own 'built-in' security manager, which we will explain in the first place. But SFS can also work with an External Security Manager (ESM) such as RACF/VM. We'll give some views of how an ESM can change things throughout the article.

File Organization, Directories

First of all you must understand that SFS organizes files in directories. We can distinguish File Control Directories and Directory Control Directories. The latter ones are discussed later as a special case.

The Top directory is automatically created when the SFS administrator enrolls a user in the SFS filepool. Sub-directories are created with the CREATE DIRECTORY command. Examples:

```
IECSYSU:BUELENSC.  is a top directory
IECSYSU:BUELENSC.TOOLS is a sub-directory
```

SFS files are stored in an SFS directory, and the space a user has in the filepool is called a filespace. The end-user himself can use GRANT AUTHORITY and REVOKE AUTHORITY commands to control who gets access to the files in his filespace. To give the maximum granularity, the user can grant authority on a file by file base.

Note: The SFS Administrator has all the authorities to create, grant and store objects in any user's filespace, but in this article we mainly discuss the possibilities offered to the end-user himself.

Where SFS runs without an ESM, it must be clear that only the owner of the filespace (and the SFS administrator) can use the GRANT or REVOKE commands.

At the end of the article, an exec is shown that can help bypassing this restriction (with the help of a service machine such as PROP).

Using SFS files

Before being able to fully understand the SFS authority design, you must also understand how SFS files can be used. Even today where SFS exists after about 10 years, SFS files are mostly used just as if they reside on a regular minidisk. This means, before an SFS file is used, the CMS ACCESS command is used to access the directory hosting the file. For example:

```
ACCESS IECSYSU:BUELENSC.TOOLS C
```

Once the directory is accessed, most CMS commands and applications won't notice any difference between SFS files and minidisk files. It is however possible to refer to an SFS file without accessing the directory first. This is possible for any program or REXX procedure by using CSL calls where the directory name replaces the filemode. CMS Pipelines, for example, is enhanced since VM/ESA Version 2.1.0 to allow this "direct access".

For example:

```
PIPE < PRFGUI VARS IECSYSU:BUELENSC.TOOLS |FIND .....
```

Authority Structure

In SFS two sorts of objects can be distinguished: files and directories.

Authority for Files

To read a file, one needs READ authority to that file. By default, the filespace owner and an SFS administrator have write authority, other users have no authority at all. Updating a file requires WRITE authority. The commands to grant or revoke authorities are:

```
GRAnt AUTHority fn ft dirid TO userid ( READ | WRITE
REVoke AUTHority fn ft dirid FROM userid ( options
```

The above GRANT command is enough to authorize someone to use a file. But, as told above, SFS files are usually used after ACCESSing the directory, and to SFS, the files and the directory are two different objects.

Remember: Giving some authority to a file will never imply authority to the directory, or vice versa.

Authority for Directories

Understanding authorities on SFS directories is easier if one sees a directory as "a file containing the names of other files". ACCESSing a directory means reading it, hence at least READ authority to that directory is required.

The commands to manipulate authorities to directories are:

```
GRAnt AUTHority dirid TO userid ( READ ] WRITE
REVoke AUTHority dirid FROM userid ( options
```

Storing a new file in a directory requires an update of this directory, so WRITE authority to the directory is required in that case.

Note: When you create a file in another user's filespace, you automatically get write authority to that file too. However, you do not become the owner of the file, which means you can not give grants to other

users, only the filespace owner and SFS administrators can (see the GRANTAUT EXEC that is appended below for a bypass). So, to be able to create a new file in another user's directory, one needs write authority to that directory (as the fileid has to be written to the directory). If you have write authority to a file, it is also possible to ERASE it. But, if the file resides in another user's filespace, the fileid has to be removed from the directory as well, which requires write authority to the directory too.

You may now conclude that if you want to use a file of someone else, read access to the directory is almost always required. Hence, SFS seems poorly designed. However, there is yet another possibility: you can create an ALIAS for the file in one of your own directories. For example, after issuing:

```
CREATE ALIAS
BUELENSC NAMES IECSYSU BUELENSC.
KRIS = IECSYSU DECEULAE.
```

the file BUELENSC NAMES can be used as if it where a file in DECEULAE's top directory and DECEULAE no longer needs to access BUELENSC's directory.

Granting access to many files

To make working with many files easier, the GRANT and REVOKE commands allow the LISTFILE wildcard characters. For example:

```
GRANT AUTH * * IECSYSU:BUELENSC.TOOLS TO DECEULAE
```

gives read permission to all files in BUELENSC's TOOLS directory. But you have to understand that the above generic GRANT command will be translated by the system into distinct grants to each individual file residing in the directory at that precise moment. Consequently, a file created at a later time by user BUELENSC will not be granted to DECEULAE.

Here an ESM can have benefits: RACF/VM for example can store the generic permission.

Enhancements after CMS Rel 6

CMS Rel 7 of VM/ESA 1.1.0 added the NEWREAD and NEWWRITE options to the GRANT command. These options are valid only for a GRANT on a dirid. With these options, SFS will give READ and/or WRITE authority to any new file created in that directory. So, to give user DECEULAE read authority to any file that will be created in the future, user BUELENSC can issue the command:

```
GRANT AUTH BUELENSC.TOOLS TO DECEULAE (NEWREAD
```

The NEWREAD or NEWWRITE options are complementary to the READ and WRITE options for directories. This means that the above command gives user DECEULAE the authority to read files created later, but does not imply that he can read any currently existing file.

We've seen that SFS authority to directories and files are completely different things. The NEWREAD and NEWWRITE options cause some confusion to many people: you specify them on a grant command for a directory, but the grants will later be given automatically to files.

Another thing to remember: the NEWREAD and NEWWRITE attributes are only checked when a file is initially created. An example, suppose that user BUELENSC creates a file in BUELENSC.TESTTOOLS. Later he uses RELOCATE to move the file into BUELENSC.TOOLS, now user DECEULAE will not automatically get read authority to it. Similar, if RELOCATE is used to move a file out BUELENSC.TOOLS, DECEULAE will not loose his authority. This is easier understood when knowing that file authorities are attributes connected to the file record in the SFS catalog.

Granting to many users

Another feature of SFS allows to grant authorities to a group of users. The group of users is defined as nickname referring to a list of names in the user's NAMES file. So, a GRANT TO nickname will be resolved by CMS to grant the authority to all users referred to by the nickname. But again, SFS will store a distinct grant attribute for each user defined by the nickname. So, when the nickname is changed, the SFS permissions are not adapted.

Here an ESM can have extra benefits: RACF/VM for example knows user groups, and a permission can be given to the group; when users are added to the group they get the permission too. Looking closer, one can say that RACF/VM stores the permission requests, whereas SFS always stores the resolved permissions.

Compatibility Issue: Q DISK Response

CMS has a table with information for all accessed filemodes, this table is called ADT, Active Disk Table. The ADT contains two flags, ADTFRO and ADTFRW, telling if the minidisk is R/O or R/W. The QUERY DISK, QUERY SEARCH and QUERY ACCESSED commands all show this flag. With minidisks, life was simple: by checking the QUERY DISK output (or the ADTFRx flag in the ADT), a program could easily check if it could write to any file on that minidisk. When an SFS directory of someone else is ACCESSed, by default the flag is set to R/O, even if you are granted write access to the directory. Why?

