
1. MFNetDisk – Guía Completa para VTS

Este documento pretende explicar cómo crear en nuestra instalación, un sistema de robótica virtual utilizando una herramienta llamada MFNetDisk, disponible para descarga en <http://www.mfnetdisk.com>

Esta herramienta emula de una manera muy eficaz, un sistema de volúmenes de disco 3390 utilizando la pila TCP/IP, así como un sistema de almacenamiento en cinta 3490, tanto MTL (librería de cintas robotizada, como si fuera una 3494), como unidades standalone – drives- (No-MTL).

Este documento se centrará en la instalación de la parte de cintas, con el fin de crear un Virtual Tape Server o VTS (como los B18, o B20 en la familia IBM 3494 VTS).

1.1 Introducción a MFNetDisk

MFNetDisk es una aplicación escrita y disponible en la Web por Shai Hess, y se puede obtener de <http://www.mfnetdisk.com>

Esta aplicación, utilizando un túnel basado en TCP/IP, emula con total precisión y rendimiento (dependiendo del ancho de banda de comunicaciones TCP/IP) dispositivos de discos y/o cintas, de forma que pueden coexistir en nuestra instalación de discos reales, este equipamiento para guardar datos, mucho mas económico ya que se monta en un servidor normalito Wintel que utiliza su almacenamiento propio. Vendría a parecerse a un servicio NFS.

Pero además de esto, este sistema montado en otro servidor remoto, nos permite replicar datos asincrónicamente y de ese modo, disponer de salvaguarda de todos los datos que MFNetDisk ofrece al entorno mainframe. Para mas información, consultar la web <http://www.mfnetdisk.com> con el fin de leer todas las posibilidades que tiene este producto.

Pero en este caso concreto, solo vamos a utilizar una PEQUEÑÍSIMA función que ofrece nuestra aplicación, que será la de actuar como un VTS o un Virtual Tape Server.

NOTA: Este documento NO pretende sustituir el actual sistema VTS que puedas tener instalado en tu entorno, ya sea un IBM 3494 VTS B18, B20, etc, sino que es una aplicación que ofrece un funcionamiento ANALOGO (salvo por las cintas “stacked” propias de un VTS).

Una vez registrado en la pagina, procederemos a descargar el producto, que es un ZIP. En su interior, vemos que existen varias carpetas, ya que por su filosofía, es un entorno cliente-servidor.

Por una parte, se instala en un equipo con Windows que será quien sirva el espacio de almacenamiento de cintas o discos que MFNetDisk va a ofrecer como espacio libre para uso del mainframe, por lo que dependiendo de los datos a manejar, nuestro disco duro del servidor Windows necesitará más o menos capacidad.

Por otra parte, existe una parte que se debe instalar en el mainframe, que servirá de nexo de unión entre la parte del servidor Windows y el z/OS, y creará el peer TCP/IP que “hablará” con el servidor Windows.

Por tanto, pasaré a explicar paso a paso como se instala el MFNetDisk, y posteriormente, como realizar una configuración desde cero, incluyendo la habilitación del OAM, DFSMS y RMM para toda la gestión de las cintas de robotica “virtual”.

Este documento, SOLO tratará de habilitar la parte del almacenamiento en cinta, el almacenamiento en disco y todas sus posibilidades lo dejaré para un artículo posterior. Cabe decir que MFNetDisk por el momento, SOLO emula cintas 3490E con tecnología de 36 pistas, esto es importante tenerlo en mente para cuando toque definir el sistema en z/OS.

1.2 Instalación MFNetDisk

Para instalar MFNetDisk debemos instalarlo en dos sitios: Uno, el propio z/OS que utilizará los servicios, mediante una started task que estará siempre en memoria. Y dos, el servidor Windows donde se alojaran los datos de cintas y discos y donde configuraremos todo el proceso.

Por tanto, lo primero una vez descargado el producto desde <http://www.mfnetdisk.com> será descomprimirlo en nuestro equipo Windows que hará de servidor de almacenamiento, creándose varias carpetas.

1.2.1 Instalación en z/OS

Lo primero de todo es subir al z/OS dos ficheros con extensión XMI que están dentro de la carpeta MVSFiles. Los ficheros en cuestión son **MPC.VXXX.LOAD.XMI** y **MPC.VXXX.SAMPLIB.XMI**, siendo XXX el número de versión. En la escritura de este manual, la versión era la 013.

Un fichero XMI es un fichero de formato XMIT, de transmisión estándar que agrupa en su interior uno o mas ficheros (como el TAR en Unix), por tanto, su contenido tendrá todas las utilidades y miembros necesarios para el correcto funcionamiento de MFNetDisk.

Para subir los ficheros al z/OS, podemos utilizar varios métodos, como FTP, pero por seguridad en la reserva de datasets FB, usaremos el File Transfer que viene con la emulación de terminal 3270 del IBM Personal Communications. Para ello, seguiremos los siguientes pasos:

1.- Situarnos en el ISPF en la opción **6 – Commands**, y a continuación, pulsar el icono Enviar archivos al sistema principal tal y como se muestra la siguiente figura:

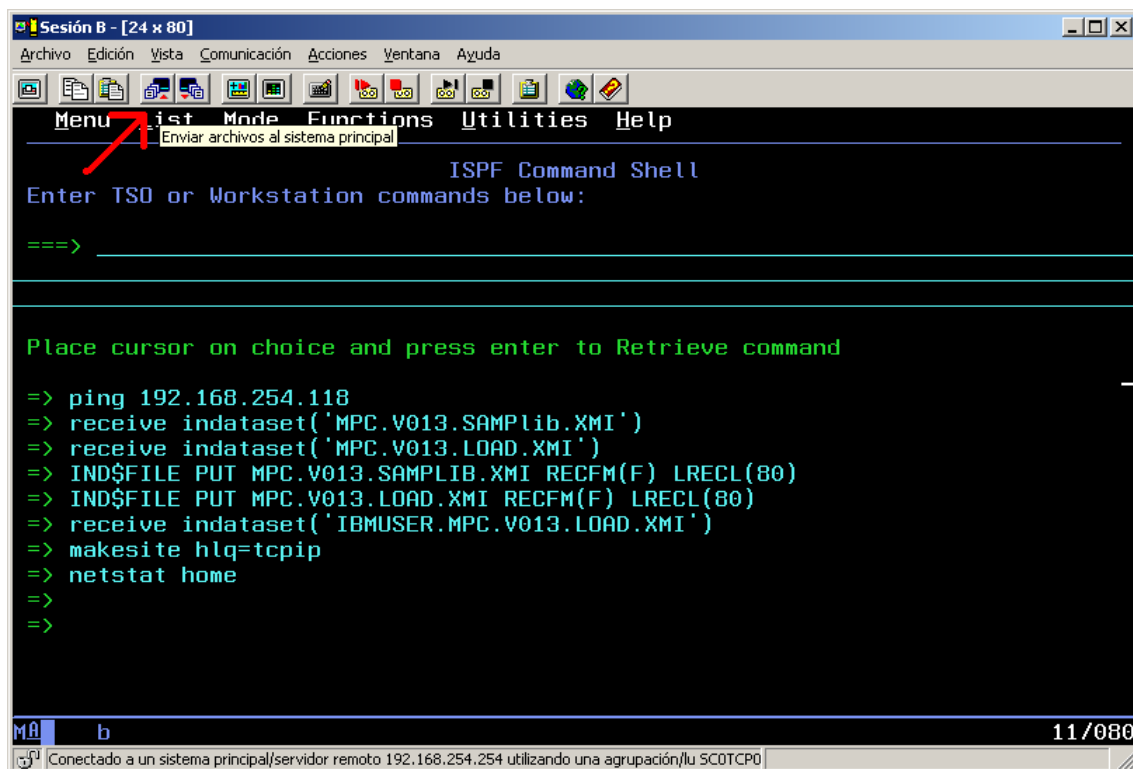


Fig. 1: Enviar Archivos al sistema principal

2.- Nos aparecerá una ventana para la transmisión de ficheros, similar a la de la figura siguiente:

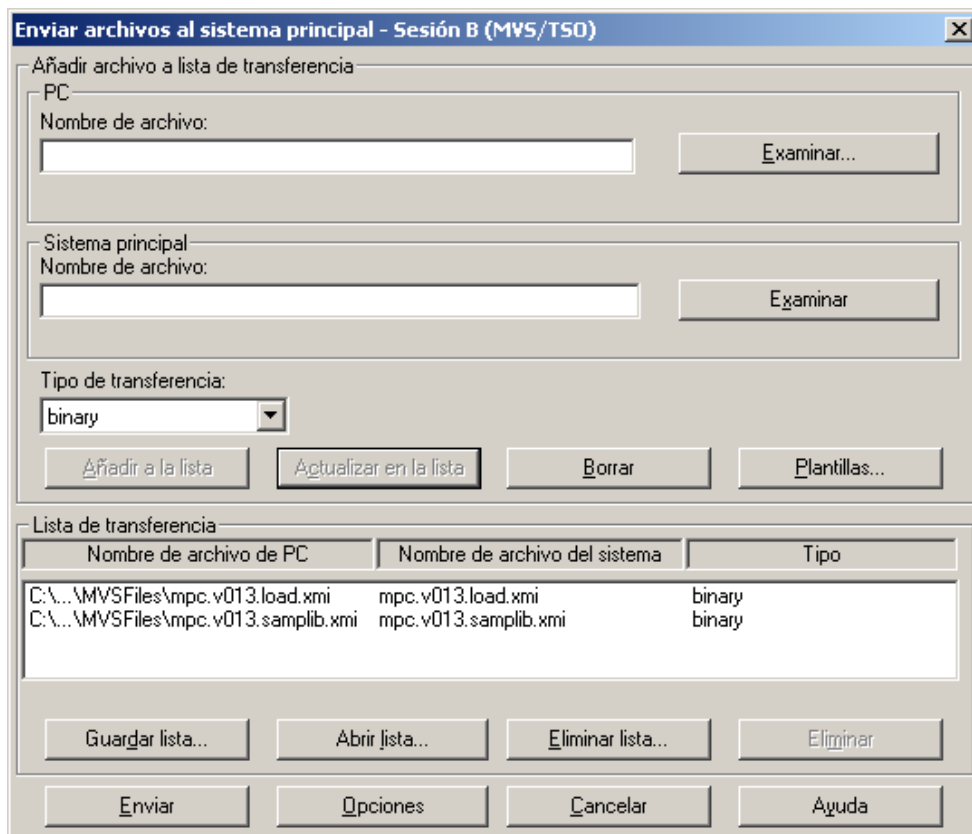


Fig. 2: Transferencia de ficheros

3.- Hay que poner **ESPECIAL ATENCION** al botón Opciones, ya que la transferencia debe ser **BINARIA** y con registros fijos de 80 caracteres (**FB 80**), ya que de lo contrario, el fichero XML se podría corromper. Para cambiar la transferencia –por defecto el bloqueo lo pone variable, en Opciones, saldrá la siguiente ventana para poder especificar los parámetros de subida:

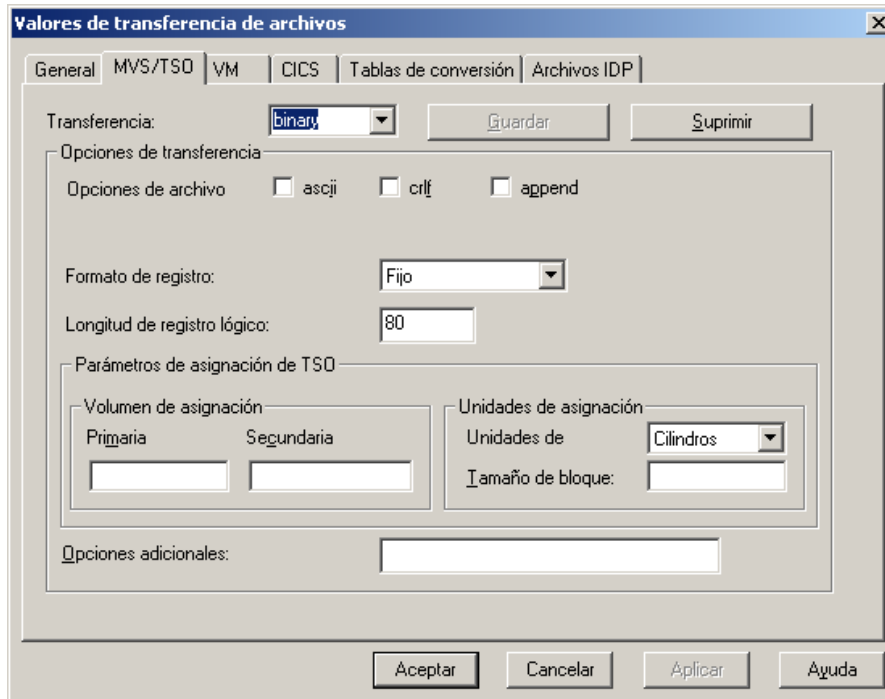


Fig. 3: Opciones de transferencia de ficheros.

Elegimos el tipo de transferencia (BINARY), cambiamos el formato y damos aceptar, con lo que BINARY ya subirá el dataset con el formato correcto.

4.- Al dar aceptar, la transferencia comenzará saliendo la barra de progreso oportuna, hasta que termine:

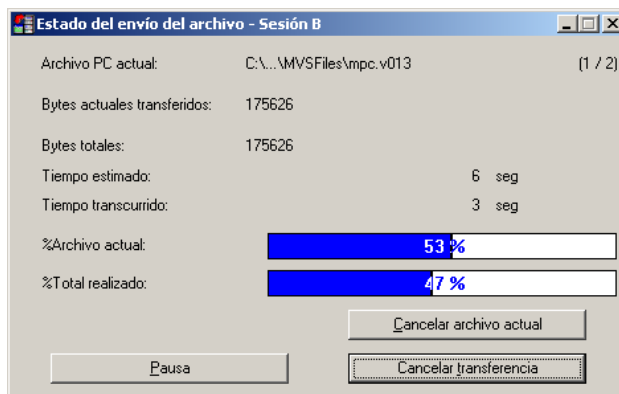


Fig. 4: Barra de progreso

5.- Una vez terminada, la transferencia, nos aseguraremos que los datasets existen desde la opción 3.4 del ISPF. En el manual, los he denominado, MPC.V013.SAMPLIB.XMI y MPC.V013.LOAD.XMI.

```

Sesión B - [24 x 80]
Archivo Edición Vista Comunicación Acciones Ventana Ayuda
Menu Options View Utilities Compilers Help

DSLIST - Data Sets Matching MPC
Command ===>
Row 1 of 2
Scroll ===> PAGE

Command - Enter "/" to select action      Message      Volume
-----
MPC.V013.LOAD.XMI                        ZASYS1
MPC.V013.SAMPLIB.XMI                    ZASYS1
***** End of Data Set list *****

MA  b 04/015
Conectado a un sistema principal/servidor remoto 192.168.254.254 utilizando una agrupación/lu SCOTCP0

```

Fig. 5: Datasets subidos correctamente

```

Sesión B - [24 x 80]
Archivo Edición Vista Comunicación Acciones Ventana Ayuda
Data Set Information
Command ===>

Data Set Name . . . . : MPC.V013.LOAD.XMI

General Data
Management class . . : **None**
Storage class . . . . : **None**
Volume serial . . . . : ZASYS1
Device type . . . . . : 3390
Data class . . . . . : **None**
Organization . . . . . : PS
Record format . . . . : FB
Record length . . . . : 80
Block size . . . . . : 6160
1st extent blocks . . : 8
Secondary blocks . . . : 24
Data set name type . . :

Current Allocation
Allocated blocks . . : 56
Allocated extents . . : 3

Current Utilization
Used blocks . . . . . : 53
Used extents . . . . . : 3

SMS Compressible : NO

Creation date . . . . : 2011/08/08
Expiration date . . . : ***None***
Referenced date . . . : 2011/08/08

MA  b 02/015
Conectado a un sistema principal/servidor remoto 192.168.254.254 utilizando una agrupación/lu SCOTCP0

```

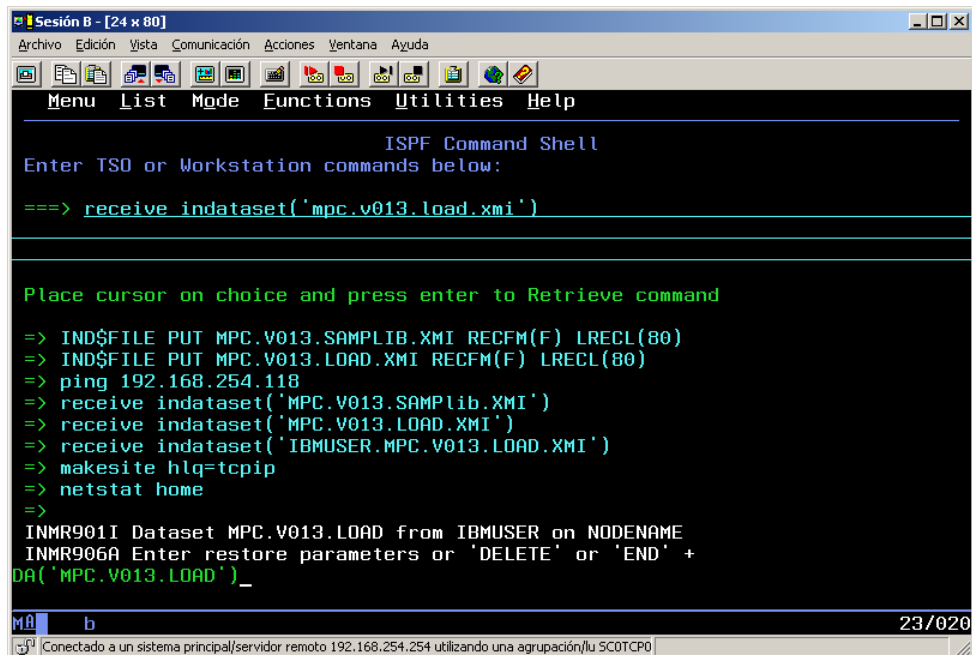
Fig. 6: Tipo y disposición de los datasets correcta

6.- En este punto, descomprimiremos el fichero XMI con el siguiente mandato (dentro de la opción 6 – **Commands** del ISPF):

RECEIVE INDATASET('MPC.V013.LOAD.XMI')

Con lo que nos preguntará los parámetros de restauración, a lo que responderemos:

DA('MPC.V013.LOAD')



```
Sesión B - [24 x 80]
Archivo Edición Vista Comunicación Acciones Ventana Ayuda
Menu List Mode Functions Utilities Help

ISPF Command Shell
Enter TSO or Workstation commands below:

==> receive indataset('mpc.v013.load.xmi')

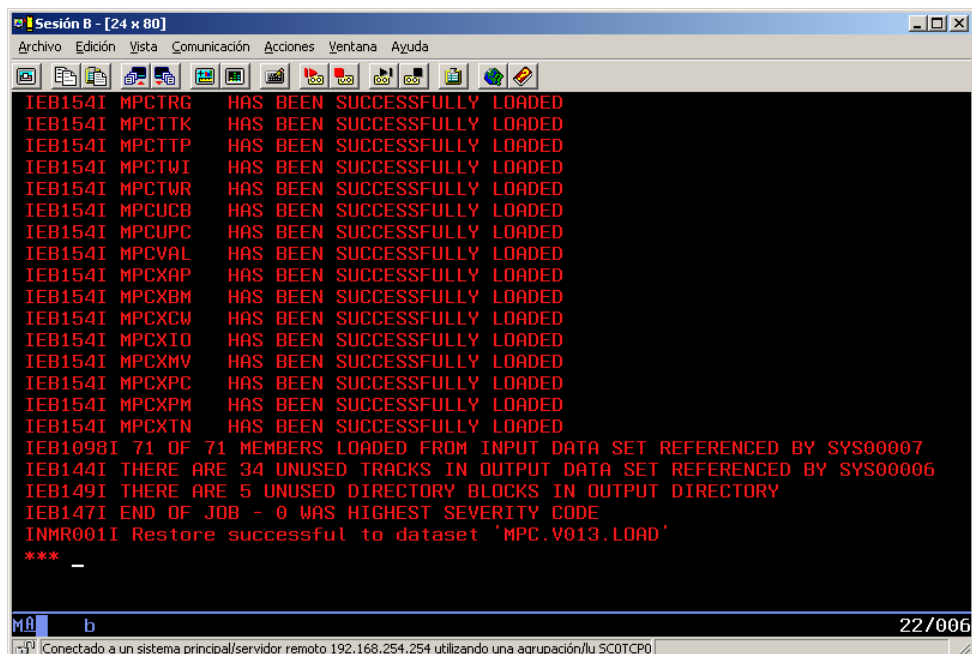
Place cursor on choice and press enter to Retrieve command

=> IND$FILE PUT MPC.V013.SAMPLIB.XMI RECFM(F) LRECL(80)
=> IND$FILE PUT MPC.V013.LOAD.XMI RECFM(F) LRECL(80)
=> ping 192.168.254.118
=> receive indataset('MPC.V013.SAMPLIB.XMI')
=> receive indataset('MPC.V013.LOAD.XMI')
=> receive indataset('IBMUSER.MPC.V013.LOAD.XMI')
=> makesite hlq=tcPIP
=> netstat home
=>
INMR901I Dataset MPC.V013.LOAD from IBMUSER on NODENAME
INMR906A Enter restore parameters or 'DELETE' or 'END' +
DA('MPC.V013.LOAD')_

M b 23/020
Conectado a un sistema principal/servidor remoto 192.168.254.254 utilizando una agrupación/lu SCOTCP0
```