Let's consider one case: compilers. Compilers, typically tend to create work files on the largest R/W disk. If the answer to the QUERY DISK MAX would be the filemode of the accessed SFS directory, the work files would be created in that directory. But, this would mean that another user also having write authority to that same directory could have trouble compiling at the same time as the work files would be open by the first user.

You still can argue that showing R/O is silly for an accessed dirid to which you have write authority and write authority to all files in it. But don't forget that the dirid owner could decide to dynamically revoke your write authority of some or even all the files. Many programs would be upset when finding out that writing to the files on a filemode appearing R/W fails and decide to abend. When working with SFS files, new programs must check the authority on file basis.

In REXX the CSL routine DMSQFMOD and the QUERY AUTH command is very useful for that purpose. PIPE's STATEW stage in recent CMS levels can be used too to check if you have R/W authority, but only when the dirid is accessed with the FORCERW option, else STATEW sets return code 28.

The CMS STATEW command still only checks the RW flag in the ADT.

Remember: By default, accessed directories other than your own appear R/O, irrespective of the authorities you have.

With a PTF in CMS Rel 6, the ACCESS command got the FORCERO and FORCERW options. These FORCERx options do set the R/O or R/W flags in the ADT. They can be used irrespective of the authorities you have. A few examples:

```
* running in user DECEULAE
ACCESS DECEULAE. A
ACCESS BUELENSC.RWGRANTED G
ACCESS BUELENSC.ROGRANTED H
ACCESS BUELENSC.RWGRANTED2 I (FORCERO
ACCESS BUELENSC.ROGRANTED2 J (FORCERW
QUERY ACCESSED
IECSYSU:DECEULAE.
IECSYSU:BUELENSC.RWGRANTED
IECSYSU:BUELENSC.ROGRANTED
IECSYSU:BUELENSC.RWGRANTED2
IECSYSU:BUELENSC.ROGRANTED2
```

To find the authority on a file or dirid, one can use the QUERY AUTH command, or use the more official CSL routines.

CMS Commands

As it was not possible to immediately rewrite all CMS commands (and user applications) to fully support SFS, most CMS commands do not allow a dirid in place of a filemode. Similar, most CMS commands refuse to write to filemodes with the ADT flag set to R/O.

In CMS Rel 6 only 4 existing commands where adapted regarding the SFS authorities:

XEDIT was enhanced to properly check SFS file authorities (and to LOCK the file during the XEDIT session). XEDIT no longer uses the ADTFRO flag. This gives a surprising result: even if Q DISK tells a directory is accessed R/O, it may be possible to update an SFS file.

COPYFILE similar to XEDIT, SFS authorities are checked.

ERASE and RENAME accept a dirid in place of a filemode. But again there are some surprises: suppose DECEULAE accessed BUELENSC.TOOLS as B-disk without using the FORCERW option. So the ADTFRO flag is set, defining B as R/O.

`ERASE A FILE B` will fail; with an fm, ERASE finds ADTFRO.
`ERASE A FILE BUELENSC.TOOLS` works; with a dirid, ERASE checks SFS authorizations.

`*** RENAME` works just like `ERASE`.

From these examples we can learn how to use the ERASE and RENAME commands from within FILELIST. Issuing ERASE or RENAME against a file, without parameters, lets EXECUTE (which is FILELIST's command processor) append the file-id, and hence the command will fail (case 1).

If however you add the symbols /ntd, and write ERASE /NTD then EXECUTE will append the filename, filetype and the dirid, which lets the command execute properly (case 2). Similarly, you should use RENAME /ntd NEWNAME = =

Note: you can issue HELP FILELIST or HELP DIRLIST to find all the available / symbols and how youi can use them in FILELIST and DIRLIST

Enhancements after CMS Rel 6

Since CMS Rel 9 (VM/ESA 1.2.0), FILELIST accepts a dirid in place of a filemode, and the READWRITE and READONLY options where added to FILELIST and DIRLIST. The READxxxx options trigger the corresponding FORCERx option when the directory is dynamically accessed. Remember that the DEFAULTS command can be used to set your own defaults for FILELIST and DIRLIST. An example:

```
DEFAULT SET FILELIST READWRITE
```

CMS Rel 11 added yet another option: SET RORESPECT ON. With minidisks one could protect files on minidisks against accidental erase or update by either LINKing in RR, or issuing ACCESS cuu fm/fm. With SFS, one could no longer protect files against accidents with XEDIT or COPYFILE. However, if you issue SET RORESPECT ON,

both XEDIT and COPYFILE refuse updating a file in a directory accessed R/O (so they use the ADT flag again).

Test if file exists: STATE, STATEW, LISTFILE, FILELIST

As authorities are on file basis, it is well possible that an accessed directory contains files to which you do not even have READ authority. This leads to a little dilemma: you can see their fileid's (because you have read authority to the directory), but you cannot edit nor execute them.

`STATE` works as if the file does not exist (i.e. yields RC=28).

STATEW similar to STATE, and it still uses the ADTFRW to check for R/W. Hence STATEW can not be used at all to check for R/W authority.

LISTFILE has a set of options: ALLFILE or AUTHFILE. The default is AUTHFILE, which means LISTFILE gives returncode 28 for a file to which you don't have READ authority.

The three commands above work that way as they are mainly used in EXECs, and another behavior could lead to failures at execution time.

FILELIST is an EXEC that internally calls LISTFILE. But FILELIST uses, by default, the ALLFILE option. Which means that by default FILELIST shows all files existing in a directory. Surprise: pressing PF11 against such a file, starts an XEDIT session with an empty file... Above, when wrote that when working with SFS files, new programs should check the authority on file basis. In REXX, the CSL routine DMSQFMOD and the QUERY AUTH command are very useful for that purpose.

PIPE's STATEW stage in recent CMS levels can be used too to check if you have R/W authority, but only when the dirid is accessed with the FORCERW option, else STATEW thinks you only have R/O authority and sets return code 28.

The CMS STATEW command still only checks the RW flag in ADT.

Here is a "ready-to-use" example extracted from one of my execs:

```
Address command
```

```
DirFile='USERS DIRECTRY'
'PIPE (end ?) Literal' DirFile,
 /* get complete fileid */
 '!STATE',
 '!SPEC W1 1 W2 nw W3 nw.1',
 '!VAR DirectFid'
 /* get complete fileid */
if symbol('DirectFid')<>'VAR' then
call ErrExit rc,'File' DirFile 'not found'

/* Use the completed fileid of the file */
parse var directFid DirFile 0 . . tfm .
/* Let's check if the file is in SFS or not */
call Csl 'DMSQFMOD retc reason tfm buffer flag'
UsingSfs=(flag<>3 & flag<>4)
/* Note: Variable "BUFFER" contains the full dirid,
but we don't need it in this program */

If UsingSfs then do
'PIPE COMMAND QUERY AUTH' DirFile,
'!DROP 2',
'!LOCATE 30.8 /'left(BaseUserId,8)'/',
'!SPEC W7 1!APPEND LITERAL 0!VAR T'
CanWrite= (t='X') /* Yes, he has R/W auth */
else do /* Not using SFS: use STATEW to check */
'STATEW' DirFile
CanWrite=(rc=0)
end
```

The code above works fine to check if you have R/W authority to a file.

There is yet another case to consider: a server might have to check the authority of a user submitting a request to it.

The code above will not reveal that SFS administrators have R/W access too.

Here is something that helps:

```
/* Let's check if the file is in SFS or not */
call Csl 'DMSQFMOD retc reason tfm buffer flag'
UsingSfs=(flag<>3 & flag<>4)
CanWrite=0
If UsingSfs then do
FullDirid=strip(buffer)
```

```
'PIPE COMMAND QUERY AUTH' DirFile,
'!DROP 2',
'!LOCATE 30.8 /'left(FromUser,8)'/',
'!SPEC W7 1!APPEND LITERAL 0!VAR T'
if t='X' then CanWrite=1
/* Yes, he has R/W auth */
Else do /* Maybe he is an SFS ADMIN */
'PIPE LITERAL' fullDirid,
'!SPEC /QUERY ENROLL ADMIN FOR' FromUser'/ 1',
'FS : F1 NW /:/ N',
/* Extract filepoolid */
'!COMMAND!DROP!SPEC W2!VAR t'
if rc=0 then
if t='YES' then CanWrite=1
/* Yes, he has R/W auth */
```

Directory Control Directories

This new type of directory was introduced with VM/ESA 1.1.0. The default directory type is since then called File Control Directory, where you have full control on file levels as explained above. Directory Control Directories are controlled on directory level, very much like a minidisk. They are meant as storage for files that are rarely updated, yielding better performance. This type of directory will be shown as DIRC instead of DIR in the output of QUERY DISK.

You give a GRANT on directory level, and all the files at once. Only one user at a time can have a "Directory Control Directory" accessed in R/W mode. The commands related to these directories are:

```
CREate DIRectory dirid (DIRC
DIRATtr dirid (DIRC
GRAnt AUTH dirid TO userid (DIRREAD ! DIRWRITE
REVoke AUTH dirid TO userid (KEEPDIRREAD
```

The FORCERW or FORCERO options on ACCESS are very useful here too. Even more, if you don't need to update one of your own Directory Control Directories, you'd better always use the FORCERO option, for these 2 reasons:

1. When accessed as R/W you surely stop other, authorized, users from updates, and
2. When the directory is dataspace eligible, the dataspace is only used by users accessing in R/O.

Becoming an SFS Administrator

To become an SFS administrator three methods exist:

1. Edit the "poolid DMSPARMS" file on the SFS server's 191 minidisk, and include your userid on an ADMIN card. Restart the SFS server. This way you'll allways be an SFS administrator
2. Find another SFS administrator and persuade him to issue "ENDROLL ADMIN youruser filepool", and you'll be SFS Admin until the SFS server is restarted.
3. Logon to the SFS server and issue "GRANT ADMIN TO youruser" or, maybe a bit easier, issue "CP SET SECUSER sfsserver youruser" and "CP SEND sfsserver GRANT ADMIN TO youruser"

When I started using SFS, I decided that I would not be an SFS administrator all the time, just to learn how "normal" CMS users can work with SFS, what limitations they'll feel etc. But, being a VM systems programmer sometimes I do need to become SFS administrator so that I can help the other SFS users. Therefore, we wrote another PROP routine by which, users authorized in PROP, can enroll themselves as SFS administrators.

The SFSADMIN exec is included below.

Strange Problem

An SFS Admin can't read files. We have been told that an SFS administrator has access to any file stored in the SFS. An ESM can change that, but, we're discussing a case without an ESM.

Here is a list of events that will make the problem appear:

1. You need an SFS administrator userid and a normal CMS user.
2. Log on to an SFS administrator userid, let's call him "SFSADMIN".
3. Create a directory, place some files in it and issue "GRANT AUTH dirid TO cmsuser".
4. Give the end-user read access to the directory, but not to the files stored in it.
5. Logon with the "normal" CMS user that is not an SFS administrator.
6. Have this user access this directory by issueing "FILELIST * * dirid". Try to XEDIT a file stored in the directory. As mentioned before, XEDIT will show an empty screen as the end-user has no read authority to the file.

Now you say, "I understand, I've been reading this article and I do know that R/O authority on a directory does not imply I have it on all the files."

Let's solve this quickly, I'll make myself an SFS administrator.

So: Return to the "SFSADMIN" user to enroll user "cmsuser" as SFS admin. But do *not* LOGOFF from user "cmsuser"; if you need the 3270 session, issue DISCONNECT in "cmsuser".

Return to "cmsuser" and verify that "cmsuser" is now indeed an SFS admin by issuing Q ENROLL ADMIN.

Then, try to XEDIT the file you couldn't XEDIT before. It still fails.

Why?

Well, when you ACCESS an SFS directory, CMS obtains much information from the SFS server about the files in that directory.

This information is basically what LISTFILE and FILELIST show you: filelid, recfm, modification date.

But it also includes the authority you have on the files: none, read or write.

When someone issues a GRANT AUTH to change the authority of the files, the SFS server broadcasts the new authority to all CMS users that did ACCESS the directory.

But, when you suddenly become an SFS administrator, the SFS servers does not broadcast the new authority, and CMS in your machine still thinks you have not even read authority.

How to solve the problem?

Your first attempt will probably be: "RELEASE dirid" and "ACCESS dirid".

But, this doesn't help at all in most cases.

You discover yet another feature to make SFS perform well: even when you RELEASE a dirid, CMS still keeps the information of the files available in your machine, but they are hidden.

Only when CMS desperately needs more virtual storage, it will clear the storage occupied by these hidden files.

Solution: an IPL CMS or the SFSDISC EXEC

Sometimes, you don't like to re-IPL CMS, maybe because you accessed quite a few SFS directories and after an IPL CMS you have to re-issue all these ACCESS commands again.

The SFSDISC EXEC helps you here. It uses a CSL call that disconnects you from *all* SFS servers you are connected too.

So, you don't need an IPL CMS.

But, SFSDISC is even better than that: by default, after the disconnect, it will re-ACCESS the SFS directories you had accessed before.

The EXEC is included below.

Conclusions and notes:

Don't use Q DISK, Q SEARCH nor Q ACCESSED to check if you can write to an SFS File Control Directrory, but use Q AUTH.

To give a "complete" read access to files in a file control directory, in practice this means issuing:

1. GRANT dirid TO userid this allows ACCESS dirid
2. GRANT * * dirid TO userid give read permission for all existing files
3. GRANT dirid TO userid (NEWREAD give read permission for all new files

The first and last command can be combined in one command.

We separated them to make you remember that the first gives a permission to a directory, whereas the latter is for files.

Each GRANT to a file/dirid is a kind of record in the SFS catalog. But, you can for example also GRANT TO PUBLIC: this is simply a flag in the file/dirid record.

Use FORCERO for Directory Control Directories you don't need in update mode.

Only the filespace owner and SFS administrators can issue GRANT commands, CREATE DIR, and ERASE dirid. If you need more, an ESM can help (the ESM is called for commands as well), or create a service machine that is an SFS administrator (have a look at the GRANTAUT EXEC that follows below).