Fig. 7: Descompresión del XMI

Esto hará que el sistema restaure el dataset **MPC.V013.LOAD.XMI** y todo su contenido lo deje dentro de **MPC.V013.LOAD**



```
Sesión B - [24 x 80]
Archivo Edición Vista Comunicación Acciones Ventana Ayuda
Menu List Mode Functions Utilities Help

IEB154I MPCTRG HAS BEEN SUCCESSFULLY LOADED
IEB154I MPCTTK HAS BEEN SUCCESSFULLY LOADED
IEB154I MPCTTP HAS BEEN SUCCESSFULLY LOADED
IEB154I MPCTWI HAS BEEN SUCCESSFULLY LOADED
IEB154I MPCTWR HAS BEEN SUCCESSFULLY LOADED
IEB154I MPCUCB HAS BEEN SUCCESSFULLY LOADED
IEB154I MPCUPC HAS BEEN SUCCESSFULLY LOADED
IEB154I MPCVAL HAS BEEN SUCCESSFULLY LOADED
IEB154I MPCXAP HAS BEEN SUCCESSFULLY LOADED
IEB154I MPCXBM HAS BEEN SUCCESSFULLY LOADED
IEB154I MPCXCW HAS BEEN SUCCESSFULLY LOADED
IEB154I MPCXIO HAS BEEN SUCCESSFULLY LOADED
IEB154I MPCXMV HAS BEEN SUCCESSFULLY LOADED
IEB154I MPCXPC HAS BEEN SUCCESSFULLY LOADED
IEB154I MPCXPM HAS BEEN SUCCESSFULLY LOADED
IEB154I MPCXTN HAS BEEN SUCCESSFULLY LOADED
IEB1098I 71 OF 71 MEMBERS LOADED FROM INPUT DATA SET REFERENCED BY SYS00007
IEB144I THERE ARE 34 UNUSED TRACKS IN OUTPUT DATA SET REFERENCED BY SYS00006
IEB149I THERE ARE 5 UNUSED DIRECTORY BLOCKS IN OUTPUT DIRECTORY
IEB147I END OF JOB - 0 WAS HIGHEST SEVERITY CODE
INMR001I Restore successful to dataset 'MPC.V013.LOAD'
***
-

M b 22/006
Conectado a un sistema principal/servidor remoto 192.168.254.254 utilizando una agrupación/lu SCOTCP0
```

Fig. 8: Restauración correcta

7.- Realizaremos la misma operación con el otro dataset **MPC.V013.SAMPLIB.XMI**, dejando su contenido en **MPC.V013.SAMPLIB** y comprobaremos que todo está correcto desde la opción 3.4 del ISPF:

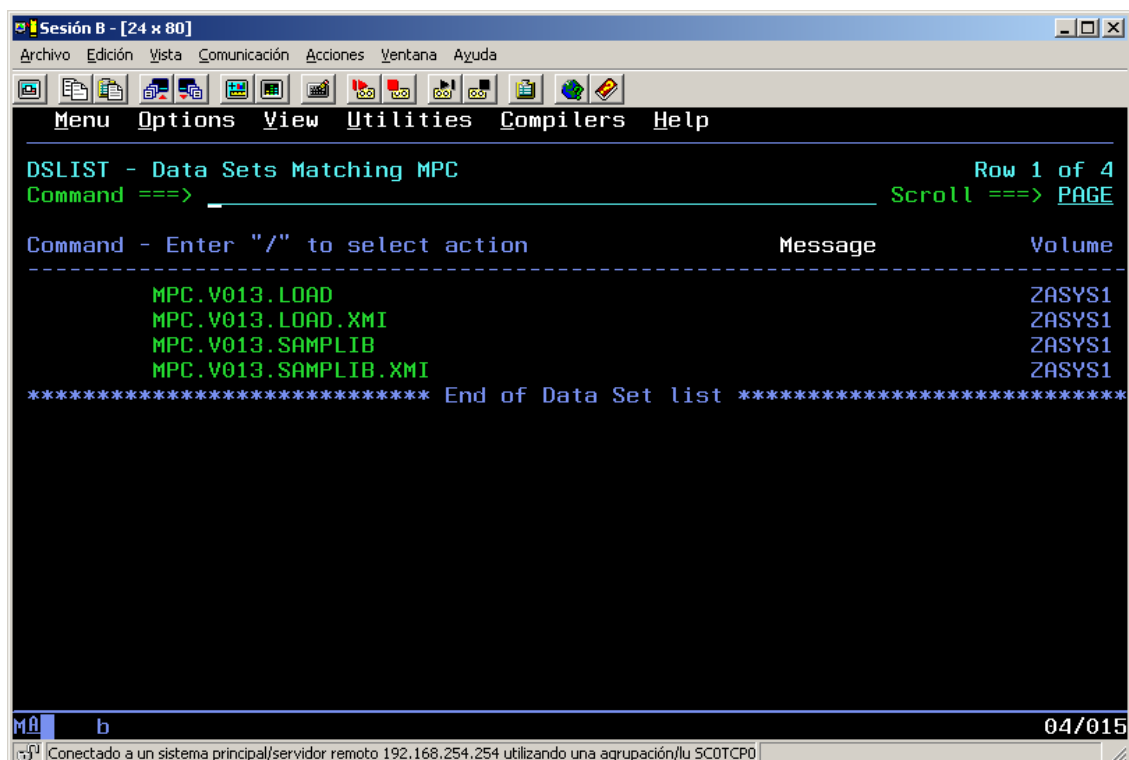


Fig. 9: Datasets restaurados.

8.- El siguiente paso es autorizar en APF la nueva librería LOAD de MFNetDisk, en este caso, **MPC.V013.LOAD**. Para ello, iremos a nuestra **PARMLIB** y editaremos el correspondiente miembro **PROGXX** añadiendo lo siguiente:

```
APF ADD
      DSNAME(MPC.V013.LOAD)                VOLUME(ZASYS1)
```

9.- Si hacemos desde la master console un **SET PROG=XX** (siendo XX el miembro activo que tengamos), actualizará la lista de autorizaciones y veremos con el siguiente mensaje su correcta actualización:

```
0090 CSV410I DATA SET CEE.SCEERUN ON VOLUME ZARES2 ADDED TO APF LIST
0090 CSV410I DATA SET SYS1.SCUNIMG ON VOLUME ZARES1 ADDED TO APF LIST
0090 CSV410I DATA SET MPC.V013.LOAD ON VOLUME ZASYS1 ADDED TO APF LIST
```

Fig 10: Lista APF actualizada con el nuevo dataset

10.- El siguiente punto es crear la **STARTED TASK** que pondrá en marcha el tinglado MFNetDisk. Para ello, copiaremos el miembro **MPCPROC** de la **MPC.V013.SAMPLIB** y lo dejaremos en nuestra **PROCLIB**, renombrándolo de MPCPROC a **MPC**. Una vez copiado, lo editaremos con los valores correctos de los datasets en la parte de **STEPLIB**, **MPCLOG** y **PARMDD**:

```

Sesión B - [24 x 80]
Archivo Edición Vista Comunicación Acciones Ventana Ayuda
File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT      ADCD.Z110.PROCLIB(MPC) - 01.04          Columns 00001 00072
Command ==>                                     Scroll ==> CSR_

***** Top of Data *****
000001 //MPCPROC  PROC PRM=WARM
000002 //*
000003 //*
000004 //MPC010   EXEC PGM=MPCMSR,PARM='&PRM',
000005 //          TIME=1440,
000006 //          REGION=0M
000007 //*
000008 //STEPLIB   DD DISP=SHR,DSN=MPC.V013.LOAD
000009 //MPCLIST   DD SYSOUT=*,OUTLIM=16777215
000010 //MPCSOUP   DD SYSOUT=*,OUTLIM=16777215
000011 //MPCTRC   DD SYSOUT=*,OUTLIM=16777215
000012 //SYSABEND DD SYSOUT=*,OUTLIM=16777215
000013 //SYSPRINT DD SYSOUT=*,OUTLIM=16777215
000014 //MPCLOG   DD DSN=MPC.V013.MPCLOG,DISP=SHR
000015 //PARMDD   DD DSN=MPC.V013.SAMPLIB(MPCOPT),
000016 //          DISP=SHR
***** Bottom of Data *****

M  b 04/015
Conectado a un sistema principal/servidor remoto 192.168.254.254 utilizando una agrupación/lu SCOTCP0

```

Fig. 11: Aspecto final de la started task MPC

11.- Y por último, crearemos con el JCL que viene en **MPC.V013.SAMPLIB(ALCLOG)** el log que el MFNetDisk va a usar, y lo submitiremos.

```

Sesión B - [24 x 80]
Archivo Edición Vista Comunicación Acciones Ventana Ayuda
File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT      MPC.V013.SAMPLIB(ALCLOG) - 01.05        Columns 00001 00072
Command ==>                                     Scroll ==> CSR_

***** Top of Data *****
000001 //MPCLOGA   JOB 0,AAA,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
000002 //* ALLOC MPCLOG FILE USING BY MFNETDISK MVS
000003 //* AFTER THIS, START THE MPC WITH PRM=COLD
000004 //ALCLOG   EXEC PGM=IEBGENER
000005 //SYSUT1    DD DUMMY,
000006 //          DCB=(RECFM=F,LRECL=27000,BLKSIZE=27000,DSORG=PS)
000007 //SYSUT2    DD DISP=(,CATLG),DSN=MPC.V013.MPCLOG,
000008 //          DCB=(RECFM=F,LRECL=27000,BLKSIZE=27000,DSORG=PS),
000009 //          UNIT=3390,VOL=SER=ZASYS1,SPACE=(CYL,(80,10))
000010 //SYSPRINT  DD SYSOUT=*
000011 //SYSIN     DD DUMMY
***** Bottom of Data *****

M  b 04/015
Conectado a un sistema principal/servidor remoto 192.168.254.254 utilizando una agrupación/lu SCOTCP0

```

Fig. 12: Job ALCLOG

Con estos pasos, tenemos el sistema listo para poder parametrizarlo.

1.2.2 Parametrización en z/OS

En este punto explicare como parametrizar en la parte z/OS nuestro MFNetDisk. Para ello, lo primero que debemos tener en mente es la ARQUITECTURA de la solución, y sus nomenclaturas.

Para ello, en mi instalación se que direcciones y nomenclaturas usar, y máxime cuando existe en la actualidad robóticas que están en producción en la misma instalación. A este respecto, explicaré como se instalaría desde cero todo el entorno SMS, pero lo hare más adelante, ahora lo más importante es tener en mente lo que necesitaremos para parametrizar nuestra robótica virtual:

- Por una parte, las direcciones de los drives virtuales que queremos habilitar.

En mi caso, quiero habilitar dos tipos de direcciones, un rango para robótica virtual y otro rango para drives standalone. He elegido en mi instalación el rango 580-580F para la robótica virtual con 16 drives 3490E de 36 pistas y el rango 590-59F para 16 drives standalone 3490E.

- Y por otra, el rango de volúmenes de cinta que utilizará el sistema. En mi ejemplo, A00000 en adelante para la robótica virtual y B00000 en adelante para drives standalone.

En este contexto, cabe también reseñar la IP de la máquina Windows que va a correr el MFNetDisk, para que el z/OS sepa DONDE conectarse para hacer uso de la virtualización del MFNetDisk.

Con estos datos, editaremos el dataset **MPC.V013.SAMPLIB** y modificaremos el miembro **MPCOPT** comentando todas las líneas con asteriscos, excepto las de la siguiente figura:

```
Sesión B - [24 x 80]
Archivo Edición Vista Comunicación Acciones Ventana Ayuda
File Edit Edit Settings Menu Utilities Compilers Test Help

EDIT MPC.V013.SAMPLIB(MPCOPT) - 01.99 Columns 00001 00072
Command ==> Scroll ==> CSR
000018 SERVER TASKS=4
000019 **TAPE SET IF MFNETDISK CONTROL SCRATCH REQUESTS (MFNETDISK TAPE MNGR)
000020 TAPE CTLSCR=Y
000021 **ALIASUCB IN HCD SELECT 3390A (3390 ALIAS)
000022 *ALIASUCB MVSDVNUM=0380,RANGE=32
000023 * NO MORE SUPPORT FOR PORT=11000.
000024 TAPEDRV MVSDVNUM=0580,RANGE=8 3490E MTL
000025 TAPEDRV MVSDVNUM=0590,RANGE=8 3490E NOMTL
000026 **TAPELOC-WHAT PC HANDLE THE TAPES.
000027 TAPELOC VOL=A00000,COUNT=32, MPT000-MPT031 FOR MTL
000028 PCIP=192.168.254.118,SCRATCH=Y
000029 TAPELOC VOL=B00000,COUNT=32, MPT032-MPT063 FOR NONMTL
000030 PCIP=192.168.254.118,SCRATCH=Y
000031 *TAPELOC VOL=MPX000,COUNT=32, MPT032-MPT063 FOR NONMTL
000032 * PCIP=169.254.0.1,SCRATCH=Y
000033 *TAPELOC VOL=M267AA,COUNT=1, MPT032-MPT063 FOR NONMTL
000034 * PCIP=169.254.0.1,SCRATCH=N
000035 *TAPELOC VOL=TEST00,COUNT=1, TAPE FROM OUTSIDE
000036 * PCIP=169.254.0.1,SCRATCH=N
000037 **NONSMS TAPE
MA b 19/009
Conectado a un sistema principal/servidor remoto 192.168.254.254 utilizando una agrupación/lu SCOTCP0
```

Fig. 13: Parametrización de MFNetDisk

TAPEDRV Especificará el número de dispositivos de cinta que MFNetDisk va a utilizar, en mi caso, 580 y 590.

TAPELOC especifica el número de volúmenes que queremos que el mainframe utilice, de momento, la A00000 hasta la A00031 porque solo he definido 32 cintas. **PCIP** es la IP donde hemos instalado el MFNetDisk en nuestra máquina Windows, en mi caso el servidor tiene la IP 192.168.254.118.

Y con esto, la parte z/OS de instalación y configuración estaría terminada.

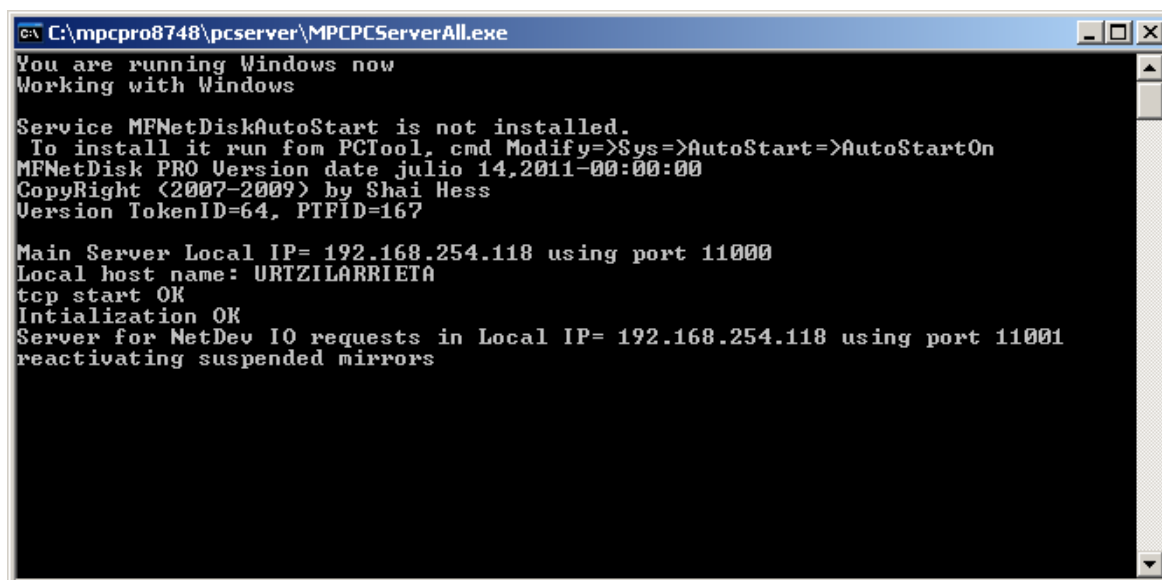
1.2.3 Instalación en el servidor Windows

Con la instalación z/OS terminada, vamos a proceder a configurar el otro extremo, es decir, nuestra máquina Windows con mucho disco duro, que será el que sirva el almacenamiento en disco y cinta virtual, en nuestro caso concreto, nuestras cintas.

La instalación es de lo más sencilla: De hecho, ya la hemos realizado cuando hemos descomprimido el ZIP descargado de <http://www.mfnetdisk.com> por lo que el sistema ya está listo para parametrizarlo y ejecutarlo.

Con el ZIP descomprimido, se crean una serie de carpetas, pero comentaré las mas importantes:

PCSERVER: Carpeta que contiene 2 ejecutables, **MPCPCServerAll.EXE** y **MPCAutoStart.EXE**. Este último se utiliza porque existe la posibilidad de instalar el MFNetDisk como servicio, así que el importante es **MPCPCServerAll.EXE**. Si lo ejecutamos, saldrá una ventana como la siguiente:



```
C:\mpcpro8748\pcserver\MPCPCServerAll.exe
You are running Windows now
Working with Windows

Service MFNetDiskAutoStart is not installed.
To install it run fom PCTool. cmd Modify=>Sys=>AutoStart=>AutoStartOn
MFNetDisk PRO Version date julio 14,2011-00:00:00
CopyRight (2007-2009) by Shai Hess
Version TokenID=64, PTFID=167

Main Server Local IP= 192.168.254.118 using port 11000
Local host name: URTZILARRIETA
tcp start OK
Intialization OK
Server for NetDev IO requests in Local IP= 192.168.254.118 using port 11001
reactivating suspended mirrors
```

Fig. 14: PCServer lanzado con éxito.

NOTA: La primera vez nos preguntará que no encuentra ningún fichero de configuración, así que le responderemos un **Y** (Yes) para que lo cree.

A partir de este momento, podemos comenzar la parametrización del MFNetDisk.

1.2.4 Parametrización en el servidor Windows PCServer

Para parametrizar el MPCNetDisk en la parte Windows, es **OBLIGATORIO** tener funcionando el MPCPCServer. Con este detalle tenido en cuenta, nos iremos a la carpeta PCTOOL y lanzaremos el ejecutable de su interior, el **MPCPCTOOL.EXE**, que nos generará una ventana como la de la siguiente figura:

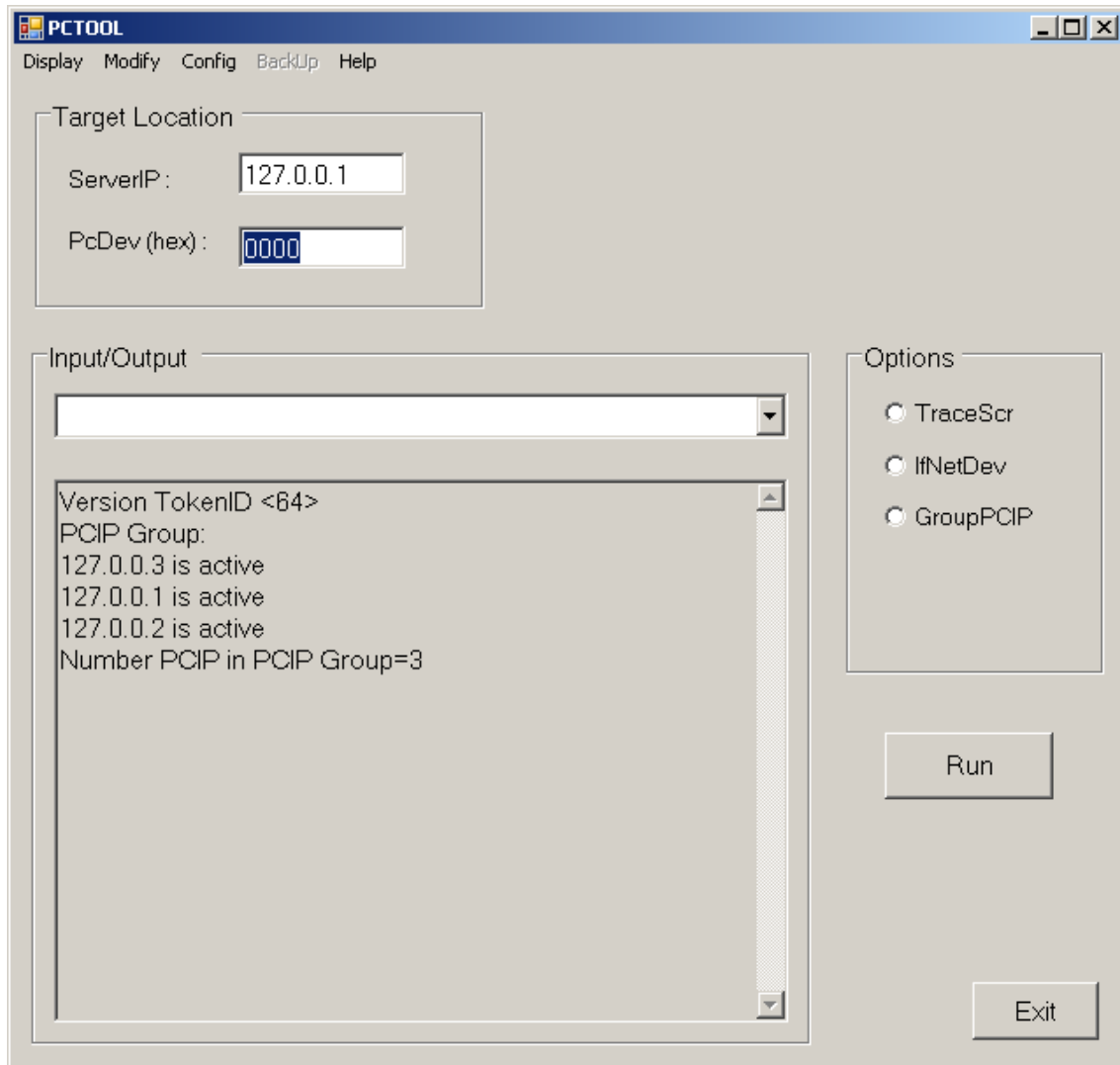


Fig. 15: Ventana inicial del PCTOOL, configurador del MPCNetDisk

Una vez llegados a este punto, comenzaremos a parametrizar el MFNetDisk, de acuerdo a la configuración que previamente hemos creado en el z/OS. Para ello, seguiremos los siguientes pasos:

1.- Crearemos el espacio donde se guardaran las cintas en el disco duro de nuestro servidor. Para ello, nos crearemos una carpeta en nuestro disco duro y le diremos al PCTOOL que la use. Por tanto, Desde el PCTOOL, navegaremos por el menú **MODIFY -> SYS -> PCSTORAGE -> ADD PCSTORAGE** y en la línea de comandos que genere, le ponemos la

carpeta creada anteriormente. En mi caso, la carpeta creada se llama **MPCSTORAGE** y está en el disco **C**. Por tanto, el comando a generar debería ser el siguiente:

cmd=FAddPCStorage,path=C:\MPCSTORAGE

Una vez comprobado que el comando es el correcto, hacemos clic sobre el botón RUN, por lo que este comando se enviará al PCServer y creará la configuración oportuna. A partir de este momento, cualquier objeto (discos, cintas, etc) que utilice el MFNetDisk se almacenará en esta carpeta.