Useful EXECs:

Here are the execs we promised:

GRANTAUT EXEC

This exec has been written to act as a PROP routine, but, a simple change to the way the input parameters are analyzed can make the exec ready to use in other service machines.

```
/* This exec is a PROP action routine. It helps to enlarge the GRANT
AUTH
 command scope:
 - If user A grants write auth to user B & C on his SFS directory DIR
 ==> user B can successfully create a file ZZ ZZ on dir SFSxx:A.DIR
 - If now user B wants to GRANT AUTH ZZ ZZ TO USER C, it will be
 refused by SFS (only the directory owner (and SFS admin) can issue
 GRANT commands for the files in that directory).
 (SFS misses the SQL possiblity of "GRANT with GRANT option")
 (Even a GRANT AUTH * * DIR TO C issued by user A is not sufficient
 as this grant will be given for currently existing files only).
 Solution:
 Install an ESM as RACF 1.10
 - install this exec in PROP (the user running PROP must be defined
 SFS administrator)
- add an entry to your PROP RTABLE to call this exec
- issue:
Format:
 +-------------------------------------------------------------+
 ! MSG prop GRANT AUTH fn ft dirid TO userid!nickname (READ!WRITE !
 +-------------------------------------------------------------+
 Note that the "dirid" must be complete and thus contain the filepoolid

Format if you want to use this possiblity from an EXEC:
 +-----------------------------------------------------------+
 ! format:
 MSG prop GRANT AUTH ........ [ ; SMSG ! OK ]
 +-----------------------------------------------------------+
 (the operands for the CMS GRANT AUTH command remain the same)

If you specify anything behind the ';' you indicate that you want a
confirmation msg from the GRANTAUTH EXEC when all is OK
If you specify SMSG, this exec will send all its message (errormsg
or confirmation message) as SMSG

Written by: Kris Buelens IBM Belgium;
```

```
BUELENSC at IECVM
15 Feb 1990*/
trace n
parse arg user node lgloper lglnode type propu propn rscs rtable
parse pull msgtext 0 . . fid ' TO ' to ';' ans
parse pull action
address command

'MAKEBUF'
if verify(fid,'*%','M')^=0 ! words(fid)^=3 then
 call tell 'Invalid fileid:' fid
'QUERY AUTH' fid '(STACK'
if rc^=0 then call tell 'Can''t find your authorisation on file:' fid
parse pull .
do queued()
 parse pull . . . . uid read write .
 if uid=user then do
 if write^='X' then call tell 'You need yourself WRITE authority to pass
the GRANT to others'
'GRANT AUTH' fid 'TO' to
if rc>4 then call tell 'Returncode' rc 'on GRANT AUTH' fid 'TO' to
if ans^='' then call tell 'OK'
signal exit
end
end
call tell 'You need at least WRITE authority to pass the GRANT to
others'
EXIT: 'DROPBUF'
exit


TELL:
if ans='SMSG' then cmd='SMSG'
 else cmd='MSGNOH'
'CP' cmd user arg(1)
'DROPBUF'
```

SFSADMIN EXEC:

This exec can be used to enroll yourself as administrator via PROP /* This exec allows to enroll you as SFS admin

```
(if you are authorized in PROP and if PROP is SFS admin)
+-----------------------------------------------------------+
! format:
! SFSADMIN <filepoolid>
+-----------------------------------------------------------+
Written by: Kris Buelens IBM Belgium;
BUELENS at DIEVMD 9 Mar 1992*/
/* 10 Jun 1993: works now also if not enrolled as end-user in "pool" */
parse upper source . . myname mytype . syn .
address command

parse upper arg pool .
parse value diag(8,'Q USERID') with userid . node '15'x

if pool='' then do
'QUERY FILEPOOL CURRENT(LIFO'
parse pull . pool .
if pool='NONE' then pool=''
end
if pool='' then do
if node='VMKBBR01' then pool='SFS72:'
else pool='SFSD:'
end

else if right(pool,1)^=':' then pool=pool':'


'PIPE COMMAND CMDCALL QUERY ENROLL ADMIN FOR' userid pool ,
 '!VAR emsg!DROP 1!VAR qenr'
if rc=99 then do
say emsg
exit rc /* 99 = not existing filepool */
end
parse var qenr . yesno .
if yesno='YES' then do
say 'You are already SFSadmin for pool' pool
exit 0
end

'CP SMSG PROP CMD ENROLL ADMIN' userid pool
exit rc
```

SFSDISC EXEC

This exec can be used to disconnect from all SFS servers, maybe after that you made yourself SFS administrator and need to refresh authorizations on files you accessed before.

```
 /* This exec will force a complete disconnect from ALL SFS servers

Drastic for sure, BUT, it helps for two occasions:
 1. When you enroll yourself as SFS admin (e.g. via SFSADMIN) CMS does
not update your access permissions of all dirs you have accessed (or had
accessed a while before).
So even though an SFS admin is allowed to do anything in SFS, files in
such directories could still remain inaccessible after that you become
SFS admin. (RELEASE the dirids is not enough: after a RELEASE, CMS still
keeps the dirid information available in the background (for better
performance).  CMS really releases this information only when your
virtual storage consumption approaches 100%, or when the connection
with the SFS server is broken).
2. CMS allows a connection to an SFS server only with one name.  Example
if you first connect to VMSERVD using poolid SFSD (a nick- name) you can
later not go to VMSERVD with poolid SFS71 (the real poolid).

CMS displays:

DMSJCD2524E Concurrent use of multiple file pool identifiers
DMSJCD2524E that resolve to file pool SFS71
As the SFSDISC exec will re-ACCESS all dirs it finds as ACCESSed, be
sure to release dirs with the "wrong" poolid before invoking SFSDISC

+-----------------------------------------------------------+
! format:
! SFSDISC
<FORCE> <NOreaccess>
<QUIET>
+-----------------------------------------------------------+

FORCE:

To also issue the disconnect when one or more workunits active (so this
can cause rollback of not committed updates)

NOREACCESS then SFSDISC will not ACCESS again the dirs it found
accessed before we disconnected from all SFS-ses
```

```
QUIET then SFSDISC works without prompts or console (unless errors)

13 Jul 1998: RE-access dirs found when reply = YES
11 Aug 1998: add NOREACCESS option
11 Aug 1998: avoid prbs with "FORCE", a REXX variable found by CSL
9 Oct 1998: allow NOREACC and change poolid during prompt
15 Jan 1999: option QUIET added
Written by: Kris Buelens IBM Belgium;
KRIS at VMKBBR01 27 Feb 1995*/

address command

parse upper value translate(arg(1),,'(') with options
noisy=1 ;force=0; ReAcc=1
 /* init all our options */
do while options<>'' /* analyse requested options */
parse var options option options
select
when option='QUIET' then noisy=0
when abbrev('FORCE',option,2) then Force=1
when abbrev('REACCESS',option,2) then Reacc=1
when abbrev('NOREACCESS',option,2) then Reacc=0
Otherwise call errexit 5, 'Invalid option:' option
end
end
if noisy then
Say 'This exec will force a complete disconnect from ALL SFS servers'

if Reacc then t='reACCESSed' ; else t='RELEASEd'
if noisy then ToCons='!CONS'
 else ToCOns=''
'PIPE (END ?) COMMAND Q ACCESSED!LOCATE 23.3 /DIR/',
 '!FO: FANOUT',
 '!C: COUNT LINES!BUFFER!CHANGE //
 /!F: FANIN 1 0',
 tocons,
 '?C:!NFIND 0'!!,
 '!SPEC /Following directories will be' t'./ 1 Write',
 Files
 Type  Directory/ 1',
 !F:',
 '?FO:!SORT 3',
 /* Place fmode extensions last */
 '!CHANGE W2 "R/O"FORCERO"', /* Prepare for ACCESS RW/RO option */
 '!CHANGE W2 "R/W"FORCERW"',
 '!SPEC /CMDCALL ACCESS/ 1 W5 NW W1 NW /(/ NW W2 NW',
 '!JOIN * X15!APPEND LITERAL!VAR toREACC'
```

```
If linesize()<>0 & noisy then
if toReacc<>'' then do
Say 'Please Select an answer'
If Reacc then
Say ' -Yes-
(disconnect and we reACCESS the above dirs)'
Say ' -Yes-
(disconnect completely from SFS)'
If Reacc then do
Say ' CHange pool1 pool2
(same, but use "pool2:" iso "pool1:" )'
Say ' NOReaccess
(disconnect and don''t reACCESS dirs )'
Say ' QUIT
(and we do nothing)'
Parse upper external ans 0 w1 .; ans=strip(ans)
Select
when abbrev('NO',ans,1) ! ans='QUIT' then
do;say 'Nothing done';exit;end
when abbrev('YES',ans) then nop
when abbrev('NOREACCESS',ans,3) then toReacc=''
when abbrev('CHANGE',w1,1) then do
parse var ans . pool1 pool2 .
if right(pool1,1)<>':' then pool1 =pool1':'
if right(pool2,1)<>':' then pool2 =pool2':'
if pool1='' ,
! pool2='' then call ErrExit 6,'CHANGE needs two fpoolids'
'PIPE VAR toREACC!CHANGE /'pool1'/'pool2'/!VAR REACC2'
if toreacc=reacc2 then
call ErrExit 7,'"'pool1'" not found in dirs...'
toreacc=reacc2
Otherwise
Call ErrExit 6,'What ? "'ans'"' ,'Nothing done'
end
```

```
if Force then do ; t='FORCE'; call CSL 'DMSPURWU RC REAS T 5' ; end
call CSL 'DMSPURWU RC REAS'
If rc<>0 then Say 'DMSPURWU Retcode:' rc 'Reasoncode' reas
if Reacc then
if toReacc<>'' then
'PIPE VAR toREACC' ToCons '!SPLIT X15!COMMAND' toCons
exit rc
ERREXIT: /* general errorexit routine */
parse upper source . . myname mytype . syn .
do i=2 to arg()
 /* give errormessages (if any) */
 say myname':' arg(i)
exit arg(1)
```

Kris Buelens and Guy De Ceulaer
Advisory Systems Engineers
IBM (Belgium)

A full screen console interface – part 19

Editor's note: the following article is the conclusion of an extensive piece of work which has been published over several issues of VM Update. It was felt that readers could benefit from the entire article and from the individual sections. Any comments or recommendations would be welcomed and should be addressed either to SDS or directly to the author at feranado_durante@vnet.ibm.com.

MATCH HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
```

```
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
```

Use the MATCH command to search backward the CSC log file for a.22

character mask.

EXAMPLE: MATCH ABC*

```
Search log file for record beginning with ABC.
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
```

Use the MATCH command to search backward the CSC log file for a character mask.

```
.cs 1 off
.cs 2 on
[]Format[%
>>—.-Match-.—mask———————————————————>><
‘-\——'
[]Class[%
4
.cs 2 off
.cs 3 on
[]Operands[%
```

mask

is the data mask to use. It can be from 1 to 36 characters long.

Two characters have a special meaning in a mask, * (asterisk) and

% (percent).

```
*
represents any number of characters.
%
represents one single character.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
```

```
[]Usage Notes[%

1. The MATCH command expects one single space between the command

name, or a valid abbreviation, and the mask. MATCH *ABC* and

MATCH *ABC* are not equivalent. The latter will only locate a

record with a blank as the first character..23

2. If the character \ is used instead, the mask must follow it with

no spaces in between. MATCH *ABC* produces the same results as

\*ABC*.

.cs 5 off

.cs 6 on

[]Messages[%
Ø31ØE Missing operand(s)
Ø35ØE String too long. Must be from 1 to 36 characters
Ø351I String not found
.cs 6 off
```

OP HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%

Use the OP command to enter a command for a controlled service machine.
EXAMPLE: OP USER ABC DISPLAY ALL
Send command DISPLAY ALL to the console of ABC machine.
.cs Ø off
.cs 1 on
```

```
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the OP command to enter a command for a controlled service machine.
.cs 1 off
.cs 2 on
[]Format[%
>>—OP—.———————.—command————————————————><
]-prefix———]
‘-User userid-'
[]Class[%
6.24
.cs 2 off
.cs 3 on
[]Operands[%
prefix
one letter prefix identifying destination service machine.
userid
name of the destination service machine. This format is required
when you define more than one service machine with the same prefix.
command
command to be sent.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
```

1. OP uses the CP SEND command. This command requires the service

machine to be disconnected to work.

2. You cannot execute a command that start a full screen operation.

Unexpected, but predictable, results may occur.

```
.cs 5 off
.cs 6 on
[]Messages[%
Ø31ØE Missing operand(s)
Ø37ØE Missing value for userid
Ø371E Value value... is too long. Must be from 1 to 8 characters
Ø372E Userid userid is not defined
Ø373E prefix is not a valid prefix. Must be one character long
```

```
Ø374E Prefix prefix is not defined
Ø375E You are not authorized to operate the userid machine
.cs 6 off
```

PRINT HELPCSC

.cm VM Software Services

```
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
Use the PRINT command to create a PRT file from the CSC log file..25
EXAMPLE: PRINT 1ØØ
Print 1ØØ lines from the CSC log file.
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the PRINT command to create a PRT file from the CSC log file.
.cs 1 off
.cs 2 on
[]Format[%
.-2ØØ-.
>>—Print—+——+————————————————————><
]-n—]
‘-*—'
[]Class[%
3
.cs 2 off
.cs 3 on
[]Operands[%
n
number of lines to print. Enter “*” to print all lines to the end
of the log file.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
```

1. Printing starts a the first line displayed on the screen. Use

the SWITCH/INCLUDE/EXCLUDE to select lines to print.

2. Records are truncated if longer than 121 bytes.

3. Use the WRITE command to create a disk file.

```
.cs 5 off
.cs 6 on

[]Messages[%
Ø292E Error creating PrintLog file.26
Ø311E Invalid PRINT operand: operand
Ø312E Unexpected PRINT operand: operand
Ø340W Data file is empty
Ø341E Command interrupted. Reason code is code
.cs 6 off
```

REDSPLY HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
Use the & command prefix to redisplay the last entered command.
EXAMPLE: &/ABC
Execute command /ABC and redisplay it.
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the & command prefix to redisplay the last entered command.
.cs 1 off
.cs 2 on
[]Format[%
>>—&—.———————.—————————————————————><
‘-CSC command-'
[]Class[%
Ø
```

```
.cs 2 off
.cs 3 on
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
```

1. The & command prefix is processed locally by the user program.27 (CSCUSR). It is removed, before the command is sent to the CSC service machine.