2.- Según el miembro MPCOPT de la SAMPLIB del z/OS, hemos definido dos rangos de cintas: A00000 en adelante con 32 cintas iniciales para que las use en nuestra robótica virtual y B00000 en adelante con 32 cintas iniciales para montajes manuales. Evidentemente, este rango de cintas también debe ser definido en el MFNetDisk, por lo que desde el PCTool, nos iremos a **CONFIG -> TAPE -> TAPESRANGE -> ADD TAPESRANGE -> MTL** para añadir cintas virtuales que use nuestra Robótica y **CONFIG -> TAPE -> TAPESRANGE -> ADD TAPESRANGE -> No MTL** para añadir rangos de cinta que no serán de robótica virtual, sino manual. Los comandos generados en cada caso serían los siguientes:

cmd=AddNewMTLTapesRange,mtllib=MPCROBOT,vol=A00000,count=32

siendo **MPCROBOT** el nombre de nuestra librería virtual que luego definiremos y

cmd=AddNewNonMTLTapesRange,vol=B00000,count=32

Si hacemos clic sobre RUN en la generación de cada comando, el PCServer tendrá la misma configuración que nuestro miembro MPCOPT. **ATENCIÓN A ESTO**, si cualquiera de los miembros se modifica, se debe hacer lo mismo con los mismos datos a modificar en el PCServer.

3.- Habilitaremos el SCRATCH Manager para que MFNetDisk sepa como comportarse ante una petición de cintas SCRATCH por parte del z/OS, para ello, nos iremos a **MODIFY -> TAPE -> SCRATCH MANAGER -> ENABLE** lo que nos generará un comando tal que:

cmd=FEnableTapeManager y hacemos clic al botón RUN

4.- Verificaremos que está activo todo haciendo un **DISPLAY -> TAPE -> RULES**, generando un comando tal que:

cmd=DTapesRules y dando al botón RUN.

Con estos pasos, acabamos de terminar de parametrizar la parte Windows del MFNetDisk, ergo, la que nos proveerá de todo el almacenamiento necesario.

Pero todavía queda una parte muy importante que definir dentro del z/OS para que todo funcione como es debido, y es la generación del HCD, la puesta en marcha del OAM y la parametrización del SMS para su uso con el RMM, que gestionará nuestras cintas automáticamente dentro de la instalación.

1.3 Configuración SMS, OAM y RMM para nuestro VTS

Hasta este punto, hemos preparado la base en la que se comunicarán nuestro z/OS y nuestro servidor Windows, pero ahora debemos configurar la parte de z/OS que utilizará la robótica para el tratamiento de datos.

A este respecto, todo el tratamiento, disposición y políticas de los datos los lleva todo el subsistema del DFSMS, en su modalidad OAM (Object Access Method), tanto las políticas de disposición SMS, como el RMM (Removable Media Manager).

Pero antes de todo eso, nuestra robótica virtual no existe a menos que la definamos en el HCD.

1.3.1 DEFINICIÓN HCD

En los anteriores puntos hemos decidido que nuestra robótica ocuparía las direcciones 580 a 580F y que los drives manuales ocuparían de la 590 a la 59F. Esto lo debemos definir en el HCD (Hardware Configuration Definition), siguiendo los siguientes pasos:

1.- Desde el ISPF nos iremos al HCD, hasta ver una ventana como la siguiente:

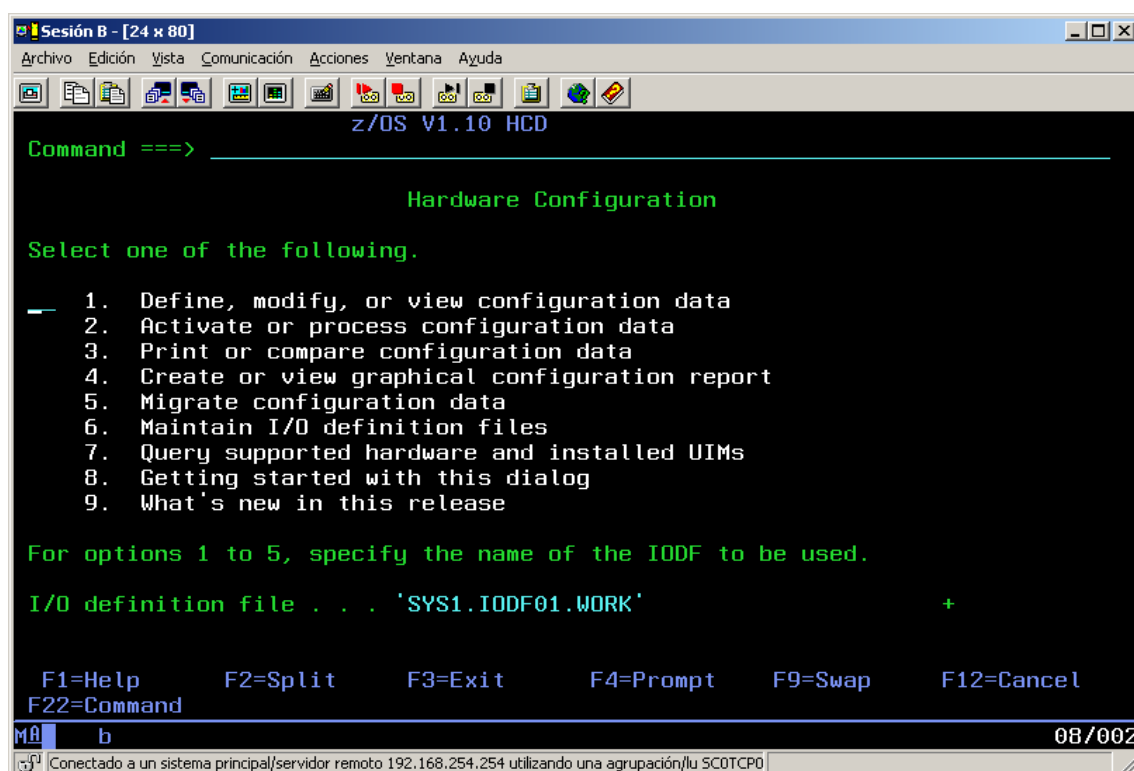


Fig. 16: Ventana inicial del HCD

2.- Iremos a la opción **1. Define, modify, or view configuration data** y de ahí, a la opción **4. Control Units**, donde definiremos 2 Unidades de Control: la **580** y la **590**, que mas tarde controlarán los 16 dispositivos de cada rango. Con **PF11**, añadiremos una nueva U.Control (Fig. 18).

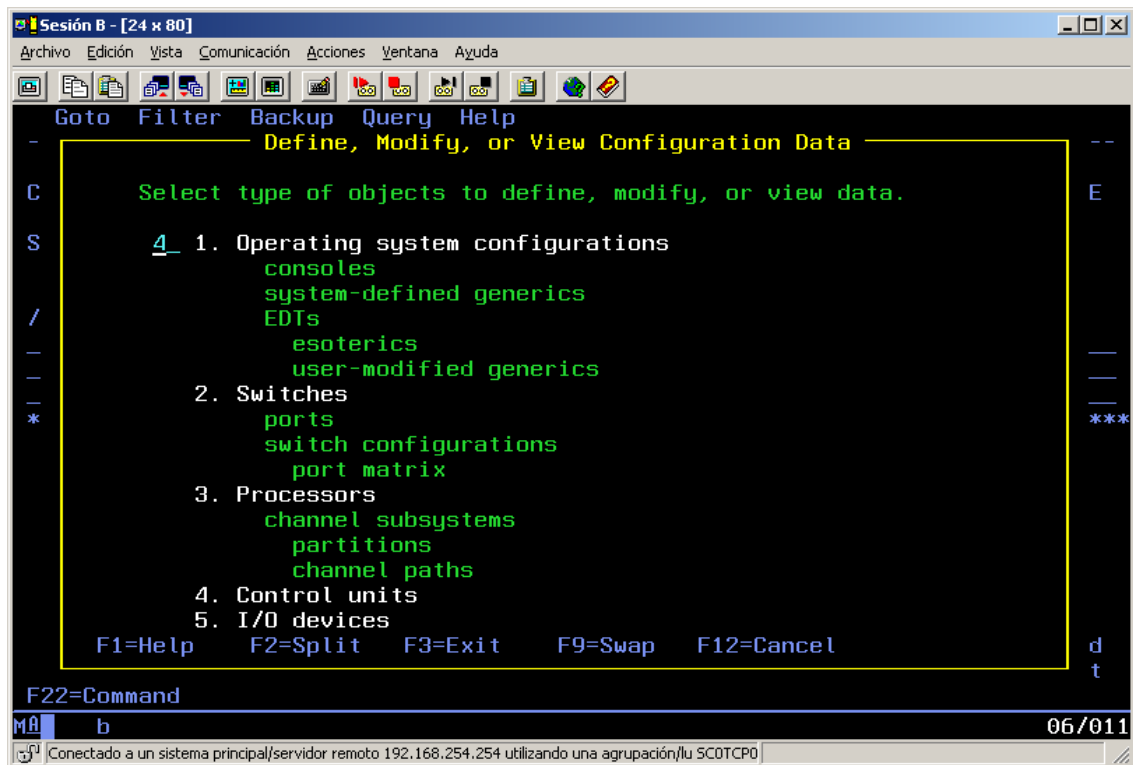


Fig. 17: Control Units

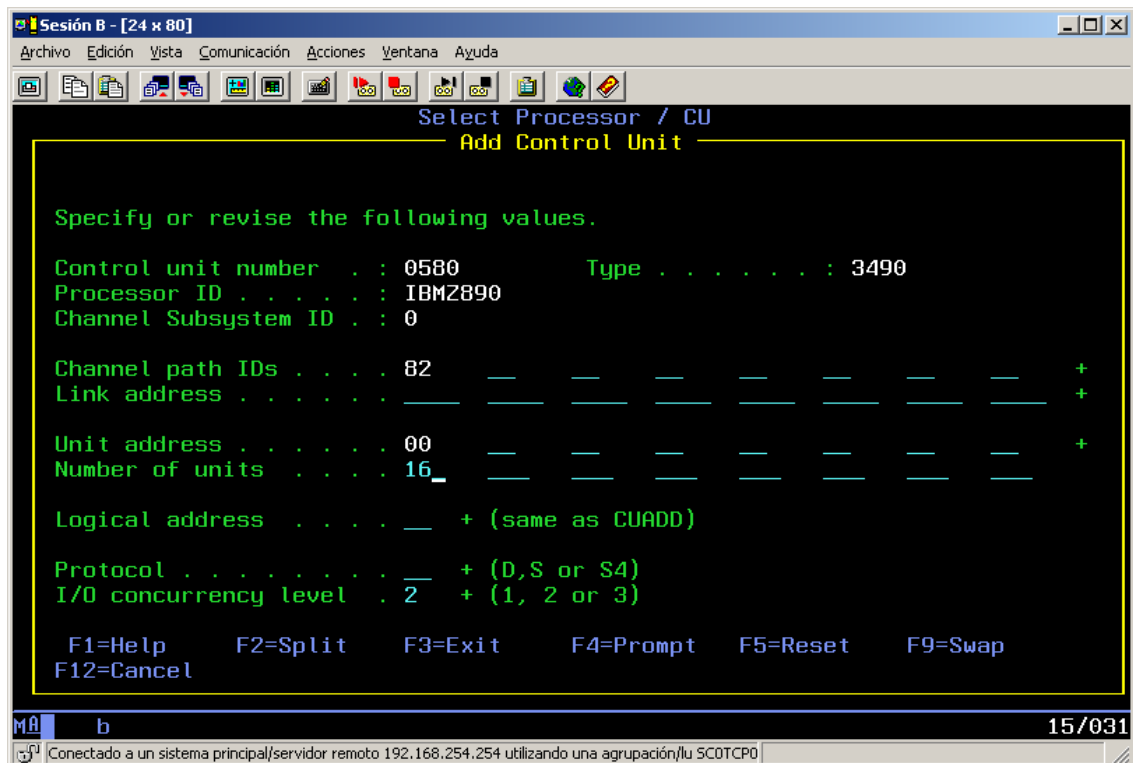


Fig. 18: Creación de la Control Unit 580

3.- Una vez definida, la conectaremos a nuestro host y quedará definida en el S.O.

NOTA: NO es necesario añadir CHPIDs o PCHIDs ya que nuestra librería será virtual.

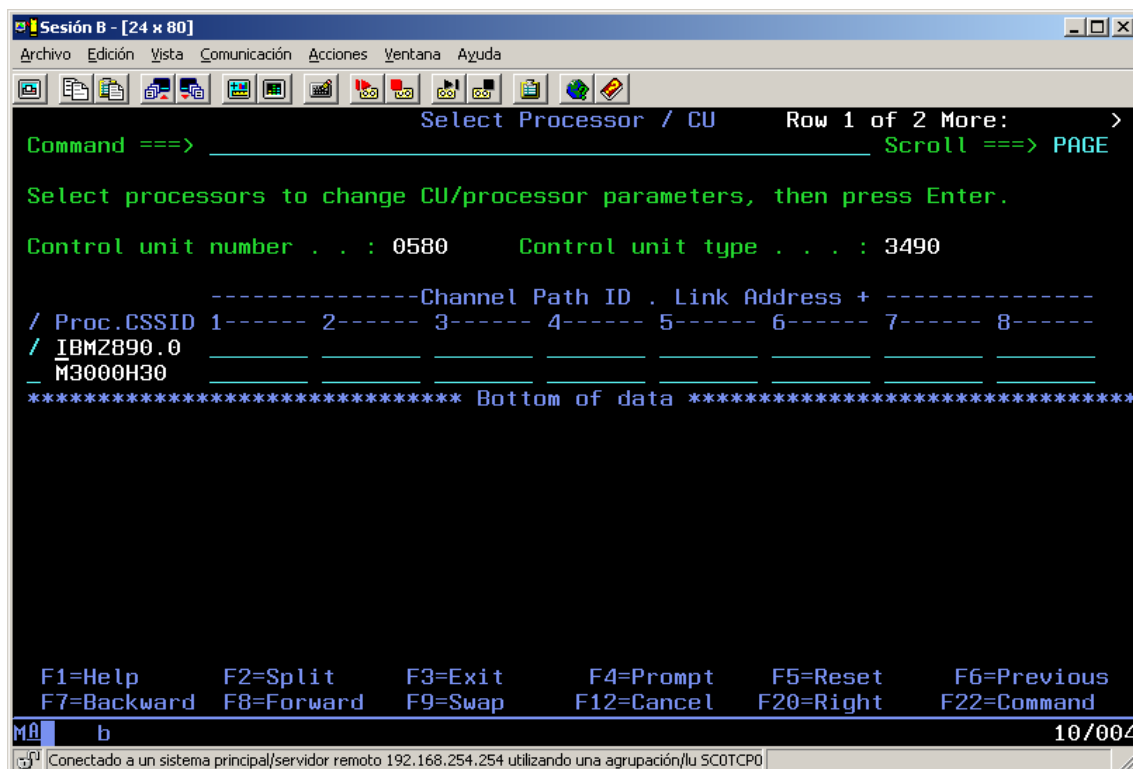


Fig. 19: Elección del Procesador a conectar la U. Control

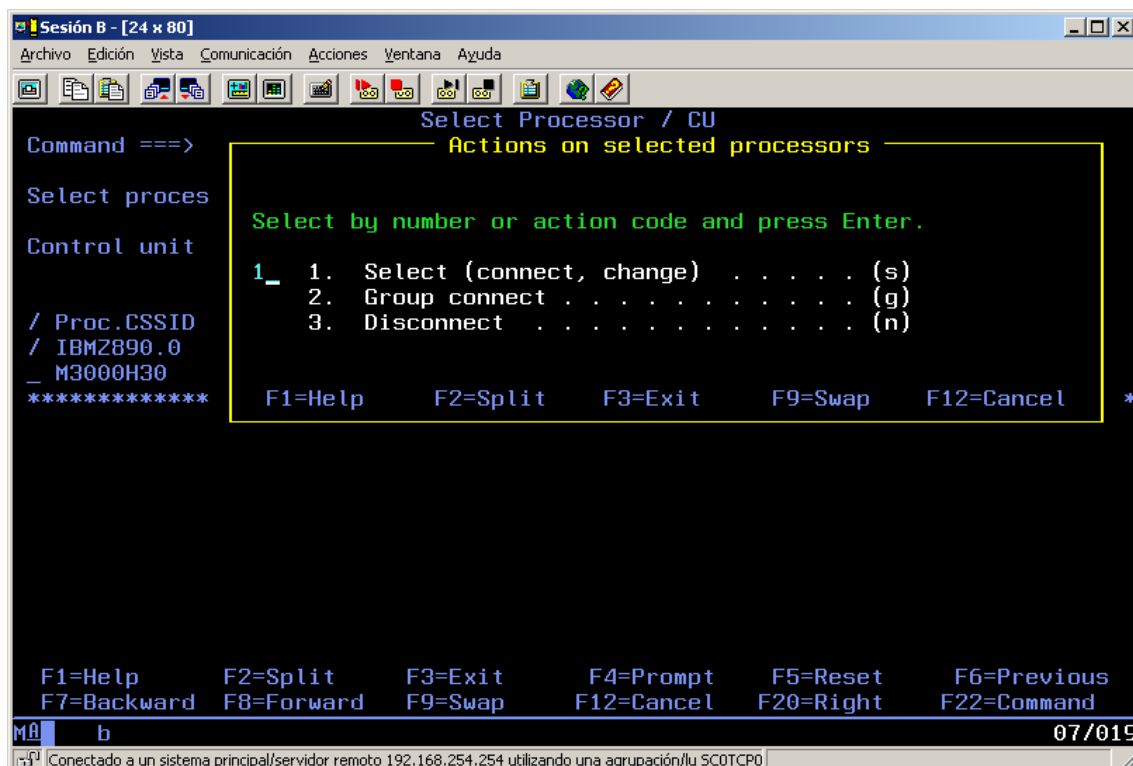


Fig. 20: Conectando la U. Control

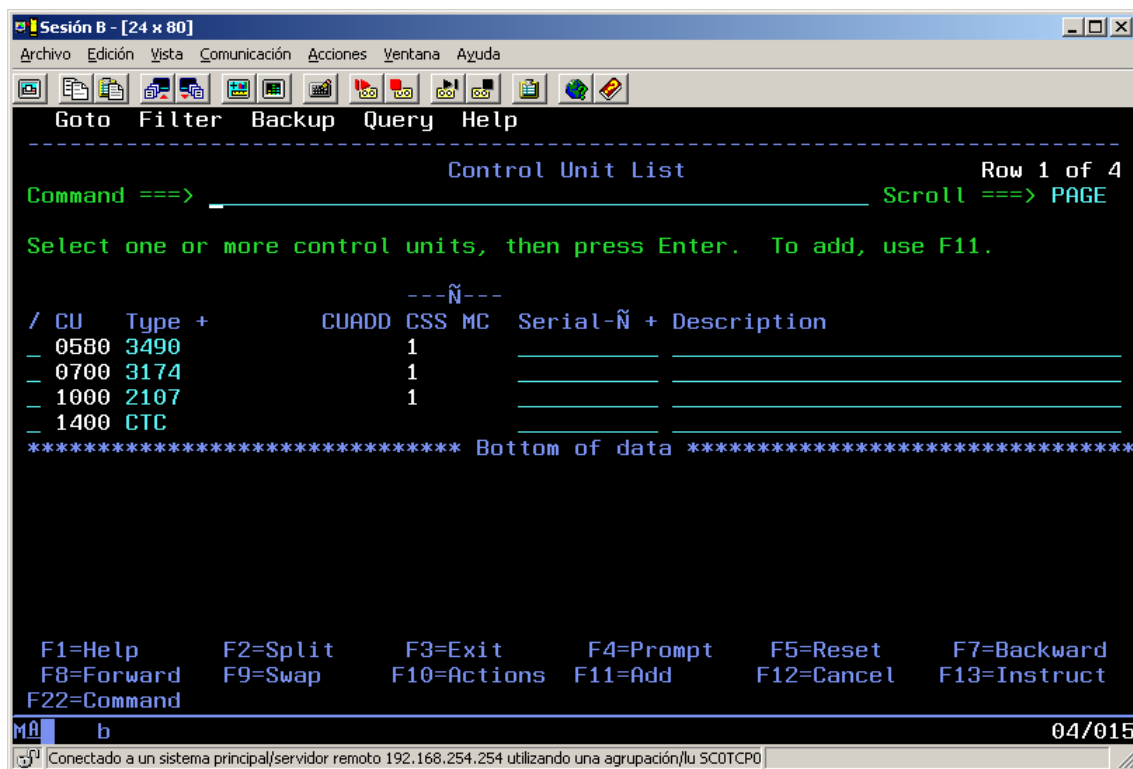


Fig. 21: U. control definida en la lista.