```
.cs 5 off
.cs 6 on
.cs 6 off
```

RELEASE HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
```

Use the RELEASE command to remove the HOLD attribute from a message.

EXAMPLE: RELEASE 2

Release the second message displayed with the HOLD attribute.

```
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the RELEASE command to remove the HOLD attribute from a message.
.cs 1 off
.cs 2 on
[]Format[%
>>—Release—.-1-.—.-1-.————————————————————><
‘-n-’ ‘-m-’
```

```
[]Class[%
5
.cs 2 off
.cs 3 on
[]Operands[%
n
first message on hold to be affected.
m.28
last message on hold to be affected.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
```

1. Messages on hold are identified by a > sign immediately after the user prefix.

2. If a message is released while you are in Browse mode, the screen will not be updated until a command is entered that causes it to be refreshed.

```
.cs 5 off
.cs 6 on
[]Messages[%
Ø311E Invalid RELEASE operand: operand
Ø312E Unexpected RELEASE operand: operand
Ø360W You are not authorized to release user_id messages
Ø361W Message number number not found
Ø362E Invalid message number: number. Must be between n1 and n2
.cs 6 off
```

REPEAT HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
```

Use the RELEASE command to remove the HOLD attribute from a message.
EXAMPLE: RELEASE 2
Release the second message displayed with the HOLD attribute.

```
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the RELEASE command to remove the HOLD attribute from a message..29
.cs 1 off
.cs 2 on
[]Format[%
>>—Release—.-1-.—.-1-.————————————————————><
‘-n-' ‘-m-'
[]Class[%
5
.cs 2 off
.cs 3 on
[]Operands[%
n
first message on hold to be affected.
m
last message on hold to be affected.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
```

1. Messages on hold are identified by a > sign immediately after the user prefix.

2. If a message is released while you are in Browse mode, the screen will not be updated until a command is entered that causes it to be refreshed.

```
.cs 5 off
.cs 6 on
[]Messages[%
Ø311E Invalid RELEASE operand: operand
Ø312E Unexpected RELEASE operand: operand
Ø360W You are not authorized to release user_id messages
Ø361W Message number number not found
```

```
Ø362E Invalid message number: number. Must be between n1 and n2
.cs 6 off
```

RETRIEVE HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on.30
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
```

Use the ? command to re-display the previous command.
EXAMPLE: ?
Recall previous command.

```
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the ? command to re-display the previous command.
.cs 1 off
.cs 2 on
[]Format[%
>>—?————————————————————————————><
[]Class[%
Ø
.cs 2 off
.cs 3 on
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
```

1. PF12 and PF24 are assigned to the ? command.
2. This is a local command, executed by the user program (CSCUSR).

The service machine is not informed.

3. Operands entered with this command are ignored.

4. CSCUSR uses a buffer to store commands as they are entered.

Use PF11 or PF23 to scan forward this buffer.

There is no command to perform this function.

```
.cs 5 off
.cs 6 on
[]Messages[%.31
Ø28ØW Only valid after first command is entered
Ø282W No more data found in Retrieve buffer
.cs 6 off
```

SHIFT HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
```

Use the SHIFT command to change the columns displayed from a message.
EXAMPLE: SHIFT RIGHT 1Ø
Shift the screen 1Ø columns to the right.

```
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the SHIFT command to change the columns displayed from a message.
.cs 1 off
.cs 2 on
[]Format[%
>>—.——.—.-LEft—.—n————————————————————><
‘.SHift-' ‘-RIght-'
[]Class[%
1
.cs 2 off
.cs 3 on
[]Operands[%
Left
shift the screen to the left.
Right
shift the screen to the right.
```

```
n.32
number of columns to shift the screen.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
```

1. PF1Ø and PF22 perform a special form of this command.

If the first column is displayed is 1, a SHIFT RIGHT 64 is

executed; if not, column 1 will become the first column

displayed.

2. A shift value of zero, will force column 1 to be displayed.

```
.cs 5 off
.cs 6 on
[]Messages[%
Ø31ØE Missing operand(s)
Ø311E Invalid SHIFT operand: operand
Ø312E Unexpected SHIFT operand: operand
Ø32ØE SHIFT value value is too big
.cs 6 off
```

SWITCH HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
```

Use the SWITCH command to enable or disable display options.
EXAMPLE: SWITCH DATE USER

Reverse the setting of the DATE and USER fields.

```
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the SWITCH command to enable or disable display options.
.cs 1 off
.cs 2 on.33
[]Format[%
.<——<.
>>—.-Switch-.—+-Cms——+————————————————————><
‘-Swap—' ‘-Date—'
‘-Filter-'
‘-Time—'
‘-User—'
‘-Wrap—'
[]Class[%
1
.cs 2 off
.cs 3 on
[]Operands[%
```

Cms
CMS changes the way data is scrolled in current mode. By default (OFF) when a new message is received by the service machine all scrollable data is shifted up one line and the new one is inserted at the bottom of the screen.
Date
controls the display of the DATE field. The default is not to display the DATE field (OFF).
Filter
controls how to process messages with the NODISPLAY attribute in browse mode. The default (OFF) is to filter NODISPLAY messages only in current mode.
Time
controls the display of the TIME field. The default is to display the TIME field (ON).
User
controls the display of the USER field. The default is not to display the USER field (OFF).

Wrap

causes message text to be displayed on one or more screen lines.

```
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
```

1. The CLEAR command (not the CLEAR key) requires CMS to be ON to.34

function.

2. Turning CMS ON could help refreshing the screen when connected

to low speed lines, as it requires less data to append one line

than to reformat the whole scrollable area.

3. While WRAP is active the SHIFT LEFT/RIGHT command has no visible

effect.

4. Specifying the same operand twice (ex: SWITCH DATE DATE) has no effect on the operand but causes the screen to be re-freshed.

```
.cs 5 off
.cs 6 on
[]Messages[%
Ø310E Missing operand(s)
Ø311E Invalid SWITCH operand: operand
.cs 6 off
```

TOP HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
```

Use the TOP command to show the first screen of data.
EXAMPLE: TOP
Show first screen of data.

```
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the TOP command to show the first screen of data.
.cs 1 off
.cs 2 on
[]Format[%
>>—Top————————————————————————————><.35
[]Class[%
3
.cs 2 off
.cs 3 on
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
1. PFØ4 and PF16 are assigned to the TOP command.
.cs 5 off
.cs 6 on
[]Messages[%
Ø312E Unexpected TOP operand: operand
.cs 6 off
```

UP HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
```

Use the UP command to scroll backward the CSC log file.
EXAMPLE: UP 3
Scroll backward 3 lines.

```
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
```

```
[]Purpose[%
Use the UP command to scroll backward the CSC log file.
.cs 1 off
.cs 2 on
[]Format[%
.-1-..36
>>─Up─┬─┬──────────────────────────────><
'-n-'
[]Class[%
3
.cs 2 off
.cs 3 on
[]Operands[%
n
number of lines to scroll. The default is one.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
```

1. UP n will display up to n lines before the first line on screen.

This may not be the expected result if you are on the current

screen and have messages with the HOLD attribute. Using the BOTTOM command before solves this problem.

```
.cs 5 off
.cs 6 on
[]Messages[%
Ø311E Invalid UP operand: operand
Ø312E Unexpected UP operand: operand
.cs 6 off
```

WRITE HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
```

Use the WRITE command to create a disk file from the CSC log file.

EXAMPLE: WRITE 1ØØ
Write 1ØØ lines from the CSC log file.