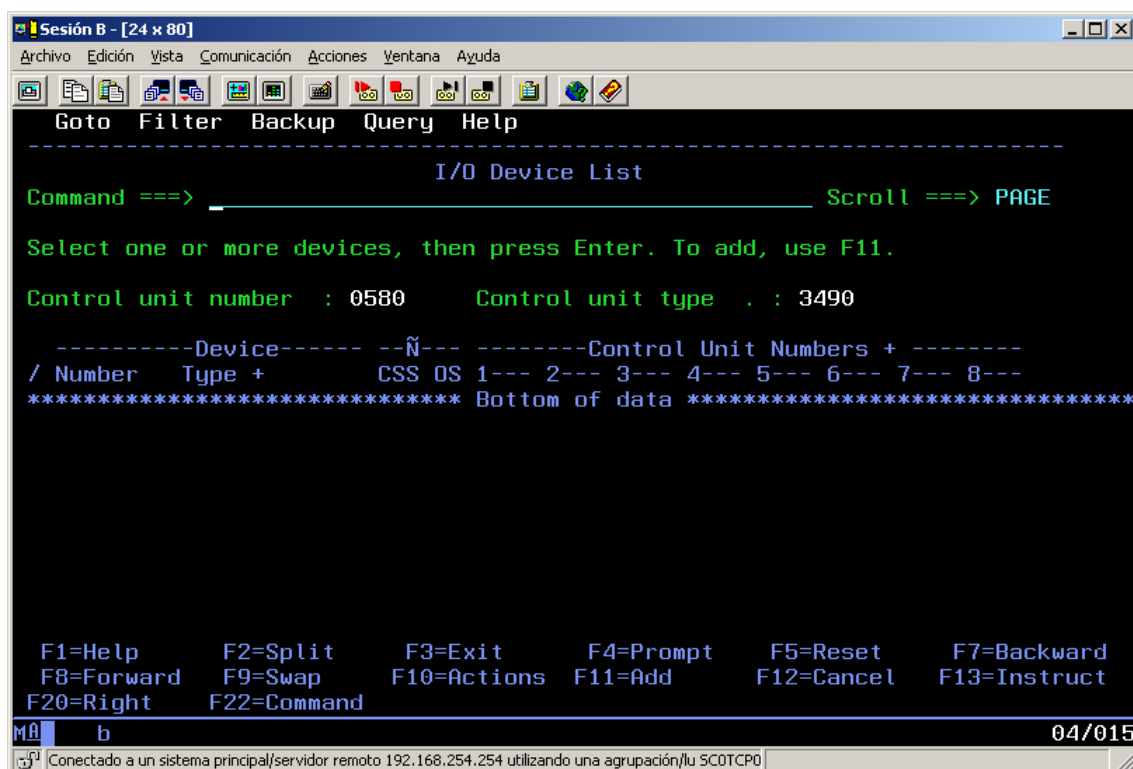


Fig. 22: Añadiendo dispositivos de cinta a nuestra robótica

4.- En este punto, dentro de la U.Control, podemos añadir dispositivos pulsando sobre **PF11**, siguiendo las siguientes figuras:

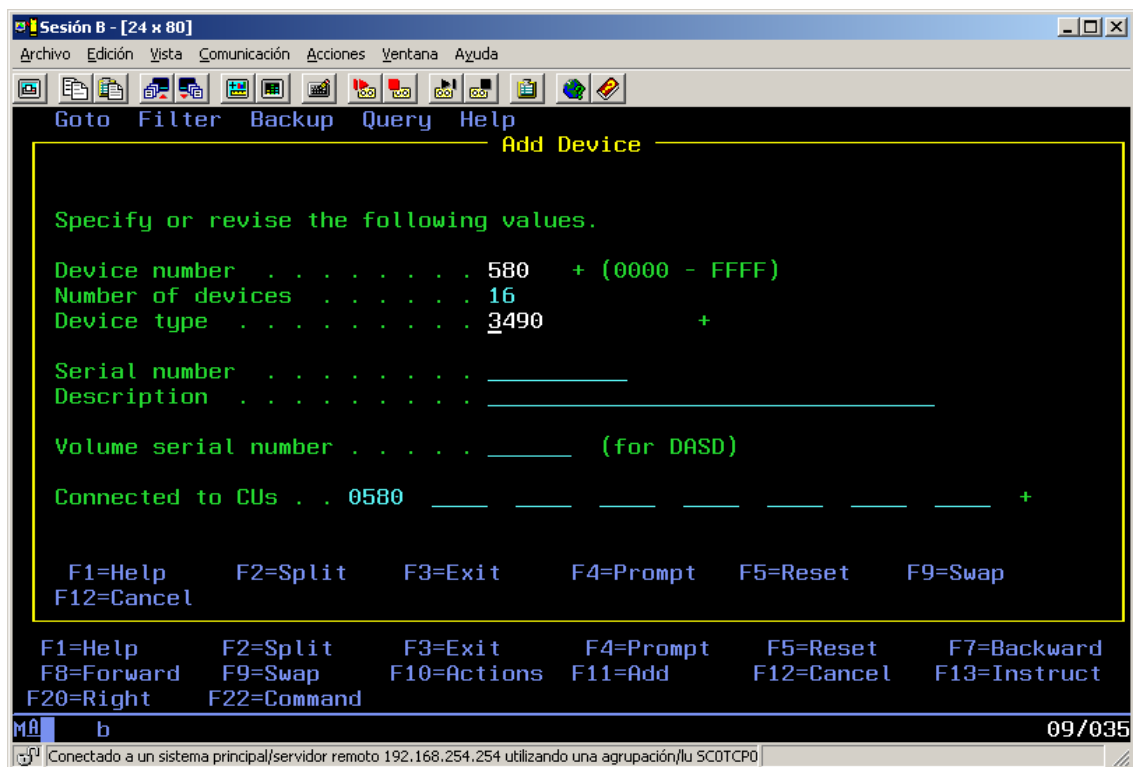


Fig. 23: Creación de los 16 drives 3490

Direcciones de la 580 a la 58F, 16 dispositivos.

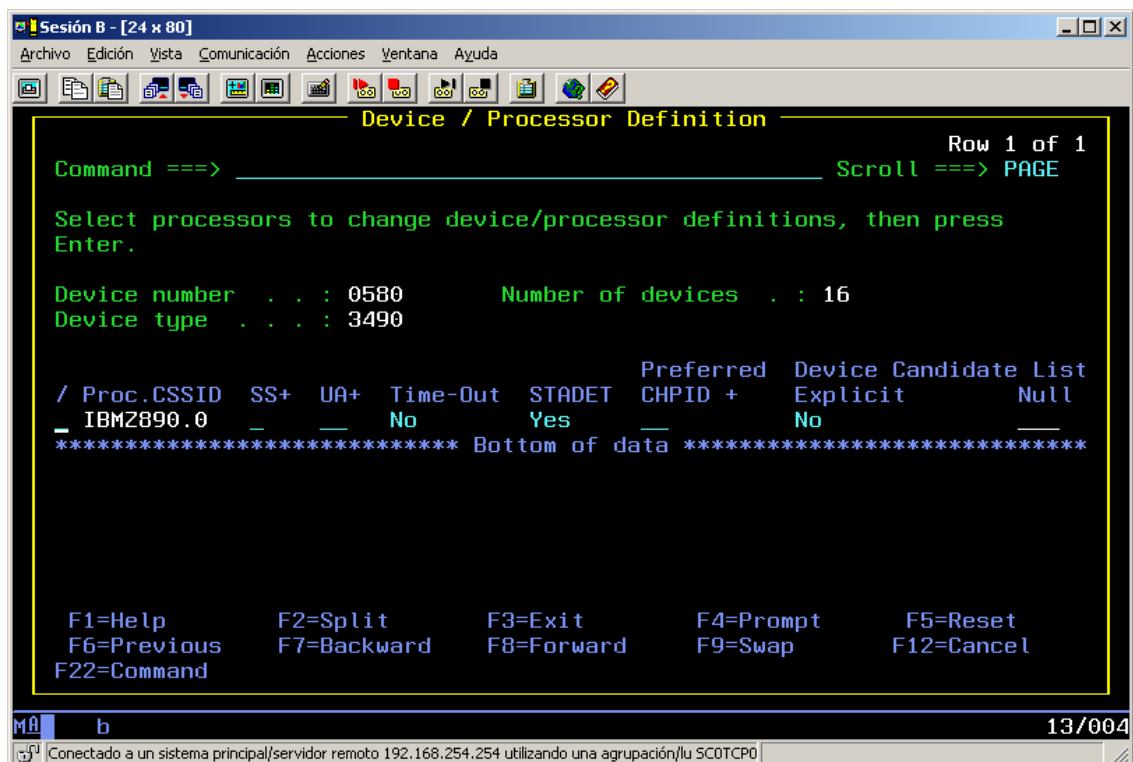


Fig. 24: Conectando los drives a la configuración

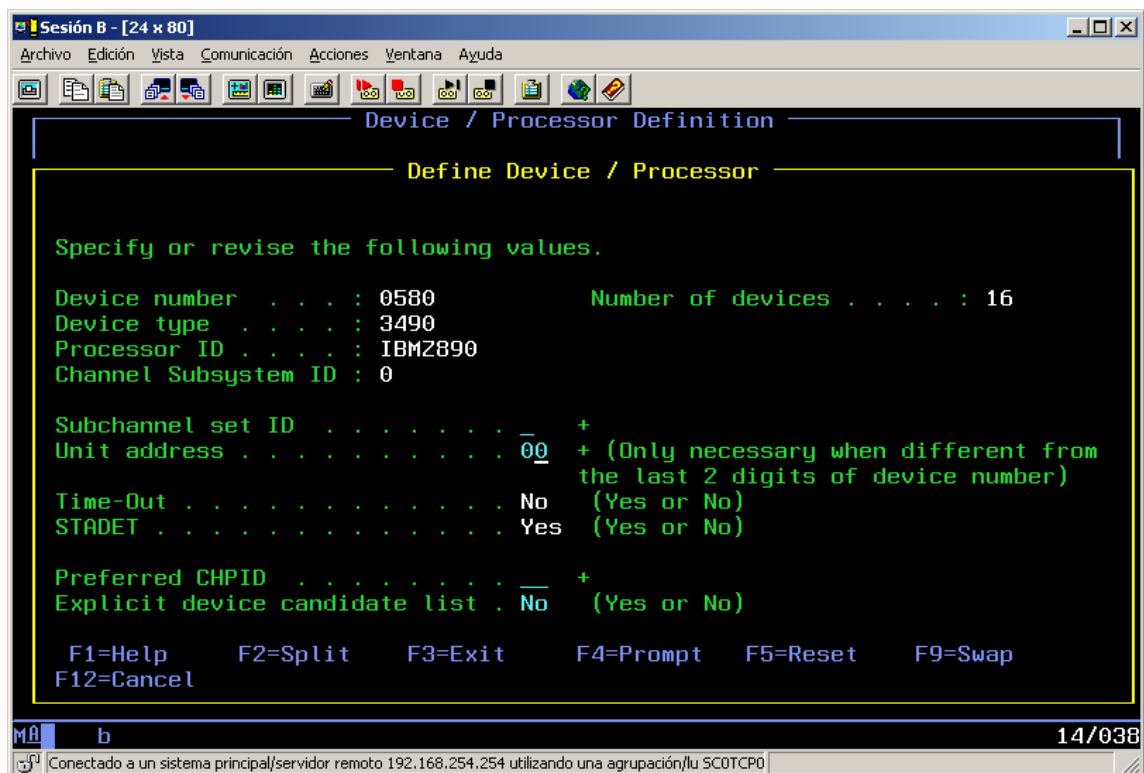


Fig. 25: Definiendo los drives al procesador.

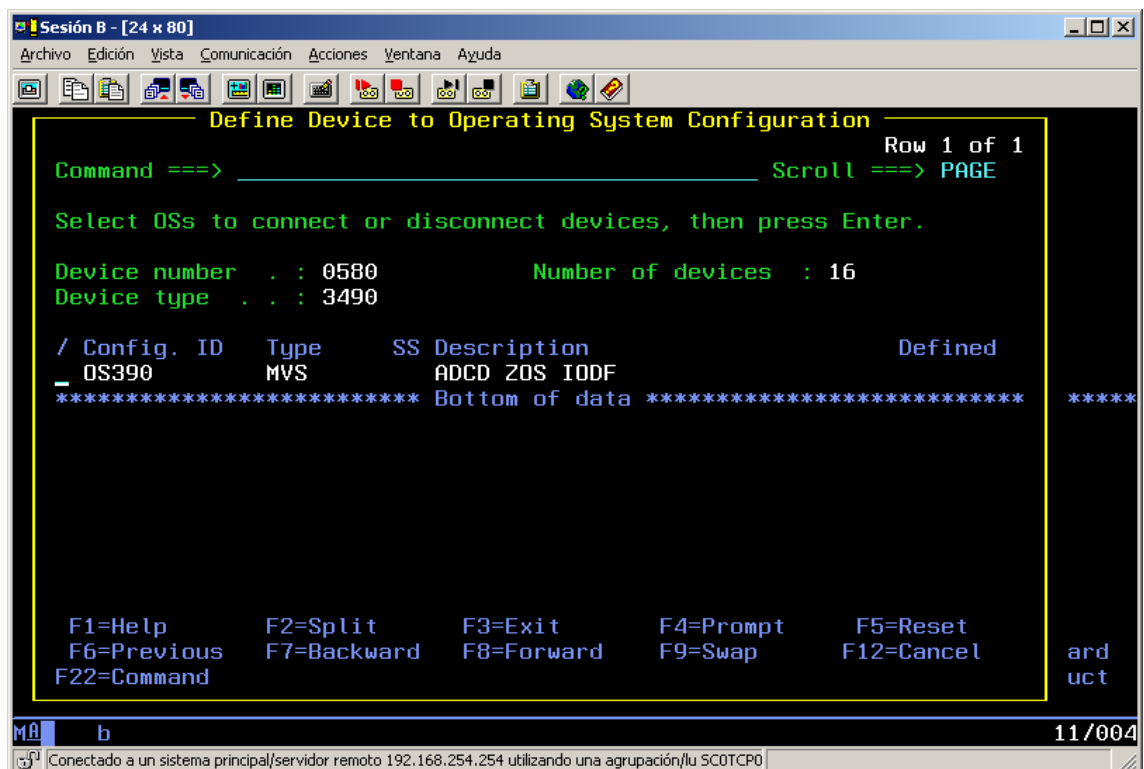


Fig. 26: Conectando los drives al z/OS

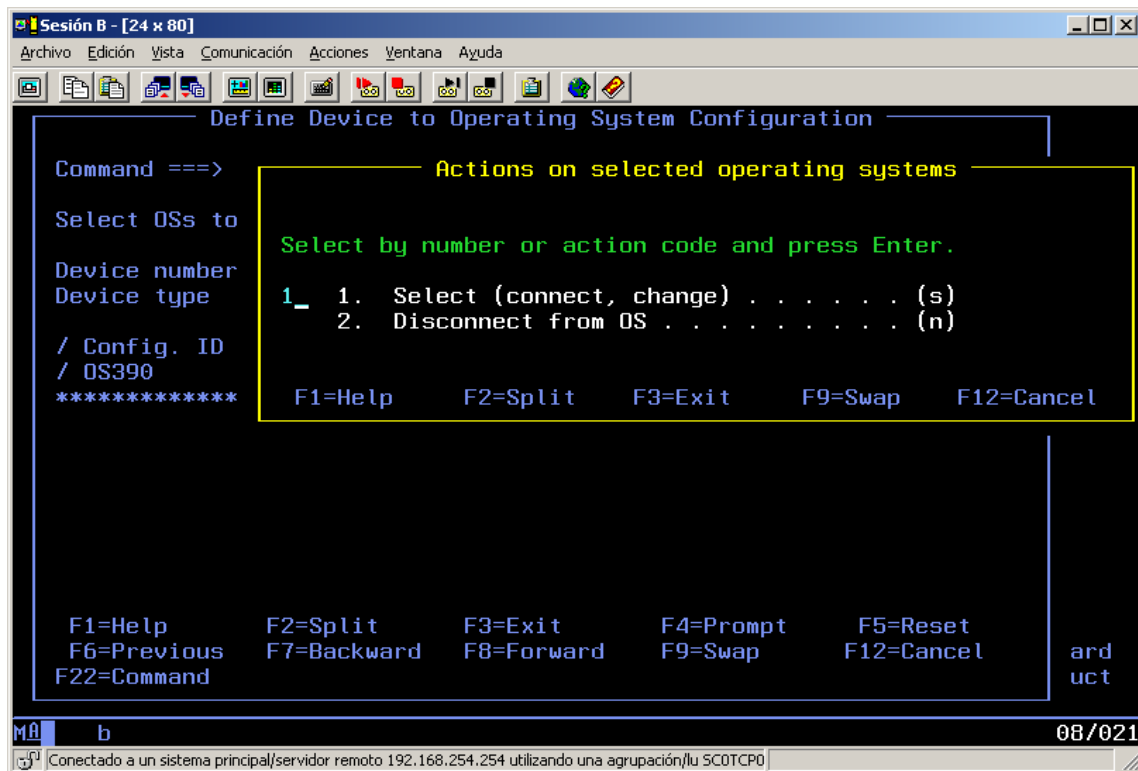


Fig. 27: Seleccionando los drives para el Sistema Operativo.

5.- Es **MUY IMPORTANTE** rellenar los siguientes datos y tenerlos en cuenta para cuando definamos esta robótica en el SMS:

- **OFFLINE** lo pondremos a **YES**, ya que será el MFNetDisk quien ponga on-line el sistema.

- **LIBRARY-ID** elegimos un número de 5 cifras, en mi caso he puesto mi fecha de nacimiento para recordarlo.

- **LIBPORT-ID** lo ponemos a **01**.

- **MTL** lo ponemos a **YES**, ya que será una Manual Tape Library –aunque luego sea virtual-.

Y con esto, lo conectamos el z/OS.

6.- Para la U. Control 590 y sus drives, se siguen exactamente los mismos pasos, pero los drives 590 a 59F no serán MTL ni tendrán LIBRARY-ID.

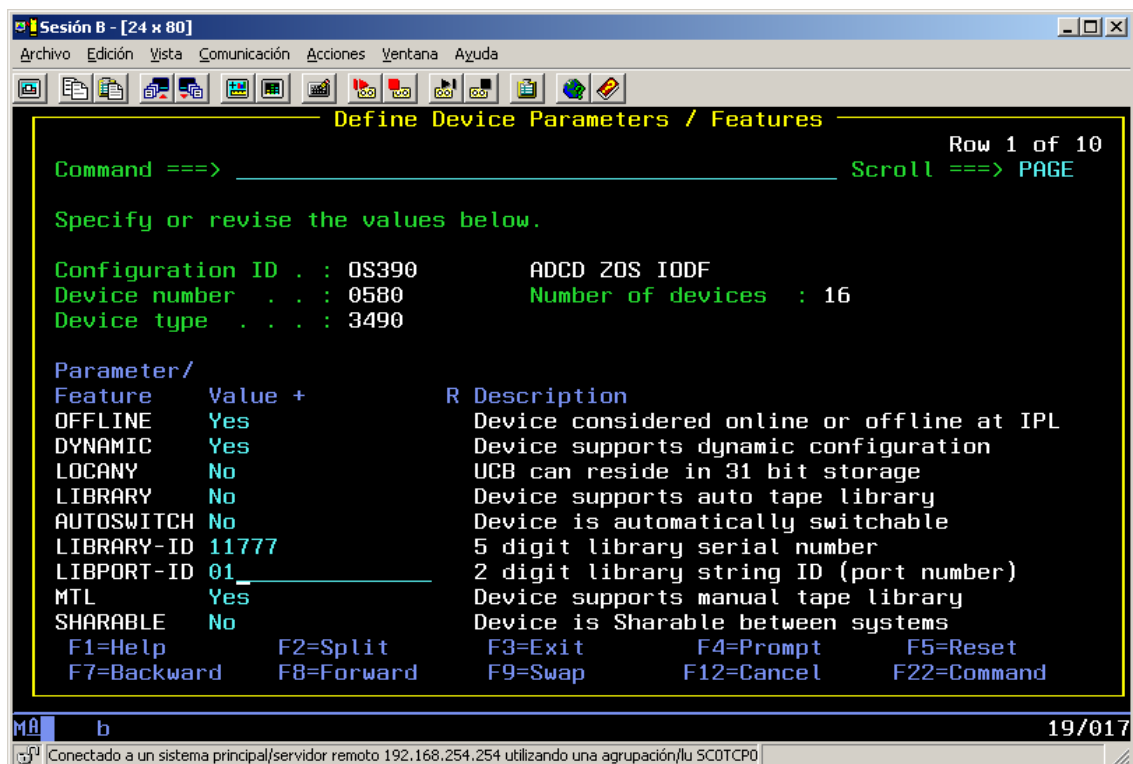


Fig. 28: Comportamiento de los Drives para con el Sistema Operativo.

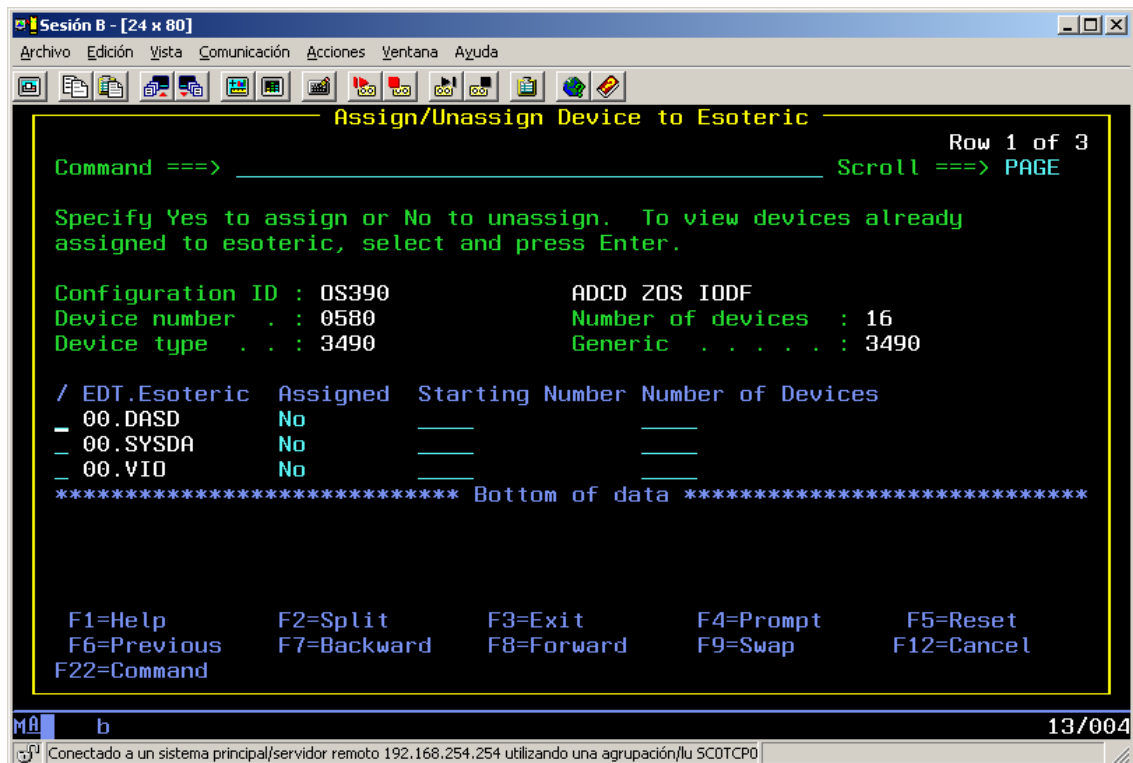


Fig. 29: Asignando dispositivos esotéricos.