```
.cs Ø off
.cs 1 on.37
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the WRITE command to create a disk file from the CSC log file.
.cs 1 off
.cs 2 on
[]Format[%
.-2ØØ-.
>>—Write—+—+—————————————————————><
]-n—]
‘-*—'
[]Class[%
3
.cs 2 off
.cs 3 on
[]Operands[%
n
number of lines to write. Enter “*” to write all lines to the end
of
the log file.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
```

1. Writing starts a the first line displayed on the screen. Use the

SWITCH/INCLUDE/EXCLUDE to select lines to write.

2. Use the PRINT command to create a PRT file.

```
.cs 5 off
.cs 6 on
[]Messages[%
Ø292E Error creating PrintLog file
Ø311E Invalid WRITE operand: operand
Ø312E Unexpected WRITE operand: operand
Ø340W Data file is empty
Ø341E Command interrupted. Reason code is code
```

```
.cs 6 off.38
```

CSCCFG HELPMENU

```
.cm VM Software Services
.cm
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
```
A file may be selected for viewing by placing the cursor under any
character of the file wanted and pressing the ENTER key or the PF1 key.
A MENU file is indicated when a name is preceded by an asterisk (*).
A TASK file is indicated when a name is preceded by a colon (:).
For a description of the HELP operands and options, type HELP HELP.
DFRecs
DFSize
Local
Message
MSG
Options
PFX
Prefix
REmote
Route
RTE
Title
TTL
User
USR

CSCCFG HELPABBR

DFRECS DFRecs 3
DFRECS DFSize 3
LOCAL Local 1
MESSAGE Message 1
MESSAGE MSG 3
OPTIONS Options 1
PREFIX Prefix 1
PREFIX PFX 3
REMOTE REmote 2
ROUTE Route 1

ROUTE RTE 3
TITLE Title 1
TITLE TTL 3
USER User 1
USER USR 3

DFRECS HELPCSCC.39

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
```
Use the DFRECS statement to define the size of the CSC log file.
EXAMPLE: DFRECS 2Ø48
Create a log file with 2Ø48 records.
```
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
```
Use the DFRECS statement to define the size of the CSC log file.
```
.cs 1 off
.cs 2 on
[]Format[%
.-1Ø24-.
>>—.-DFRecs-.—+———+————————————————————><
‘-DFSize-' ‘-n——'
.cs 2 off
.cs 3 on
[]Operands[%
n
number of records for the CSC log file.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
```

1. The minimum size for the log file is 128 records. A value less than the minimum will force the use of the default 1Ø24 records.

2. The size should be a multiple of 32, It will be rounded up if necessary.

3. The disk containing the log file should be formatted with 4KB blocks, to improve performance of browse commands..40

4. DFSIZE is a synonym for DFRECS.

```
.cs 5 off
.cs 6 on
[]Messages[%
Ø050E Missing DFRECS operand(s). Statement discarded
Ø051E Invalid DFRECS operand: operand. Statement discarded
Ø052E Unexpected DFRECS operand: operand. Statement discarded
Ø060W Value value for DFRECS is too small. Default of 1Ø24 used
Ø061W DFRECS value adjusted from n1 to n2
.cs 6 off
```

LOCAL HELPCSCC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
Use the LOCAL statement to identify the CSC local node.
EXAMPLE: LOCAL VM1 RES1
Define APPC/VM resource RES1 for local node VM1.
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the LOCAL statement to identify the CSC local node.
.cs 1 off
.cs 2 on
[]Format[%
.-Global-.
>>—Local—nodeid—resourceid—+———+————————————><
'-Local—'
.cs 2 off
.cs 3 on
[]Operands[%.41
```

```
nodeid
name to identify to local node. Must be from 1 to 8 characters.
resourceid
name for the resource associated with this node. Must be a valid
APPC/VM resource name.
Local
Define resourceid as an APPC/VM Local resource.
Global
Define resourceid as an APPC/VM Global resource.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
```

1. Global resources require TSAF to be working.

```
.cs 5 off
.cs 6 on
[]Messages[%
Ø050E Missing LOCAL operand(s). Statement discarded
Ø052E Unexpected LOCAL operand: operand. Statement discarded
Ø053E LOCAL operand “operand...” is too long. Statement discarded
Ø070E Only one LOCAL statement allowed. Statement discarded
Ø071E Node name name is not unique. Statement discarded
Ø072E Resource name name is not unique. Statement discarded
.cs 6 off
```

MESSAGE HELPCSCC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
Use the MESSAGE statement to define message rules.
EXAMPLE: MESSAGE USER U1 YELLOW HIGH HOLD NOCASE LOCATE *Abend*
Display in yellow, or highlighted, with the HOLD attribute
any message from user U1 with the text ABEND on it. Ignore
text case for this rule..42
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
```

```
[]CSC Tool[%
[]Purpose[%
Use the MESSAGE statement to define message rules.
.cs 1 off
.cs 2 on
[]Format[%
(1)
>>– .-Message-.– User– .-userid-.– Locate– mask– .– –
– – – – – – .– – – – – >
'-MSG– – ' '-*– – – ' '-Exit– exitname-'
>– .– – – – – – – .– .– – – – – – – – – .– .– – – – –
– – – – – – .– – – – >
'-Name– msgname-' '-RElease– relname-' '-ROute– .-
routeid– .-'
'-routename-'
>– .– – – – – .– .– – – – – .– .– – – – – .– – – – – –
– – – – – – – – – ><
]-Alarm– – ] ]-HIgh– – – ] ]-Blue– – – ]
]-Hold– – – ] ]-BLInk– – ] ]-RED– – – ]
]-NOCase– – ] ]-REvvideo– ] ]-Pink– – – ]
]-NODisplay-] '-UNderline-' ]-Green– – ]
'-UNique– – ' ]-Turquoise-]
]-Yellow– – ]
'-White– – '
Notes: (1) LOCATE must be the last operand entered.
Other operands may be entered in any order.
.cs 2 off
.cs 3 on
[]Operands[%
userid

originating userid for this message. Use asterisk (*)
to ignore
this field.
mask
is the data mask to use. It can be from 1 to 36 charac-
ters long.
Two characters have a special meaning in a mask, *
(asterisk) and
% (percent).
*
represents any number of characters.
%
represents one single character..43
exitname
CMS exec to be invoked for this message. The following
data is
```

passed to this exit as the first argument (arg(1)).
From To Length Contents
Ø1 Ø8 Ø8 Date, format yy/mm/dd
Ø9 16 Ø8 Time, format hh:mm:ss
17 24 Ø8 Userid originating the message
25 25 Ø1 Message flags
.... ..1. Hold message
....1 Do not display message
26 27 Ø2 (CCHPNUM and CCHCNUM bytes)
28 28 Ø1 Message length
29 — — Message text (maximum is 212 bytes)
msgname
name to be associated with this message. This value is used by the
UNIQUE and RELEASE operands.
relname
messages to be released by this message. All messages with the HOLD
attribute, whose msgname matches this relname are released.
routeid
userid to receive a copy of this message. Use routename to specify
more than one userid, or if userid resides on a different VM node.
routename
name of a list of users to receive a copy of this message. This
list must be defined by a ROUTE statement.
Alarm
sound the alarm when displaying this message. Only active sessions
in Current mode when the message is received are affected.
Hold
do not scroll this message. The HOLD attribute may be removed by
another message, or by the RELEASE command.
NOCase
ignore text case when processing this rule.
NODisplay
do not display this message. These messages are written to the log
file
but are not displayed in Current mode. They are displayed in Browse
mode unless the SWITCH FILTER command is executed.

UNique
do not scroll this message. The HOLD attribute may be removed by
the next message processed by this rule, another message with the
RELEASE operand, or by the RELEASE command..44
HIgh
display this message highlighted on monochrome terminals.
BLInk
REvvideo
UNderline
extended attributes to use with this message on terminals that support them.
Blue
RED
Pink
Green
Turquoise
Yellow
White
colour to use with this message on terminals that support them.

```
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
```

1. Message rules are processed in reverse order. Generic rules must be defined first.

2. Display attributes are reset when a message is released.

3. MSG is a synonym for MESSAGE.

```
.cs 5 off
.cs 6 on
[]Messages[%

Ø051E Invalid MESSAGE operand: operand. Statement discarded
Ø080E Missing USER option for MSG. Statement discarded
Ø081E Missing LOCATE option for MSG. Statement discarded
Ø082E Missing value value for MSG. Statement discarded
```

```
Ø083E operand value value... for MSG too long. Statement discarded
Ø084E operand mask mask... for MSG too long. Statement discarded
.cs 6 off
```

OPTIONS HELPCSCC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997.45
[]CSC Tool[%
[%
[%
```
Use the OPTIONS statement to define global CSC processing options.
EXAMPLE: OPTIONS MSG
Use CP MSG command to forward messages to users.
```
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the OPTIONS statement to define global CSC processing options.
.cs 1 off
.cs 2 on
[]Format[%
.-MSG──.
>>─Options─┼────┼─.──.──────────────────><
‘-MSGNOH-' ‘-PRINT-'
.cs 2 off
.cs 3 on
[]Operands[%
MSG
MSGNOH
CP command to use to forward a message to a user.
PRINT
Copy all Console messages to the Printer.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
.cs 5 off
```

```
.cs 6 on
[]Messages[%

Ø050E Missing OPTIONS operand(s). Statement discarded
Ø054E Invalid OPTIONS operand: operand. Ignored
.cs 6 off
```

PREFIX HELPCSCC.46

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%

Use the PREFIX statement to associate a controlled
Service machine with
a single character prefix.
EXAMPLE: PREFIX X VMSERV1 BLUE
Define prefix X for VMSERV1. Display messages in blue.
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the PREFIX statement to associate a controlled Service machine with
a single character prefix.
.cs 1 off
.cs 2 on
[]Format[%


(1)
>>— .-Prefix-.— prefix— userid— .— — — — — — — .— .— —
— — — .— — — — — — — >
‘-PFX— — ' ‘-Class— class-' ]-HIgh— — — ]
]-BLInk— — ]
]-REvvideo— ]
‘-UNderline-'
>— .— — — — — .— — — — — — — — — — — — — — — — — — —
— — — — — — — — — ><
```

]-Blue– – –]
]-RED– – –]
]-Pink– – –]
]-Green– –]
]-Turquoise-]
]-Yellow– –]
'-White– – '
Notes: (1) Optional operands can be entered in any order.
.cs 2 off
.cs 3 on

[]Operands[%
prefix.47
single character prefix to define.
userid
name of the Service Machine to be associated with prefix.
class
class for the Service Machine. Must be in the range 25-32.
BLInk
REvvideo
UNderline
default extended attributes to use with all messages from this
Service Machine.
Blue
RED
Pink
Green
Turquoise
Yellow
White
default colour to use with all messages from this Service Machine.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%

1. If a class is specified, only users with the same class are authorized to execute restricted commands.
2. PFX is a synonym for PREFIX.

```
.cs 5 off
.cs 6 on
[]Messages[%
Ø050E Missing PREFIX operand(s). Statement discarded
Ø051E Invalid PREFIX operand: operand. Statement discarded
Ø053E PREFIX operand “operand...” is too long. Statement discarded
Ø090E prefix is an invalid Prefix. Must be one character long
Ø091E Missing userid for Prefix “prefix”. Statement discarded
Ø092E Missing class value in PREFIX statement. Discarded
Ø093E PREFIX class class not in the range 25-32. Statement discarded
.cs 6 off
```

Fernando Duarte
Analyst (Canada)

VM news

What else can one talk about except for the Computer Associates purchase of Sterling Software. This is the single largest software company merger of all time. Financial analysts will continue to look at the staggering amount of the deal. $ 4 billion of stock with no cash was used for this transaction and this makes it a no brainer for CA. Talk among users of Sterling Software appears to be far more interesting. With nearly 20,000 clients worldwide and 90% of the Fortune 100 companies using some form of Sterling Software there are bound to be some operational concerns. CA has had a reputation in the past as an extremely tough negotiator when working with clients of recently acquired companies. It appears from the items on the various VM news lists that VM users are very concerned about the future of their Sterling Software and support today. The deal must still clear antitrust issues and will also probably take advantage of the SEC "fast track" rules. This means that all of the concerns of users can be addressed sooner rather than later.

***

IBM announced it has joined with Cisco to provide the Cisco client base with IBM's host integration software. This joint effort brings the two giants closer together after their technology marketing alliance announced in 1999. The IBM programs that are now part of the Cisco EAP structure are IBM Host On-Demand ®, IBM Personal Communications® and IBM Host Publisher®. This form of partnering only makes sense for the mainframe community as a whole. Industry estimates show that up to 80% of all business data resides on mainframes and larger servers. The effect of this relationship is that mainframe systems can now have a simple and effective tool to extend the data to intranet, extranet and Internet users. This should extend the life of mainframe systems and the support infrastructure.

***

SHARE will be holding its first major conference of the new millenium in Anaheim, CA March 5-10, 2000. It appears from all indications it will be a highly attended conference. IS professionals are coming off a successful Y2K transition and are looking for new or updated technologies. This conference with the hundreds of offered sessions will make it possible for IS people to obtain information on new products and services that help them in their daily work efforts and in planning for the future of their departments. It is expected that more than 2000 people will attend the conference. Some of the current topics to be discussed include focuses on EAI, Linux and its fit within the Enterprise and Language Usability. The great thing about SHARE is not only does IBM and other companies make presentations but so do everyday users. This allows attendees to gain by sharing.

SOFTWARE DIVERSIFIED SERVICES