7.- Con esto creado, salvamos la configuración y creamos un nuevo IODF, teniendo especial cuidado de modificar un nuevo miembro LOADXX de la SYS1.IPLPARM con el fin de

que coja la configuración en la próxima IPL (también podríamos activar la configuración dinámicamente, pero yo prefiero hacer una IPL para refrescar todo).

1.3.2 Configuración de OAM

OAM (Object Access Method) permite gestionar automáticamente los objetos donde se van a guardar nuestros datos, como cintas, discos, etc. Este sistema es particularmente útil cuando de lo que se trata es de administrar librerías automatizadas de cintas, como 3494 o similares, así como librerías virtuales.

El presente apartado explica como definir y activar el OAM con el fin de que podamos trabajar con librerías de cintas, ya que de no estar activado, el ISMF nos salta con un error.

1.3.2.1 Configuración de la PARMLIB

Para activar el OAM en nuestro z/OS ADCD, tenemos que editar una serie de miembros para que se active en tiempo de IPL..

1.3.2.1.1 Edición miembro IEFSSNXX

El miembro IEFSSNXX define los subsistemas que va a operar el z/OS, como el JES2, el SMS, el RACF, el RMM, etc. Para añadir el OAM como un subsistema, basta con editar el fichero y añadir al final una línea, tal que:

SUBSYS SUBNAME(OAM1) INITRTN(CBRINIT)

El fichero de ejemplo IEFSSNXX quedaría así:

```
SUBSYS SUBNAME(SMS)
  INITRTN(IGDSSIIN)
  INITPARM('ID=00,PROMPT=DISPLAY')
SUBSYS SUBNAME(JES2) /* JES2 AS PRIMARY SUBSYSTEM */
  PRIMARY(YES) START(NO)
SUBSYS SUBNAME(RACF) /* RACF SUBSYSTEM */
  INITRTN(IRRSSI00) INITPARM('Ñ')
SUBSYS SUBNAME(TNF)
SUBSYS SUBNAME(VMCF)
SUBSYS SUBNAME(DFRM) INITRTN(EDGSSSI)
SUBSYS SUBNAME(OAM1) INITRTN(CBRINIT)
```

Y con esto, ya tendríamos el sistema definido (en nuestro caso, se llama OAM1).

1.3.2.1.2 Añadir miembro CTICBR00 a la PARMLIB

En este miembro reside la configuración del OAM. Como no existe en la PARMLIB, basta con copiar de la **SYS1.SAMPLIB** el miembro **CBRCTI00** a nuestra PARMLIB y renombrarlo a **CTICBR00**. Con esto, nuestro OAM estará listo para arrancar.

1.3.2.2 Creación del catálogo VOLCAT.VGENERAL

Si hemos hecho IPL, y pretendemos ir al ISMF Opción 10 para definir una librería de cintas, cuando la definamos, nos va a dar un casque mayúsculo con un error "CATALOG

ACCESS ERROR". Esto es debido a que el OAM necesita un catálogo donde guardar todos los volúmenes y sus definiciones, de modo que al no existir, pues nos da casque.

Por tanto, vamos a definir un catálogo de usuario especial para que lleve el control de todos los objetos del OAM. Se llamará HLQ.VOLCAT.VGENERAL, siendo HLQ nuestro cualificador de alto nivel. Como este cualificador se debe poner en el LOADXX en la línea SYSCAT, para no andar mareando la perdiz, pondremos como HLQ SYS1 y punto. Por tanto, submitiremos el siguiente JCL para crearlo:

```
//DEFVOLCT JOB KUJAKU, 'DEFN VOLCAT.GENERAL', CLASS=A, MSGCLASS=H,
//          NOTIFY=&SYSUID, MSGLEVEL=(1,1)
//*****
//*   REALIZA UNA DEFINICION DEL CATALOGO SYS1.VOLCAT.VGENERAL
//*****
//PAS001 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
        DEFINE USERCATALOG -
          (NAME(SYS1.VOLCAT.VGENERAL) -
           VOLCATALOG -
           VOLUME(ZASYS1) -
           CYLINDERS(1 1))
/*
```

De esta forma, en el volumen ZASYS1, hemos reservado 1 cilindro para alojar el catalogo SYS1.VOLCAT.VGENERAL, que es un catalogo tipo VOLCATALOG.

1.3.2.3 Creación de la Started Task

Por último, crearemos la started task que arrancará el OAM y lo tendremos en memoria. Para ello, basta con copiar de la **SYS1.SAMPLIB** el miembro **CBRAPROC** a nuestra PROCLIB y renombrarlo a **OAM**. Con esto, nuestro OAM estará listo para arrancar, basta con hacer un **S OAM** desde la consola maestra.

Y con esto, ya tendríamos vía libre para definir roboticas, cintas, etc, mediante los paneles del ISMF.

1.3.3 DEFINICIÓN DFSMS DESDE CERO

Una vez tenemos el Hardware definido para el z/OS, lo siguiente es configurar el DFSMS para el tratamiento de los datos. En este caso concreto, voy a configurarlo desde cero, pero en las instalaciones existentes, solo habría que retocar las clases del SMS y ya estaría. Pero explicaré esto desde cero para hacerlo mas educativo.

1.- Creación del SCDS, ACDS y COMMDS. Los CDS son datasets VSAM que guardan toda la configuración y comportamiento del SMS. A este respecto, el SCDS es el Source Control Data Set, donde creamos y validamos las reglas del SMS, el ACDS es el Active Control

Data Set donde está la configuración activa y el COMMDS es el Communication Data Set, que gestiona las comunicaciones del SMS en un entorno SYSPLEX. Así que lo primero de todo es crear los Data Sets, con los siguientes JCLs.

Para el **ACDS**:

```
//DEFACDS JOB (),'INIT DFSMS ACDS',MSGLEVEL=(1,1),
//  CLASS=A,MSGCLASS=H,NOTIFY=&SYSUID
//PAS0001 EXEC PGM=IDCAMS
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
    DEFINE CLUSTER(NAME(SYS1.DFSMS.ACDS) LINEAR VOL(ZASYS1) -
    TRK(6 6) SHAREOPTIONS(3,3)) -
    DATA(NAME(SYS1.DFSMS.ACDS.DATA))
/*
```

Para el **SCDS**:

```
//DEFSCDS JOB (),'INIT DFSMS SCDS',MSGLEVEL=(1,1),
//  CLASS=A,MSGCLASS=H,NOTIFY=&SYSUID
//PAS0001 EXEC PGM=IDCAMS
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
    DEFINE CLUSTER(NAME(SYS1.DFSMS.SCDS) LINEAR VOL(ZASYS1) -
    TRK(6 6) SHAREOPTIONS(2,3)) -
    DATA(NAME(SYS1.DFSMS.SCDS.DATA))
/*
```

Y para el **COMMDS**:

```
//DEFCOMDS JOB (),'INIT DFSMS COMMDS',MSGLEVEL=(1,1),
//  CLASS=A,MSGCLASS=H,NOTIFY=&SYSUID
//PAS0001 EXEC PGM=IDCAMS
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
    DEFINE CLUSTER(NAME(SYS1.DFSMS.COMMDS) LINEAR VOL(ZASYS1) -
    TRK(1 1) SHAREOPTIONS(3,3)) -
    DATA(NAME(SYS1.DFSMS.COMMDS.DATA))
/*
```

2.- Lo siguiente es editar el miembro de la PARMLIB **IGDSMSXX** (en mi caso, es el IGDSMS00) y reflejar en él, los nuevos Data Sets creados:

```

SMS ACDS(SYS1.DFSMS.ACDS)
  COMMDS(SYS1.DFSMS.COMMDS)
  INTERVAL(15)
  DINTERVAL(150)
  REVERIFY(NO)
  ACSDEFAULTS(NO)
  TRACE(ON)
  SIZE(128K)
  TYPE(ALL)
  JOBNAME(*)
  ASID(*)
  SELECT(ALL)

```

NOTA: Si quisiéramos tener otro miembro distinto al **IGDSMS00**, deberíamos reflejarlo en el **IEASYSXX**, con una línea adicional llamada **SMS=XX**, siendo XX el número distinto al 00. En este caso, si no se quiere esperar a la IPL, se puede introducir en la master console un **T SMS=XX**, siendo XX el número elegido para la IGDSMSXX.

3.- Una vez editado este miembro, nos iríamos, también en la PARMLIB al miembro IEFSSNXX, para comprobar los parámetros del SMS:

```

SUBSYS SUBNAME(SMS)
  INITRTN(IGDSSIIN)
  INITPARM('ID=00,PROMPT=DISPLAY')
SUBSYS SUBNAME(JES2) /* JES2 AS PRIMARY SUBSYSTEM */
  PRIMARY(YES) START(NO)
SUBSYS SUBNAME(RACF) /* RACF SUBSYSTEM */
  INITRTN(IRRSSI00) INITPARM('Ñ')
SUBSYS SUBNAME(TNF)
SUBSYS SUBNAME(VMCF)
SUBSYS SUBNAME(DFRM) INITRTN(EDGSSSI)
SUBSYS SUBNAME(OAM1) INITRTN(CBRINIT)

```

1.3.4 PARAMETRIZACION DFSMS DESDE CERO

Con estos pasos, ya hemos definido la estructura SMS. Ahora, pasaremos a parametrizarla desde cero. Para ello, nos iremos a los menús ISMF desde nuestro TSO ISPF. En el, debemos tener vista de **Storage Administrator**, de lo contrario, no saldrán muchas opciones. Para obtenerla, Deberíamos elegir la opción **0 ISMF Profile** y una vez dentro, elegir **0 User Mode Selection** y seleccionar un **2 Storage Administrator**. Si salimos y volvemos a entrar, deberíamos ser capaces de ver un menú con muchas más opciones como el de la figura siguiente:

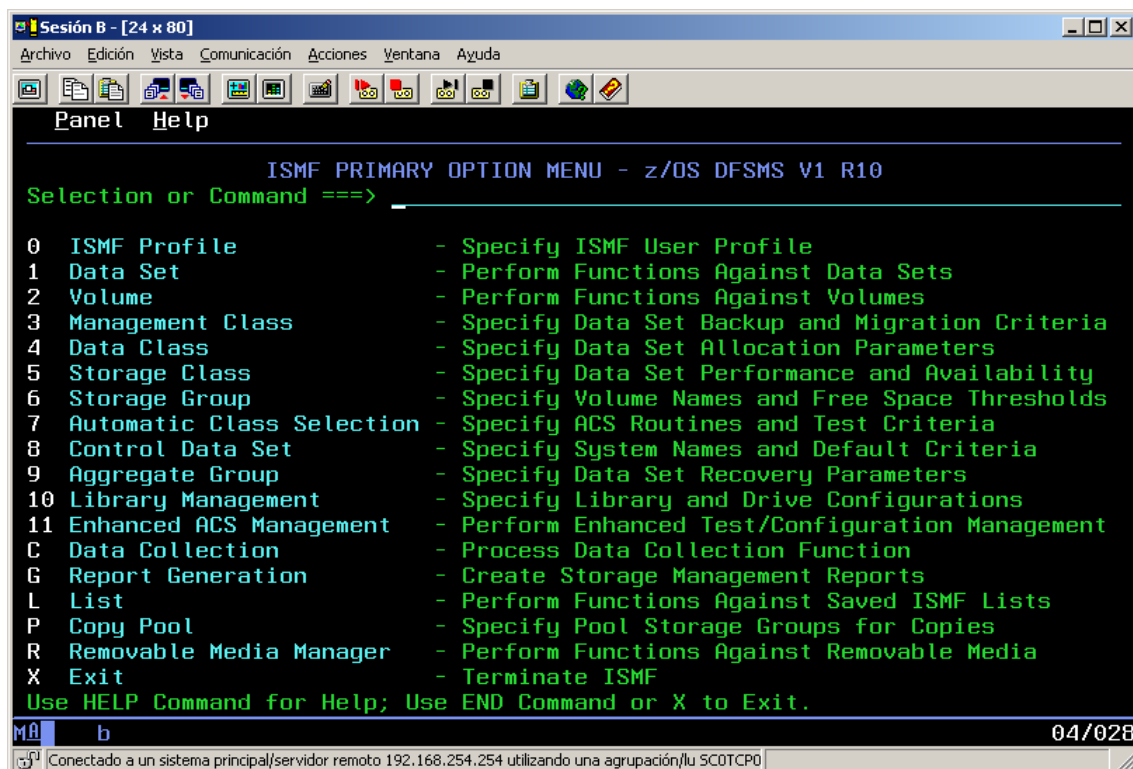


Fig. 30: Menú del administrador ISMF

Comenzaremos a parametrizar, siguiendo los pasos de a continuación:

1.- Definición de nuestra robótica Virtual. Para ello, nos vamos a la opción **10 Library Management** y elegiremos **3 Tape Library**.

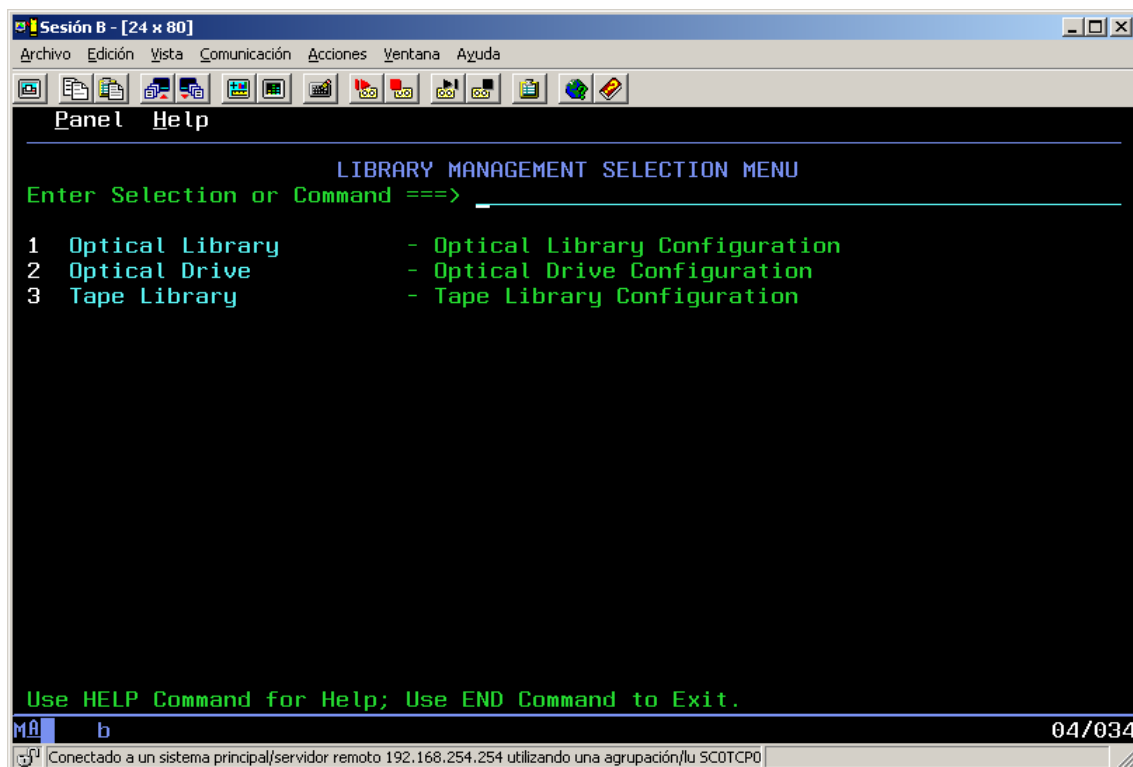
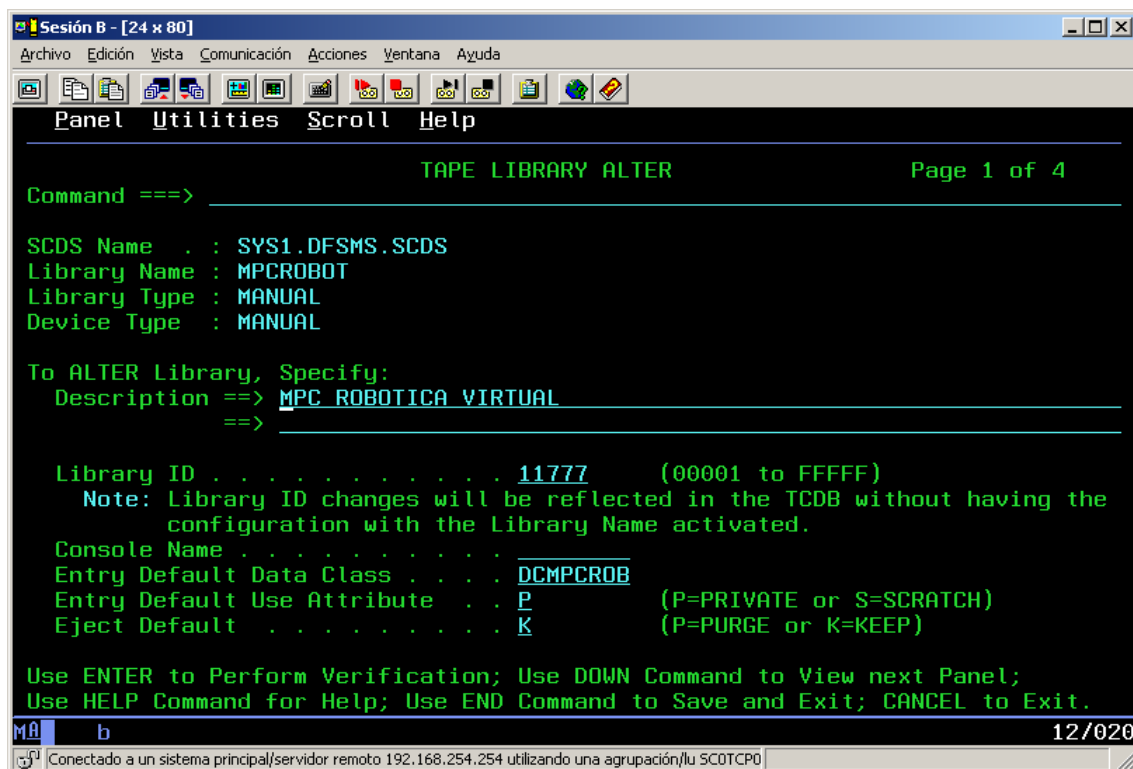
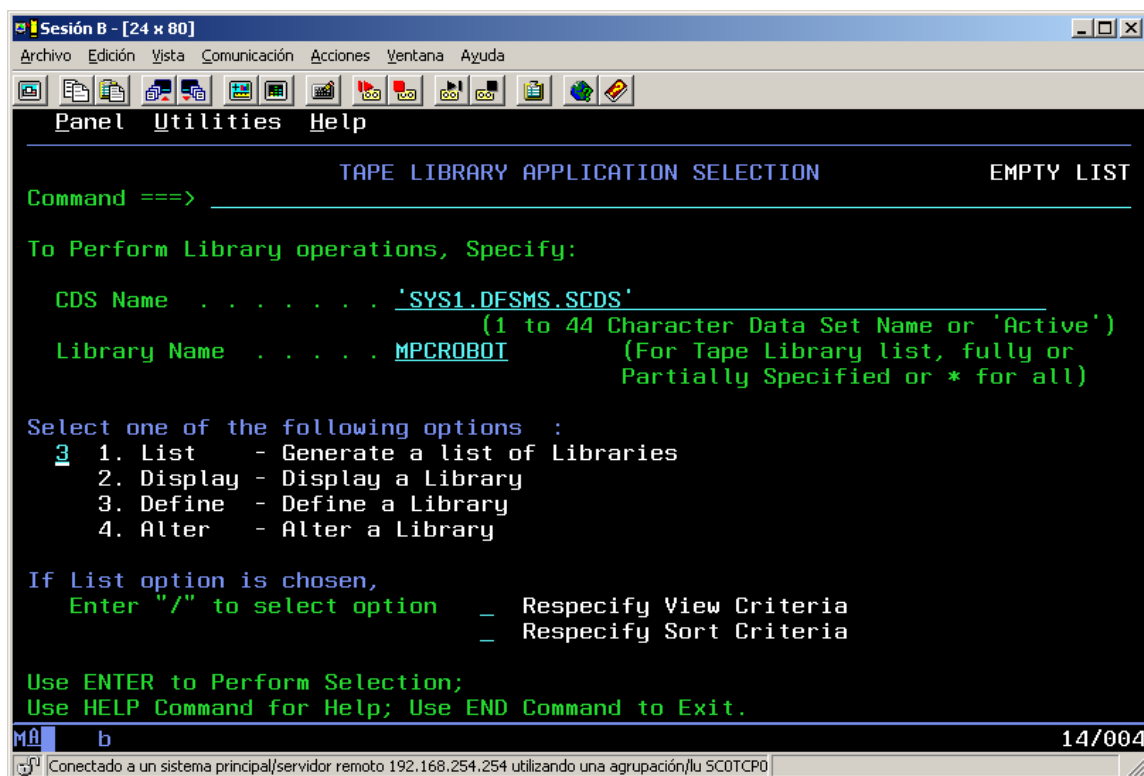


Fig. 31: Elección de la Tape Library

2.- Con un 3. Define, definiremos el nombre de nuestra robótica virtual –recordemos que era MPCROBOT- e introduciremos los datos de definición del HCD (ID 11777), así como el comportamiento de cintas (PRIVATE para las cintas asignadas y KEEP para las expulsadas).



También seleccionaremos la DataClass por defecto que vaya a utilizar esta robótica, que en los sucesivos pasos crearemos, y la denominaremos DCMPCROB (Data Class del MPC Robot).

3.- Si pulsamos **PF8**, nos lleva a la configuración de SCRATCH bajo la cual el sistema nos avisaría de que se queda sin cintas SCRATCH. Como MFNetDisk emula un 3490E, pondremos como **Media2** el umbral de SCRATCH y lo pondremos a 100 cintas.

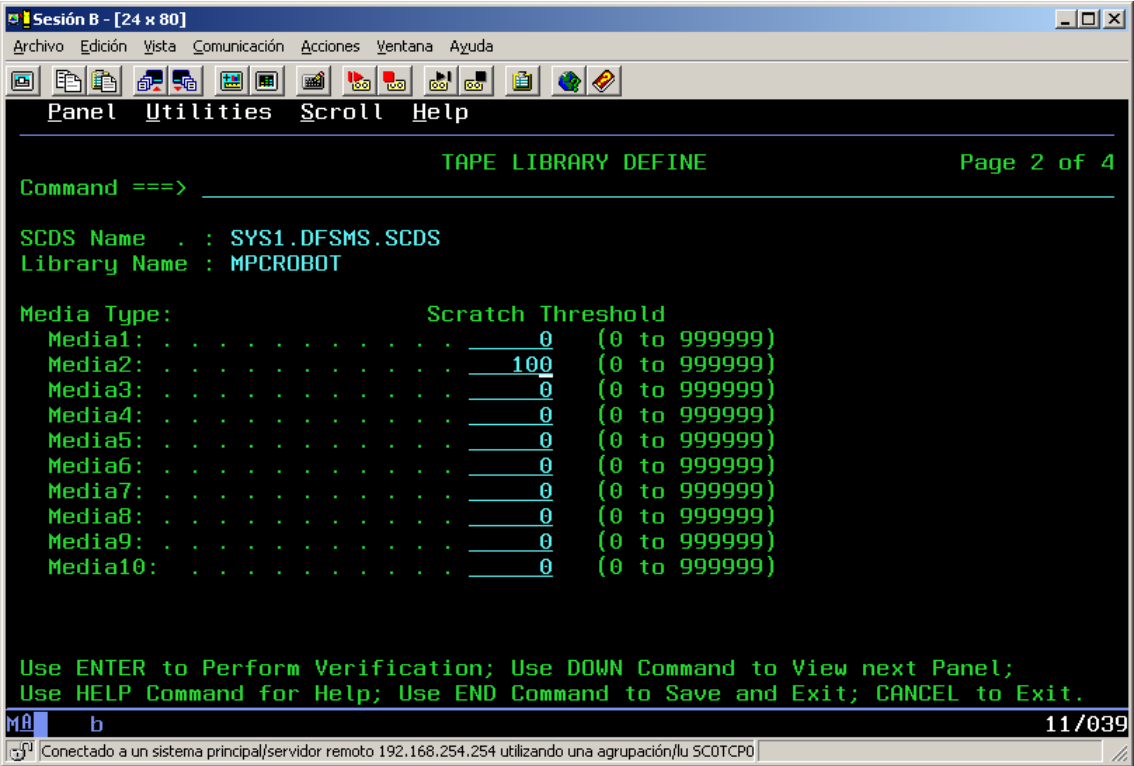


Fig. 34: Scratch Threshold

4.- Si pulsamos PF3, saldremos y se guardará la configuración. Lo siguiente es volver al menú inicial del ISMF y elegir la opción **4 Data Class**, donde elegiremos la Clase de Datos que llevará el comportamiento de los mismos.

Elegiremos los valores por defecto, salvo por los de la figura 38, ya que marcaremos el tipo de Medio a 2 y la tecnología de grabación a 36 pistas (es decir, una unidad 3490E estándar, que es precisamente la que emula el MFNetDisk).

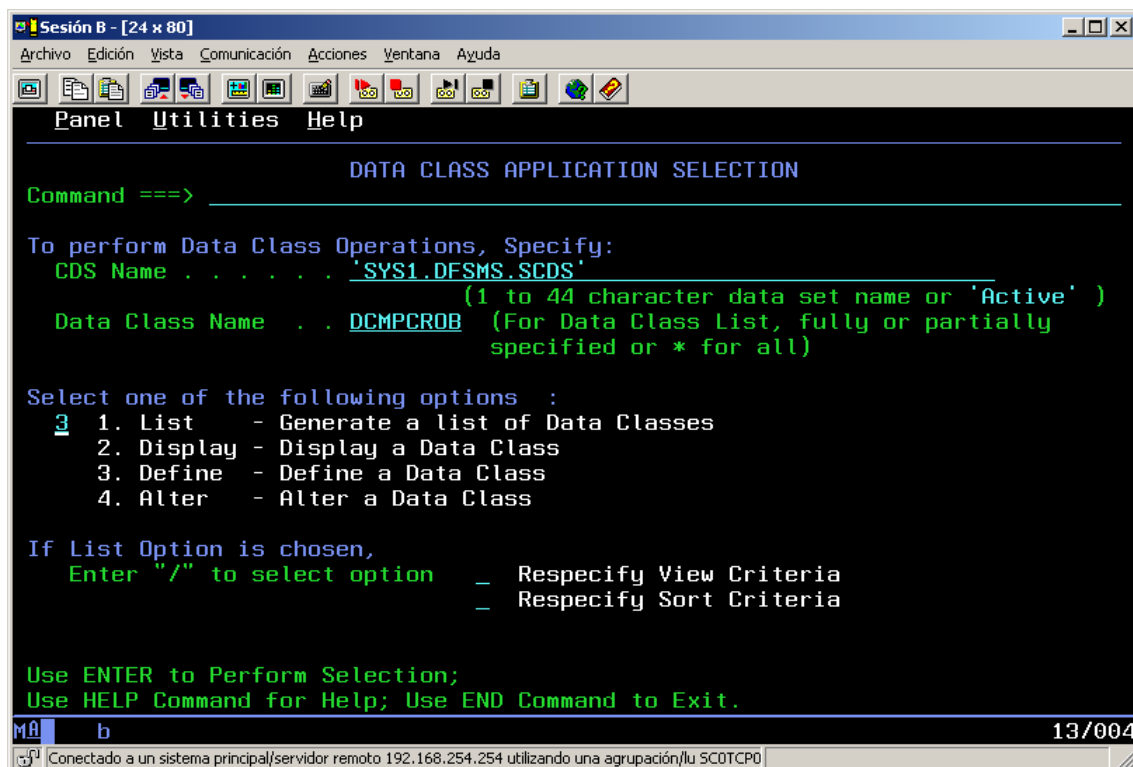


Fig. 35: Definición de la Dataclass DCMPCROB

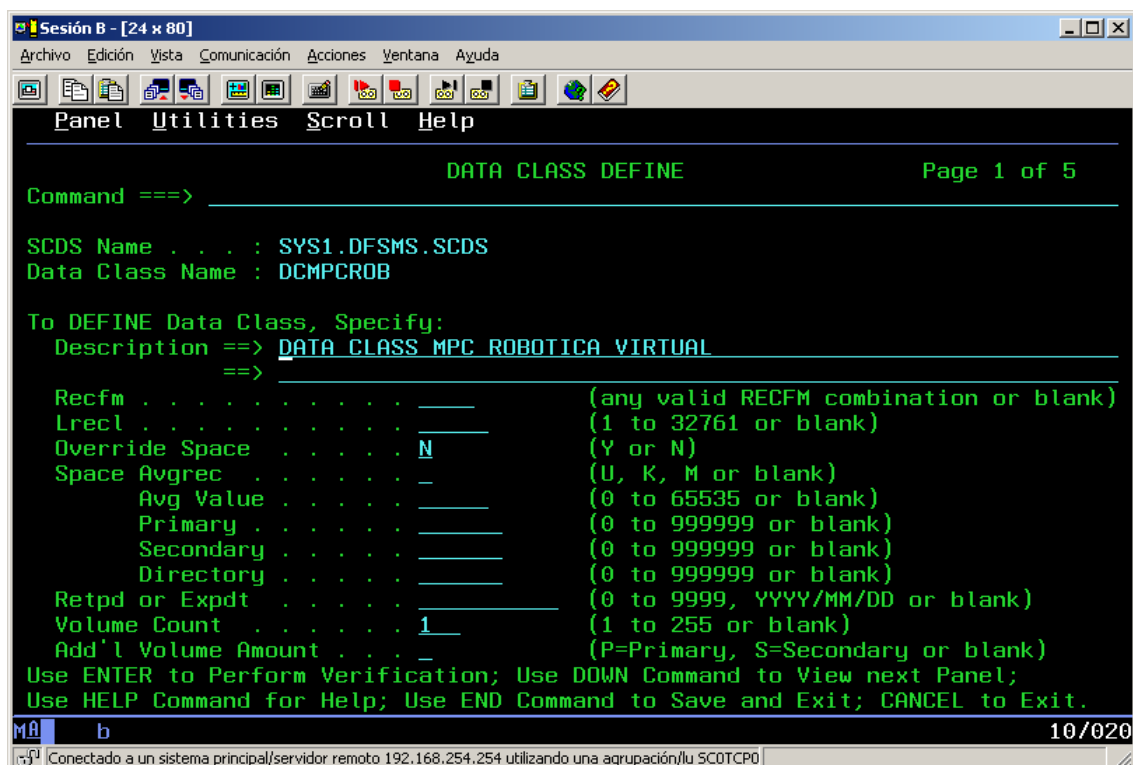


Fig. 36: Definición de la Dataclass

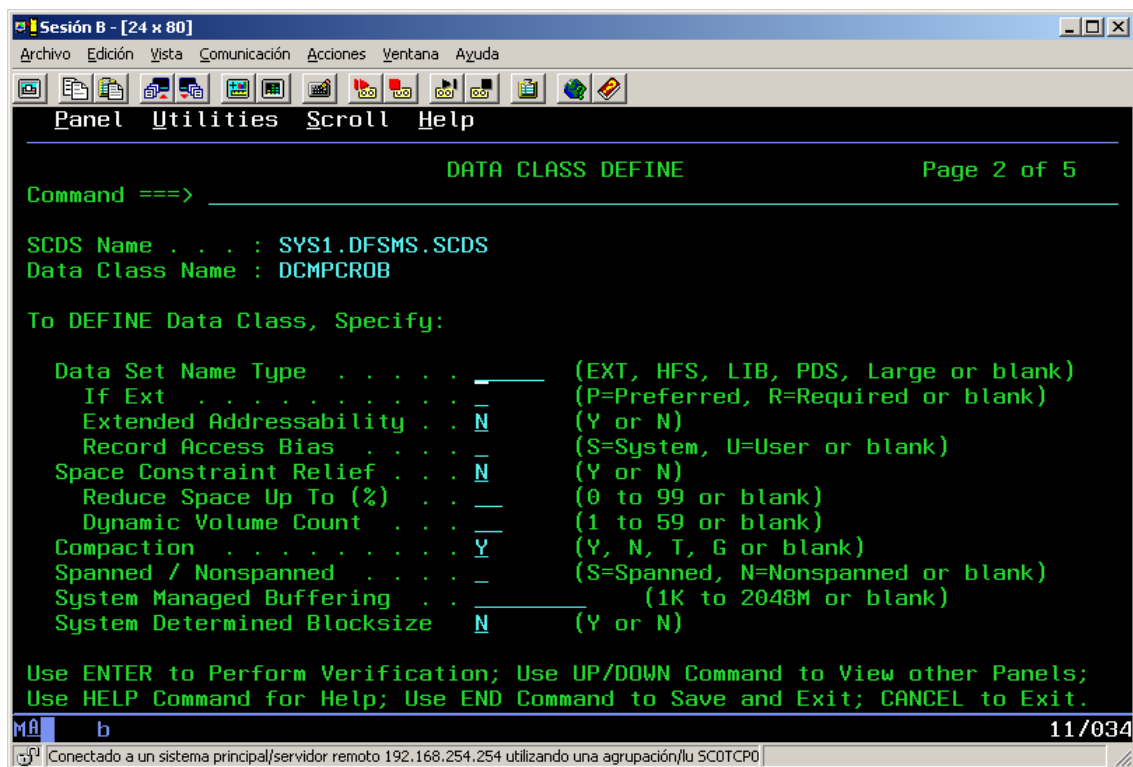


Fig. 37: Definición de la Dataclass

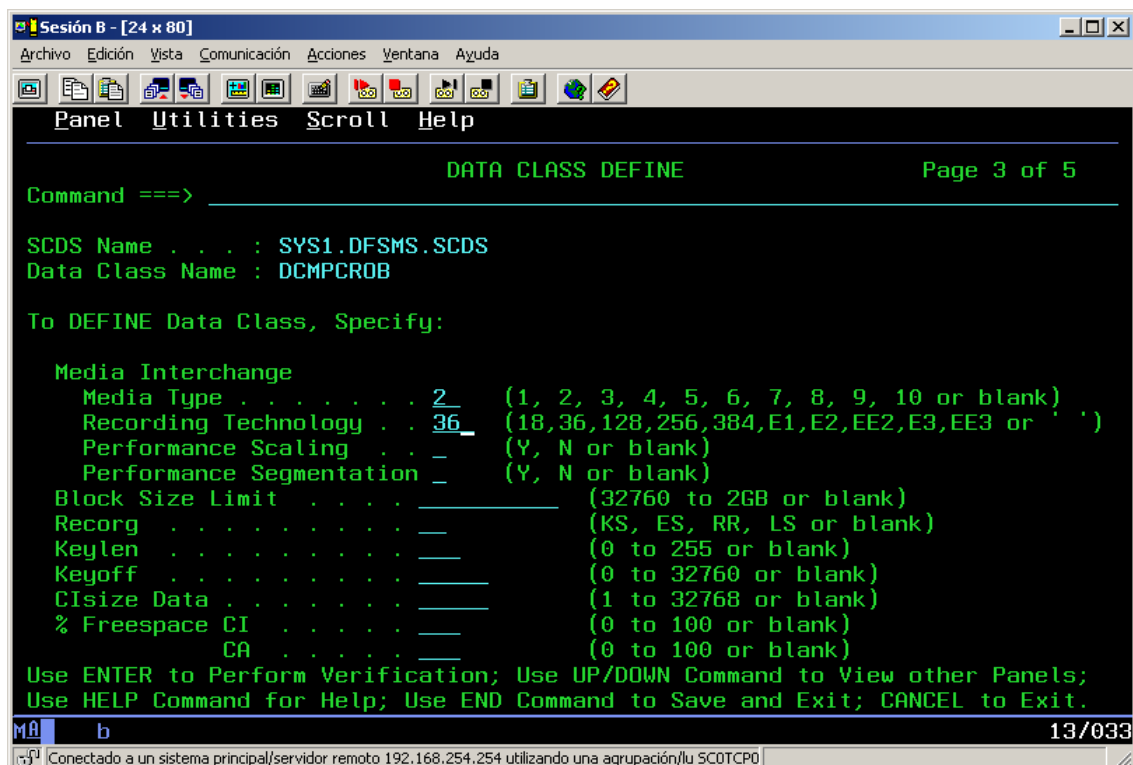


Fig. 38: Definición de Dataclass

5.- Lo siguiente es definir el Storage Group, desde el menú **6 Storage Group**. Evidentemente, al tratarse de una robótica, el tipo de Storage Group es **TAPE**, y asociaremos nuestra robótica MPCROBOT a este Storage Group, que llamaremos **SGMPCROB**.

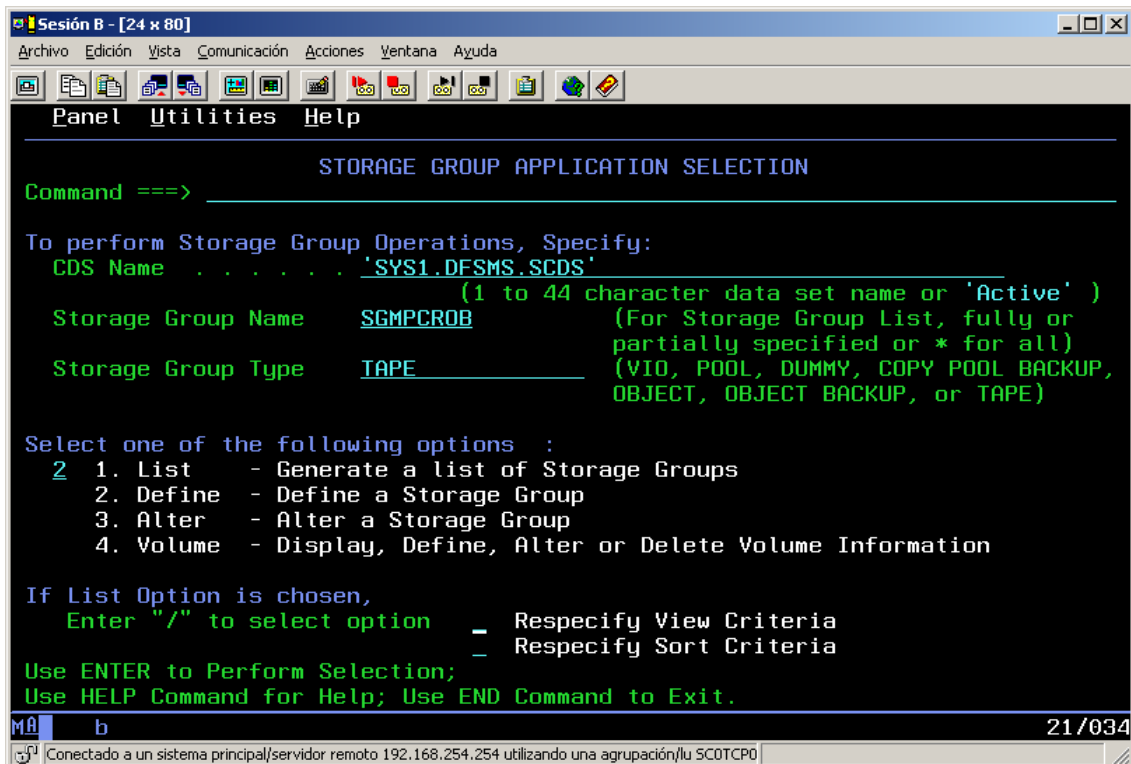


Fig. 39: Definición del Storage Group.

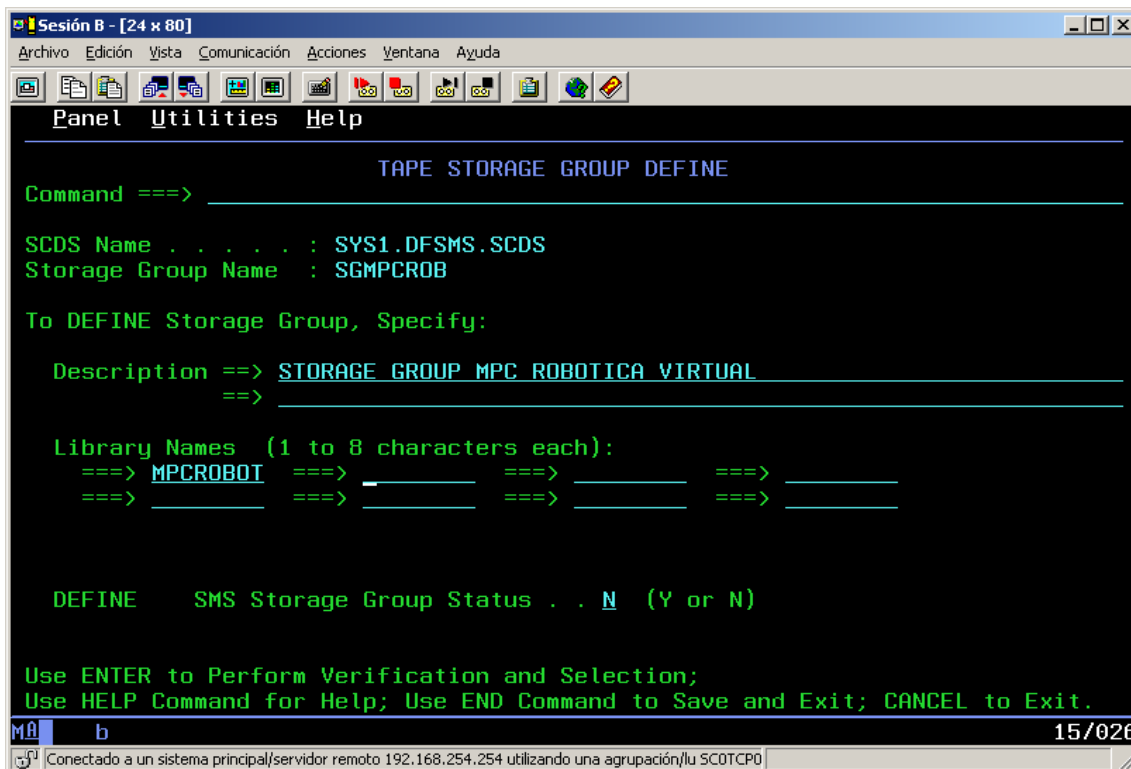


Fig. 40: Definición del Storage Group

También asignaremos nuestro Storage Group al nombre de nuestro sistema (en el ejemplo, **ADCD**) y lo habilitaremos poniendo un **ENABLE**.

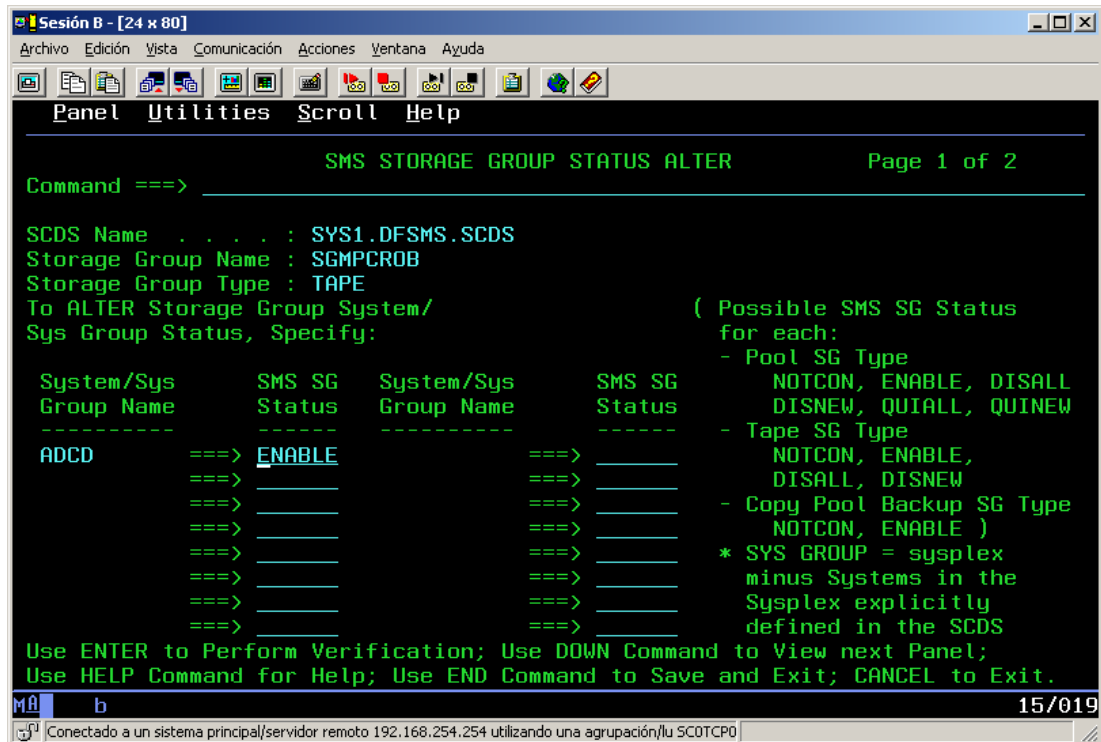


Fig. 41: Definición del Storage Group

6.- Ahora toca a la **Storage Class** (Opción 5 del ISMF) que llamaremos **SCMPCROB** pero dejaremos sus valores por defecto. Salvamos y salimos.

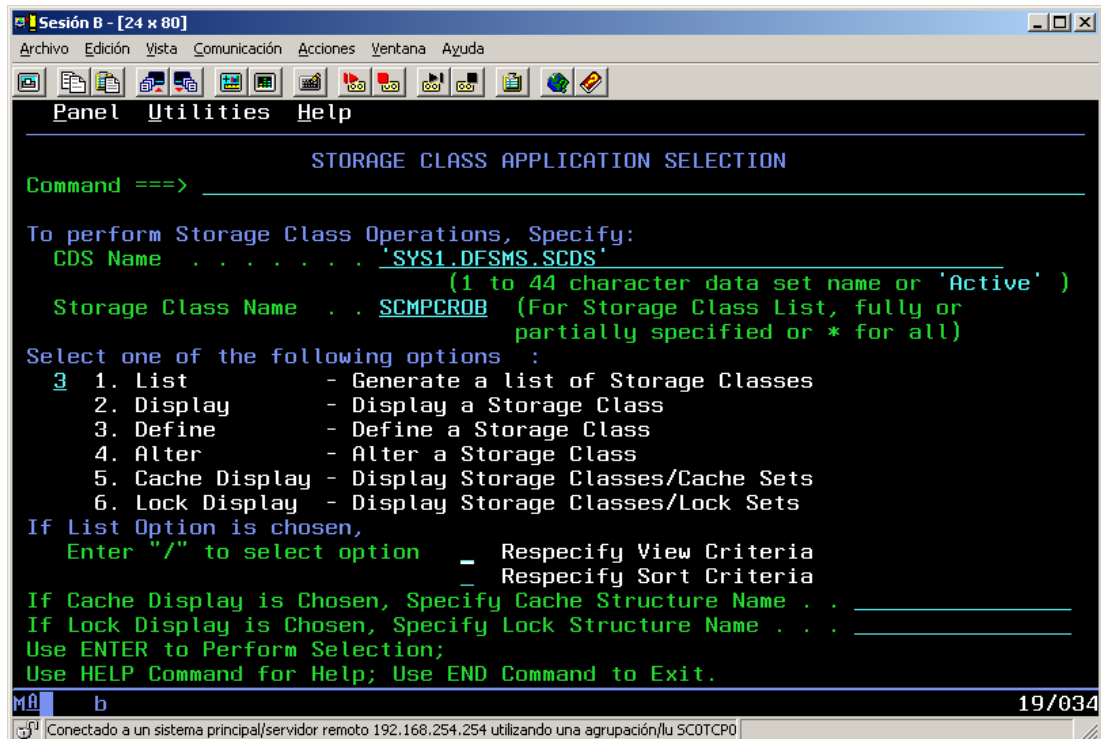


Fig. 42: Definición de la Storage Class

7.- Por último, crearemos la Management Class, vía opción 3 del ISMF, pero dejaremos los valores por defecto –expiración de datos sin límite, etc), y la denominaremos **MCMPCROB**.

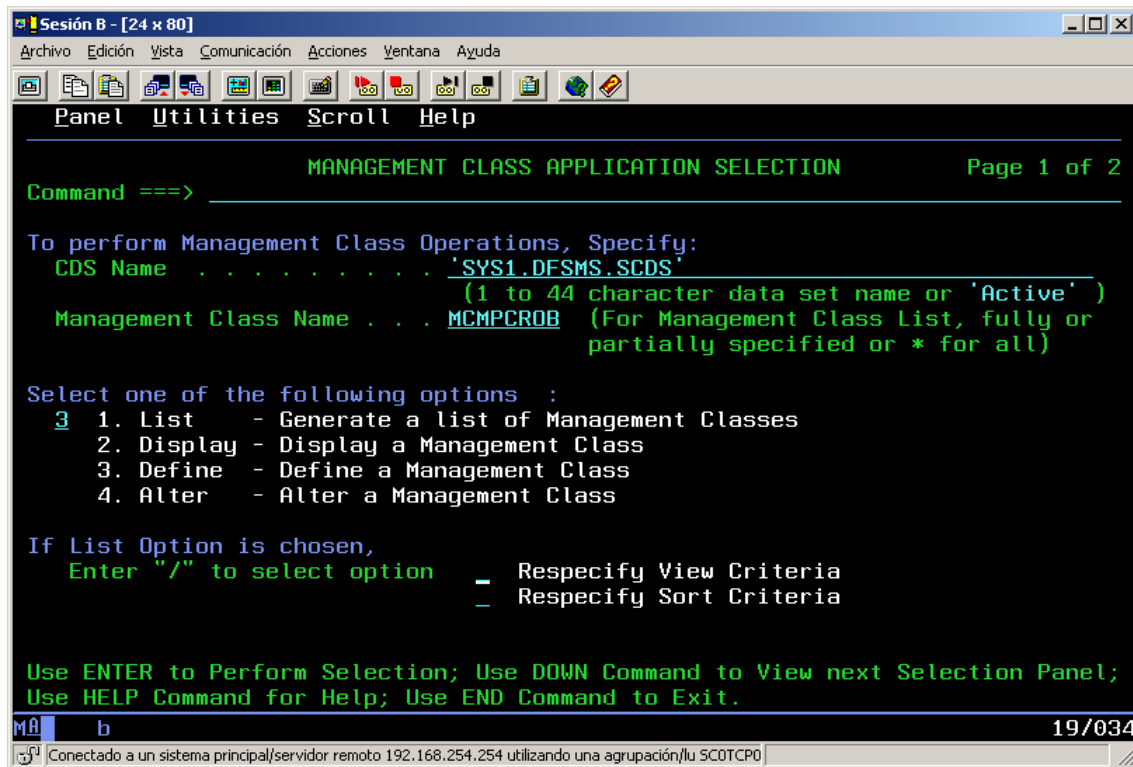


Fig. 43: Definición de la Management Class.

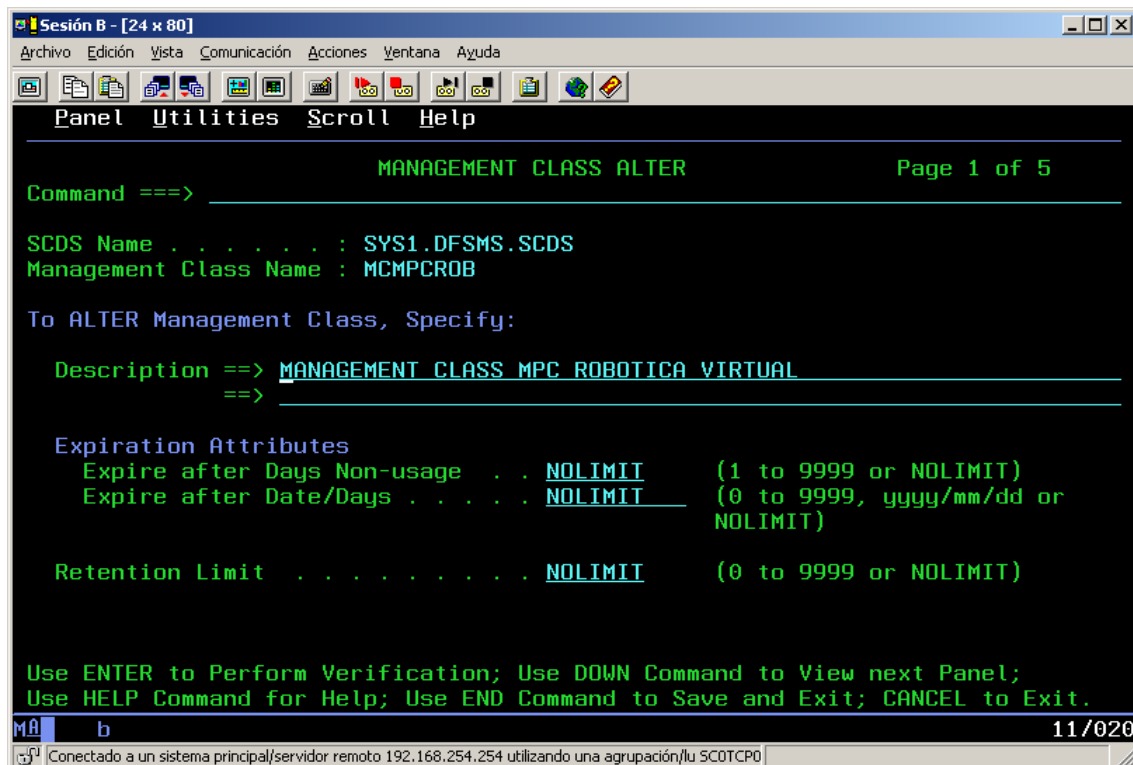


Fig. 44: Expiración de datos de la Management Class.

8.- Con las 4 clases definidas, lo siguiente es entrar en el menú **7 Automatic Class Selection**, donde definiremos las reglas de uso de las 4 clases definidas. Crearemos un dataset con 4 miembros, cada uno con una clase y los editaremos:

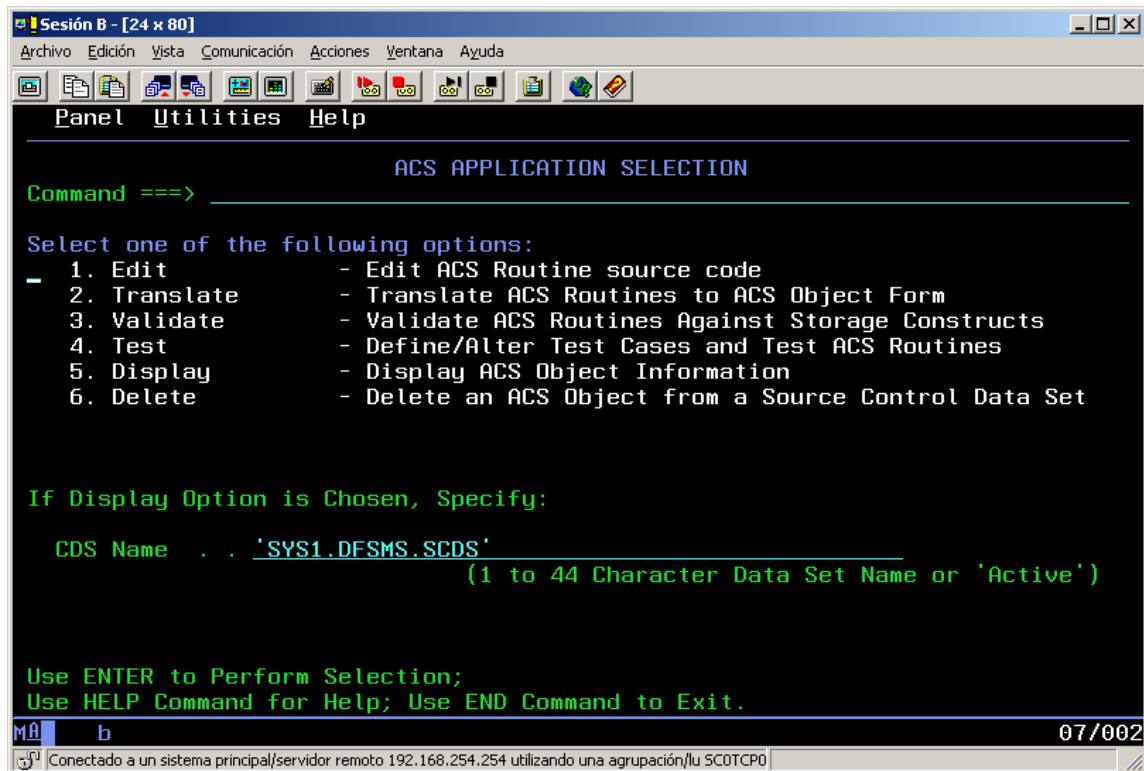


Fig. 43: Edición de las ACS

Tendrán los siguientes contenidos:

DATACLASS DCMPROB:

```
PROC DATACLAS
/*****
FILTLIST MTL_DSN INCLUDE(COPIA.MTL.** )
IF &DSN=&MTL_DSN THEN SET &DATACLAS='DCMPROB'
END
```

MANAGEMENT CLASS MCMPCROB:

```
PROC MGMTCLAS
/*****
FILTLIST MTL_DSN INCLUDE(COPIA.MTL.** )
IF &DSN=&MTL_DSN THEN SET &MGMTCLAS='MCMPCROB'
END
```

STORAGE CLASS SCMPROB:

```
PROC STORCLAS
/*****
FILTLIST MTL_DSN INCLUDE(COPIA.MTL.** )
IF &DSN=&MTL_DSN THEN SET &STORCLAS='SCMPROB'
END
```

STORAGE GROUP SGMPCROB:

```
PROC STORGRP
/*****
FILTLIST MTL_DSN INCLUDE(COPIA.MTL.** )
IF &STORCLAS='SCMPCROB' THEN SET &STORGRP='SGMPCROB'
  WRITE 'STORGRP = ' &STORGRP
  WRITE 'STORCLAS= ' &STORCLAS
END
```

El Storage Group y el Storage Class definen el comportamiento de la grabación de los datos. En este caso, todo dataset que empiece por **COPIA.MTL.**** adquirirá este Storage Group, lo que hará que automáticamente se grabe en los volúmenes de cinta de este robot.

9.- Como último paso, y una vez editados los miembros correspondientes, realizaremos un **2. Traslate** uno por uno y después un **3. Validate**, para verificar que no existen errores. El listado de salida de la verificación final, debería ser algo similar a esto:

VALIDATION RESULTS

```
VALIDATION RESULT:  VALIDATION SUCCESSFUL - WARNINGS DETECTED
SCDS NAME:          SYS1.DFSMS.SCDS
ACS ROUTINE TYPE:    *
DATE OF VALIDATION:  2011/08/10
TIME OF VALIDATION:  14:17
```

IGD06026I NO POOL-TYPE STORAGE GROUPS EXIST IN THE CONFIGURATION

En este caso concreto, nos da un Warning haciendo referencia a que no hemos definido ningún POOL de discos en el Storage Group, pero lo obviamos porque de lo que se trata es de definir robóticas virtuales.

Una vez obtenido este resultado, nos iríamos la opción **8. Control Data Set** del ISMF y haríamos un **5. Activate**, con una confirmación para realizar la activación:

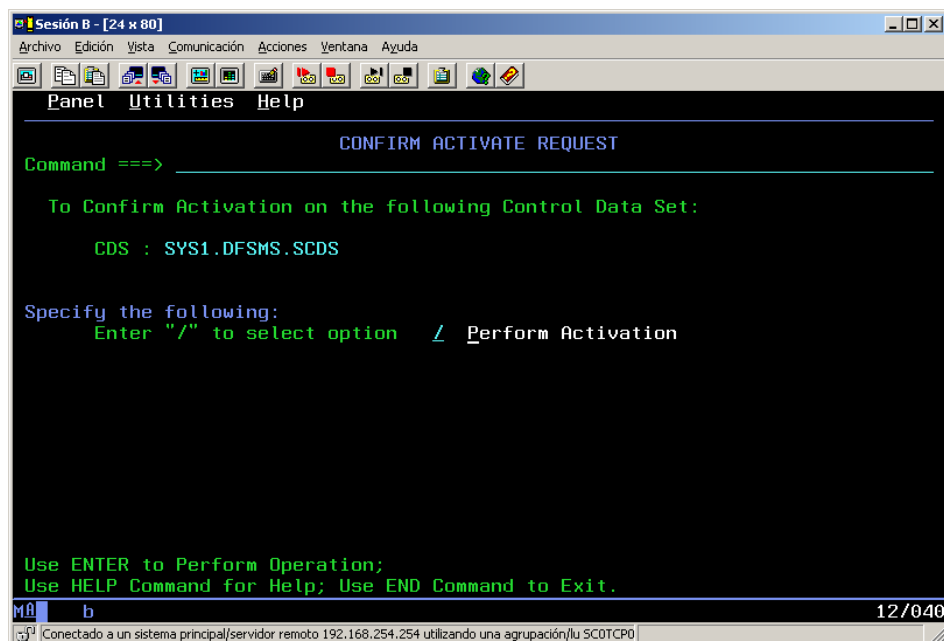


Fig. 44: Confirmar Activación.

A partir de este momento, tenemos el SMS preparado para operación con el robot.

1.3.5 Activación de DFSMSrmm

Esta apartado explica como activar el RMM (Removable Media Manager), que es un módulo que se encarga de la gestión de todas las bibliotecas de cintas, tanto automáticas (robótica) o manuales o standalone. Contamos con el hecho de que el DFSMSrmm está instalado en nuestro sistema ADCD. Para activar el RMM, ejecutaremos los siguientes pasos:

1.3.5.1 Customización de miembros de la PARMLIB

1.- Lo primero de todo, hay que verificar si en el miembro **IFAPRDXX**, se encuentra el miembro instalado. Si lo está, deberían aparecer las siguientes líneas. Si no lo está, las añadiremos:

```
PRODUCT OWNER('IBM CORP')
      NAME(Z/OS)
      ID(5694-A01)
      VERSION(*) RELEASE(*) MOD(*)
      FEATURENAME(DFSMSRMM)
      STATE(ENABLED)
```

2.- Lo siguiente es comprobar el miembro **IEFSSNXX**, y añadir la línea EDGSSI, para que el z/OS sepa que subsistema lanzar. Por tanto, se debería añadir una línea como esta:

```
SUBSYS SUBNAME(DFRM) INITRTN(EDGSSSI)
```

3.- Debemos modificar las autorizaciones de TSO para poder emitir comandos de RMM usando la opción 6 del ISPF, por lo que debemos añadir a la **IKJTSOXX** los comandos necesarios. Para averiguar que IKJTSO tenemos, con hacer un **/D IKJTSO,SEND** debería aparecernos la 00. De hecho, si no se le especifica nada en la IEASYS con la variable IKJTSO=XX, coge la 00 por defecto. Una vez sabido el miembro correcto (en nuestra instalación, IKJTSO00), añadiremos los siguientes comandos:

```
AUTHCMD NAMES(          /* AUTHORIZED COMMANDS      */ +
      RMM                /* RMM                  */ +
AUTHPGM NAMES(          /* AUTHORIZED PROGRAMS  */ +
      EDGHSKP            /* RMM                  */ +
      EDGUTIL            /* RMM                  */ +
      EDGRPTD            /* RMM                  */ +
      EDGAUD             /* RMM                  */ +
AUTHTSF NAMES(          /* PROGRAMS TO BE AUTHORIZED */ +
      /* WHEN CALLED THROUGH THE */ +
      /* TSO SERVICE FACILITY.   */ +
      EDGHSKP            /* RMM                  */ +
      EDGUTIL            /* RMM                  */ +
      EDGRPTD            /* RMM                  */ +
      EDGAUD             /* RMM                  */ +
```

Con esto, la lista de comandos de administración de RMM quedará autorizada en TSO.

4.- Si quisiéramos que el RMF obtuviera datos del RMM para estadísticas, deberíamos modificar el miembro **SMFPRMXX**, pero es un paso opcional que no seguiremos por no ser crítico.

5.- Por último, debemos crear un miembro llamado **EDGRMMXX** (en nuestro caso, EDGRMM00), copiado de la SYS1.SAMPLIB(EDGIVPPM), que es donde se definirán los parámetros de arranque del RMM cuando su STC se inicie. Dicho contenido será similar al que se muestra a continuación:

```

/*-----*/
/******/
/* DFSMS/MVS 1.1.0 DFSMSrmm */
/* */
/*PROPRIETARY V3 STATEMENT */
/*LICENSED MATERIALS - PROPERTY OF IBM */
/*"RESTRICTED MATERIALS OF IBM" */
/*5695-DF1 */
/*(C) COPYRIGHT 1979,1993 IBM CORP. */
/*END PROPRIETARY V3 STATEMENT */
/******/
/* */
/* RMM (JDZ1150) IVP. IVP EDGRMM00 PARMLIB MEMBER */
/* */
/* COPY THIS MEMBER TO 'SYS1.PARMLIB' AS MEMBER EDGRMM00. */
/* */
/* CHANGE THE MASTER AND JOURNAL FILE NAMES IF NECESSARY */
/* */
/*-----*/
OPTION DSNAME(RMM.CONTROL.DSET) /* MASTER FILE NAME*/ -
        JRNLNAME(RMM.JOURNAL.DSET) /* JOURNAL NAME */ -
        CDSID(RMM1) -
        OPMODE(R)
MNTMSG MSGID(IEF233A) ID(1) VOLUME(16) RACK(999)
MNTMSG MSGID(IEF233D) ID(1) VOLUME(16) RACK(999)
MNTMSG MSGID('IEF234E K') ID(1) VOLUME(16) RACK(999)
MNTMSG MSGID('IEF234E R') ID(1) VOLUME(16) RACK(999)
MNTMSG MSGID('IEF234E D') ID(1) VOLUME(16) RACK(999)
MNTMSG MSGID('IEF455D') ID(1) VOLUME(16) RACK(999)
MNTMSG MSGID(IEC501A) ID(1) VOLUME(16) RACK(999)
MNTMSG MSGID('IEC502E K') ID(1) VOLUME(16) RACK(999)
MNTMSG MSGID('IEC502E D') ID(1) VOLUME(16) RACK(999)
MNTMSG MSGID('IEC502E R') ID(1) VOLUME(16) RACK(999)
MNTMSG MSGID('IEC502E RD') ID(1) VOLUME(17) RACK(999)
MNTMSG MSGID('IEC502E RK') ID(1) VOLUME(17) RACK(999)
MNTMSG MSGID(IAT5110) ID(1) VOLUME(44) RACK(999)
MNTMSG MSGID(IAT5210) ID(1) VOLUME(50) RACK(999)
MNTMSG MSGID(IAT5410) ID(1) VOLUME(20) RACK(999)

```

Hay que tener en cuenta los valores en Negrita. Por un lado, tenemos el **DSNAME**, que especifica el dataset VSAM que llevará en control del número de cintas que maneja el RMM. Por otra parte, la opción **JRNLNAME** especifica donde grabaremos el JOURNAL del RMM, y por último, tenemos el **OPMODE** u "Operation Mode" que indica el modo de trabajo del RMM (**M** para manual, **P** para protegido, **R** para grabación (modo normal de funcionamiento) y **W** para warning). El **CDSID** es un concepto nuevo, introducido desde la versión 1.9 del z/OS, que nos indica el nombre de nuestro RMM ya que en un SYSPLEX podemos tener varios funcionando, sobre todo si existen distintas robóticas. Lo llamaremos **RMM1**. Esto lo tenemos que recordar, ya que cuando se creen los Dataset de Control, debemos hacer uso de este parámetro en su creación y los nombres deben figurar como idénticos.

1.3.5.2 Creación del la STC de la PROCLIB

Necesitamos un miembro en la PROCLIB que será en que se lance como STC para que el RMM esté bajo la supervisión del JES2. Para ello, copiaremos de **SYS1.SAMPLIB** el miembro **EDGDFRMM** y lo pegaremos en nuestra PROCLIB pero renombrándolo como DFRMM. El contenido de dicho miembro lo listamos a continuación:

```
//DFRMM    PROC M=00,OPT=MAIN
//*
//*    PROCEDURE FOR DFRMM SUBSYSTEM TASK
//*
//*    1. M Is used to pick up the necessary PARMLIB member name suffix.
//*    OPT can be used to execute the EDGRESET utility to inactivate
//*    the DFRMM subsystem interface before stopping DFRMM procedure.
//*    S DFRMM,OPT=RESET
//*    P DFRMM
//*    2. IEFRDER and PARMLIB are optional DD statements - see
//*    the Installation & Customization Guide for details.
//*    IEFRDR is only honoured for single step procedures
//*    3. This procedure is designed to run as a single step procedure
//*    4. If desired, the MASTER and JOURNAL DD statements can be
//*    added to the procedure. Refer to the I&C Guide for details.
//*    5. The DFRMM startup procedure can optionally include an
//*    OUTPUT JCL statement to direct sticky label output. The
//*    statement name must match the DISPDDNAME specified in parmlib.
//*    If the OUTPUT statement is not specified, RMM issues the
//*    label as write to operator messages using route code 13.
//*
//*****
//* z/OS    DFSMSrmm V1R3                                     *
//*                                                *
//*PROPRIETARY V3 STATEMENT                                     *
//*LICENSED MATERIALS - PROPERTY OF IBM                       *
//*"RESTRICTED MATERIALS OF IBM"                             *
//*5694-A01                                                    *
//*(C) COPYRIGHT 1993,2001  IBM CORP.                         *
//*STATUS = HDZ11G0                                           *
//*END PROPRIETARY V3 STATEMENT                               *
//*****
//* Change history:                                           *
//* $01=0W57435,210,021205,MW: Increase REGION value         @01A *
//*****
//*
//*IEFPROC EXEC PGM=EDG&OPT,REGION=0M,PARM='&M',TIME=1440    @01C
//*PARMLIB DD DDNAME=IEFRDER
//*IEFRDER DD DISP=SHR,DSN=SYS1.PARMLIB
//*MMDISP OUTPUT DEST=SYSTEMX,FORMS=LABEL,CLASS=L
//EDGPDOX DD DISP=SHR,DSN=RMM.RMMPDOX
//EDGPDOY DD DISP=SHR,DSN=RMM.RMMPDOY
```

Con **M=00**, le estamos diciendo que esta STC se alimente del miembro EDGRMM00 de la PARMLIB, editado anteriormente. También hay que alocar dos datasets que llevarán las trazas del RMM, llamados **RMMPDOX** y **RMMPDOY**. Como hay que crear varios datasets, el siguiente apartado muestra cómo crear todos los datasets que entran en juego.

1.3.5.3 Creación de Datasets, Catálogos de usuario, etc.

Este apartado explicará paso a paso los datasets que se deben crear para el correcto funcionamiento del RMM. Como intervienen varios datasets, crearemos un catálogo solo para ellos y su alias correspondiente. Para ello, seguiremos los siguientes pasos:

1.- Definiremos un catálogo llamado CATALOG.RMM, y lo conectaremos al catálogo maestro, CATALOG.Z110.MASTER, con el siguiente JCL:

```
//DEFCAT00 JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//DEFMCAT EXEC PGM=IDCAMS,REGION=8M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DEFINE -
    UCAT ( -
      ICFCATALOG -
      NAME(CATALOG.RMM) -
      VOL(ZASYS1) -
      CYL(10 5) -
      FREESPACE(10 10) -
      SHR(3 4) -
    IMBED -
      REPLICATE -
      STRNO(3) -
    ) -
  DATA ( -
    CISZ(4096) -
    BUFND(4) -
  ) -
  INDEX( -
    BUFNI(4) -
  ) -
  CAT(CATALOG.Z110.MASTER)
/*-----
/* CONNECT THE DRIVING SYSTEM MASTER CATALOG INTO THE NEW TARGET
/* SYSTEM MASTER CATALOG FOR THE CATCVTM JOB
/*-----
//CONNECT1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  IMPORT CONNECT -
  OBJECTS((CATALOG.RMM -
  DEVICETYPE(3390) -
  VOLUME(ZASYS1))) -
  CATALOG(CATALOG.Z110.MASTER)
/*
```

2.- Definiremos el ALIAS para que todo lo que empiece por RMM, se vaya al catálogo **CATALOG.RMM**, con el siguiente JCL:

```
//DEFALIAS JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//PAS0001 EXEC PGM=IDCAMS,REGION=1024K
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DEFINE ALIAS (NAME(RMM) -
    RELATE(CATALOG.RMM)) -
    CATALOG(CATALOG.Z110.MASTER)
/*
```

3.- Crearemos el dataset de Control del RMM (RMM.CONTROL.DSET) con el siguiente JCL:

```
//CREAVSAM JOB CLASS=A,NOTIFY=&SYSUID,MSGCLASS=X,
// MSGLEVEL=(1,1)
//PAS01 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
```

```
//DASD      DD DISP=SHR,UNIT=SYSDA,VOL=SER=ZASYS1
//SYSIN      DD *
  DEFINE CLUSTER(NAME(RMM.CONTROL.DSET) -
    FILE(DASD) -
    FREESPACE(15 0) -
    KEYS(56 0) -
    REUSE -
    RECSZ(512 9216) -
    SHR(3 3) -
    KILOBYTES(4500 1500) -
    VOLUMES(ZASYS1)) -
    DATA(NAME(RMM.CONTROL.DSET.DATA) -
    CISZ(26624)) -
    INDEX(NAME(RMM.CONTROL.DSET.INDEX) -
    CISZ(2048))
/*
```

Y lo inicializaremos, con el siguiente JCL (nótese el nuevo parámetro **CDSID**, que debe tener el MISMO NOMBRE que nuestro miembro de configuración de la PARMLIB):

```
//CDSINIT JOB CLASS=A,NOTIFY=&SYSUID,MSGCLASS=X,
//          MSGLEVEL=(1,1)
//EDGUTIL EXEC PGM=EDGUTIL,PARM=CREATE
//SYSPRINT DD SYSOUT=*
//MASTER DD DSN=RMM.CONTROL.DSET,DISP=SHR
//SYSIN DD *
  CONTROL CDSID(RMM1)
/*
```

4.- Crearemos el JOURNAL del RMM, utilizando como base el JCL del la SYS1.SAMPLIB, EDGJNLAL, y le haremos las modificaciones oportunas:

```
//EDGJNLAL JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*****
//* DFSMS/MVS 1.4.0 DFSMSrmm *
//* * * * *
//*Proprietary V3 Statement *
//*Licensed Materials - Property Of IBM *
//*"Restricted Materials of IBM" *
//*5695-DF1 (C) COPYRIGHT 1979,1993 IBM CORP. *
//*End Proprietary V3 Statement *
//*****
//* * * * *
//* RMM (HDZ11D0). ALLOCATE JOURNAL FILE *
//* * * * *
//* THIS JOB IS USED TO ALLOCATE A JOURNAL FILE FOR THE RMM *
//* CONTROL DATA SET. *
//* * * * *
//* MAKE THE FOLLOWING CHANGES TO THIS JOB BEFORE SUBMITTING *
//* IT FOR EXECUTION: *
//* 1.JOBCARD. Enter required job card information for your *
//* installation. *
//* 2.volser Set to the serial number of the volume on which *
//* to allocate the JOURNAL file. *
//* 3.JOURNAL.FILE.DSNAME Set to the name selected. *
//* 4.store_class to a storage class with GUARANTEED SPACE *
//* or delete the STORCLAS keyword. *
//* 5.SPACE. Update the values for space allocation if you *
//* wish to use a different sized Journal data set *
//* * * * *
```

```

// *-----*
//JOURNAL EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//JOURNAL DD DISP=(NEW,CATLG),DSN=RMM.JOURNAL.DSET,
//          UNIT=SYSDA,VOL=SER=ZASYS1,
//          AVGREC=U,SPACE=(4096,(1000))

```

5.- Crearemos los datasets de trazas o PDAs, con el siguiente JCL:

```

//ALLOPDO JOB CLASS=A,MSGLEVEL=1,NOTIFY=&SYSUID
//STEP1 EXEC PGM=IEFBR14
//DD1 DD DSN=RMM.RMMPDOX,DISP=(,CATLG),
//        UNIT=SYSDA,
//        VOL=SER=ZASYS1,SPACE=(CYL,(20))
//DD2 DD DSN=RMM.RMMPDOY,DISP=(,CATLG),
//        UNIT=SYSDA,
//        VOL=SER=ZASYS1,SPACE=(CYL,(20))

```

Con todo esto, estamos en condiciones de lanzar el RMM.

1.3.5.4 Ejecución de DFSMSrmm

Para ejecutar el RMM, basta con lanzar la STC creada en el punto 1.1.5.2, con un comando de consola **S DFRMM**.

En este punto, tenemos TODOS los subsistemas listos para funcionar.

1.4 Puesta en Marcha del VTS

En este punto, estamos listos para poner en marcha nuestro particular VTS. Para ello, seguiremos los distintos pasos por plataforma:

1.4.1 PC SERVER

Desde nuestro servidor Windows, lanzaremos el MPCPCServerAll.EXE y lo dejaremos funcionando. Nos aseguraremos de que la carpeta creada que alojará las cintas virtuales tenga suficiente capacidad. Por defecto, cada cinta ocupa 1 GB, por lo que en función del uso que le hagamos al VTS, debemos asegurarnos de tener espacio en disco suficiente.

1.4.2 z/OS

En el z/OS tenemos que lanzar la Started Task que pondrá en memoria el otro extremo de la comunicación con el MFNetDisk.

Por tanto, desde la consola maestra, lanzaremos la Started Task haciendo un **S MPC, PRM=COLD** es decir, la primera vez ejecutaremos un COLD-START.

```
*12.15.21 *IWM048E WLM RUNNING IN GOAL MODE WITH THE DEFAULT
* POLICY
- 12.23.38 STC00036 $HASP373 MPC STARTED
- 12.23.39 STC00036 MPC999I COLD START REQUEST FOUND
- 12.23.39 STC00036 MPC999I HW XXXXXX,XXX,XXXXXX SW z/OS 01.10.00
- 12.23.39 STC00036 MPC001I MPC PRO VER 13.00 - 07/21/11 STARTED
- 12.23.39 STC00036 MPC051I MPC PTFID=00173
- 12.23.39 STC00036 MPC035I ECSA GETMAIN SIZE 000016K
- 12.23.39 STC00036 MPC999I IO ROUTINE FLAGS =7F395CE7 7F394C67
- 12.23.39 STC00036 MPC999I START TPS INITIALIZATION
- 12.23.39 STC00036 MPC999I TPS INITIALIZATION WAS OK
- 12.23.39 STC00036 MPC999I START TWR INITIALIZATION
00- 12.23.51 STC00036 MPC999I MPCLOG DATA VALIDATION ENDED OK
- 12.23.51 STC00036 MPC999I COLD START ENDED SUCCESSFULLY <TWR>
- 12.23.51 STC00036 MPC999I TWR INITIALIZATION WAS OK
- 12.23.51 STC00036 MPC999I START TTK INITIALIZATION
- 12.23.51 STC00036 MPC999I TTK INITIALIZATION WAS OK
- 12.23.51 STC00036 MPC043I VERSION TOKENID <0064>
- 12.23.51 STC00036 MPC999I SERVER INITIALIZATION COMPLETED SUCCESSFULLY
- 12.23.51 STC00036 MPC999I CONFIG UPDATE ENDED OK
IEE612I CN=L700 DEVNUM=0700 SYS=ADCD
-
IEE163I MODE= RD
MA a 22/003
Conectado a un sistema principal/servidor remoto localhost mediante un puerto 23
```

Fig. 45: Mensajes de Inicialización del MPC

Si no están lanzados, conviene lanzar también el OAM y el DFSMSrmm, con sus respectivos Strated Tasks (OAM1 y DFRMM, respectivamente).

NOTA: Al lanzar el OAM puede salir este mensaje:

```

12.25.38 STC00037 $HASP373 OAM1      STARTED
12.25.38 STC00037 CBR0001I OAM initialization starting.
12.25.38 STC00036 MPC999I SYSTEM RESET TO PC
12.25.39 STC00037 *CBR3660A Enter MEDIA2 scratch volumes into library
MPCROBOT.
12.25.39 STC00037 CBR0002I OAM initialization completed.

```

Esto es debido a que como tenemos menos de 100 cintas SCRATCH, el sistema nos avisa de ello.

Según nuestro miembro de parámetros MPCOPT que reside dentro MPC.XXXX.SAMPLIB, tenemos configurado 8 dispositivos (especificados en el parámetro RANGE en la configuración TAPEDRV), por lo que si ahora hacemos un **DS P,580,8** nos debería salir un listado como el que sigue:

```

UNIT DTYPE  M CNT VOLSER  CHPID=PATH STATUS
0580,3490L ,F,000,      ,00=+ 07=+
0581,3490L ,F,000,      ,00=+ 07=+
0582,3490L ,F,000,      ,00=+ 07=+
0583,3490L ,F,000,      ,00=+ 07=+
0584,3490L ,F,000,      ,00=+ 07=+
0585,3490L ,F,000,      ,00=+ 07=+
0586,3490L ,F,000,      ,00=+ 07=+
0587,3490L ,F,000,      ,00=+ 07=+
***** SYMBOL DEFINITIONS *****
F = OFFLINE                      + = PATH AVAILABLE

```

Como sale el símbolo + es que está disponible. Por tanto, lo siguiente sería hacer un

V 580-587,ONLINE y pondríamos los drives de nuestra robótica ONLINE.

	UNIT	TYPE	STATUS	VOLSER	VOLSTATE
	0580	349M	0 -R		/REMOV
	0581	349M	0 -R		/REMOV
	0582	349M	0 -R		/REMOV
	0583	349M	0 -R		/REMOV
	0584	349M	0 -R		/REMOV
	0585	349M	0 -R		/REMOV
00	0586	349M	0 -R		/REMOV
	0587	349M	0 -R		/REMOV

Si hacemos lo mismo con los rangos 590 a 597, todo debería funcionar igual. En estos momentos, tenemos nuestra robótica virtual (bocas 580-587) y unidades standalone 3490E (bocas 590-597), en línea y listas para trabajar.

El comando de SMS de consola siguiente, nos dirá el estado de nuestra robótica:

D SMS,LIB(MPCROBOT),DETAIL

Resultado:

```

CBR1110I OAM library status:
TAPE      LIB  DEVICE      TOT  ONL  AVL  TOTAL  EMPTY  SCRTCH  ON  OP
LIBRARY   TYP  TYPE       DRV  DRV  DRV  SLOTS  SLOTS  VOLS   Y   Y
MPCROBOT  ML   MANUAL      16   8   8    0     0     0    Y   Y
-----
MEDIA      SCRATCH      SCRATCH      SCRATCH
TYPE       COUNT      THRESHOLD    CATEGORY
MEDIA2      14        100        0000
-----
LIBRARY ID: 11777

```

Se puede observar como de los 16 Drives que definimos en el HCD, solo tenemos 8 activos. Esto lo podemos cambiar cambiando el parámetro RANGE de TAPEDRV en el miembro MPCOPT, de 8 a 16, por lo que podríamos tener todos los drives disponibles.

Por otra parte, también se observa como no hay volúmenes SCRATCH de ningún tipo. Para añadir volúmenes y mas operatorias, se explicarán en el punto siguiente.

1.5 Operación del VTS

En este último apartado, explicaré algunas de las operatorias para trabajar con nuestra robótica virtual.

1.5.1 AÑADIR CINTAS SCRATCH AL VTS

Para añadir cintas Scratch a nuestro robot, tenemos que tener en cuenta lo siguiente:

- 1.- Debemos comprobar en el PCTool –servidor Windows- que nuestro rango de cintas definidas es el correcto. No funcionará bien si tienes definidas en el PCServer 32 cintas, y quieres definir la número 33.
- 2.- Asimismo, debemos comprobar también que el número de cintas a definir dentro del miembro MPCOPT no excede del parámetro COUNT de la línea TAPELOC. Si queremos añadir mas cintas, debemos incrementar tanto este número como el TapesRange del PCTool.

Con esto en cuenta, y estando dentro del rango, para añadir una cinta al robot virtual, introduciremos este comando en la consola maestra:

```
LI ENT A00000,MPCROBOT,MEDIA2
```

El resultado será el siguiente:

```

13.02.42          CBR3610I Volume entry processing.
The following volumes were entered into library MPCROBOT.
A00000

```

NOTA: Por defecto, este volumen se da de alta como PRIVATE. Para cambiarlo a SCRATCH, basta con hacer un alter en el ISMF opción 2 (Volume) y luego 3 (Mountable Tape) con el fin de pasarlo a SCRATCH.

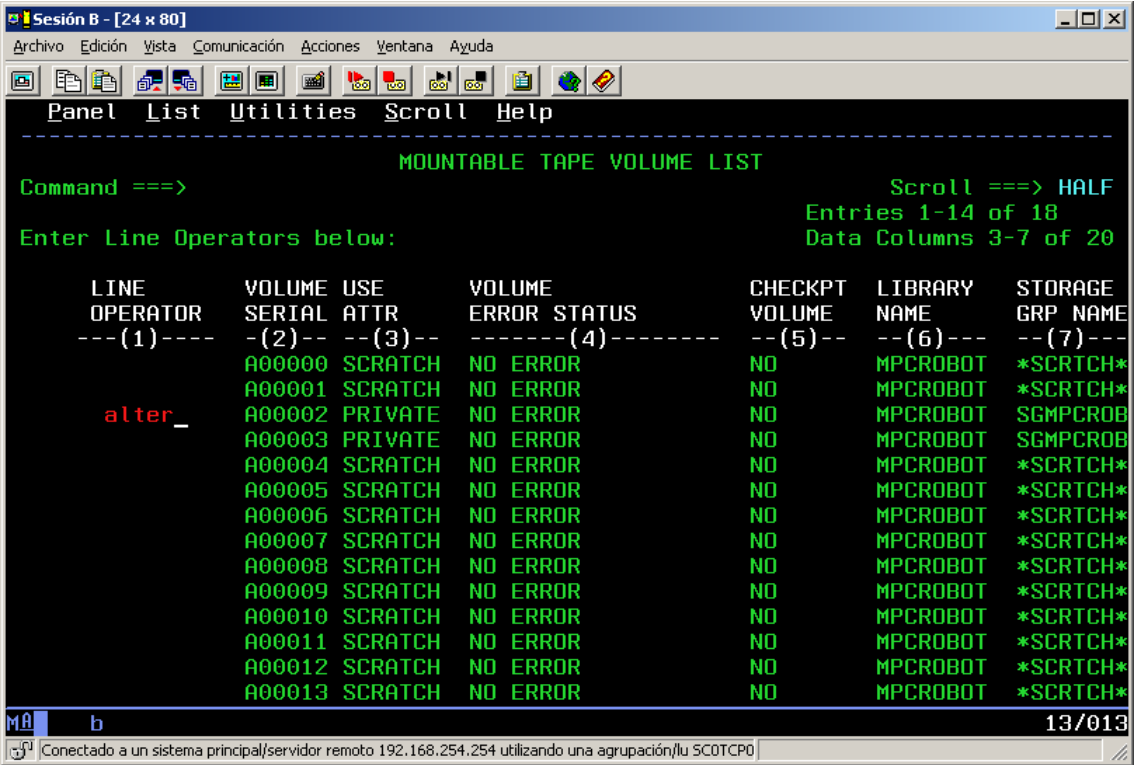


Fig. 46: Alteración de volumen de cinta desde ISMF

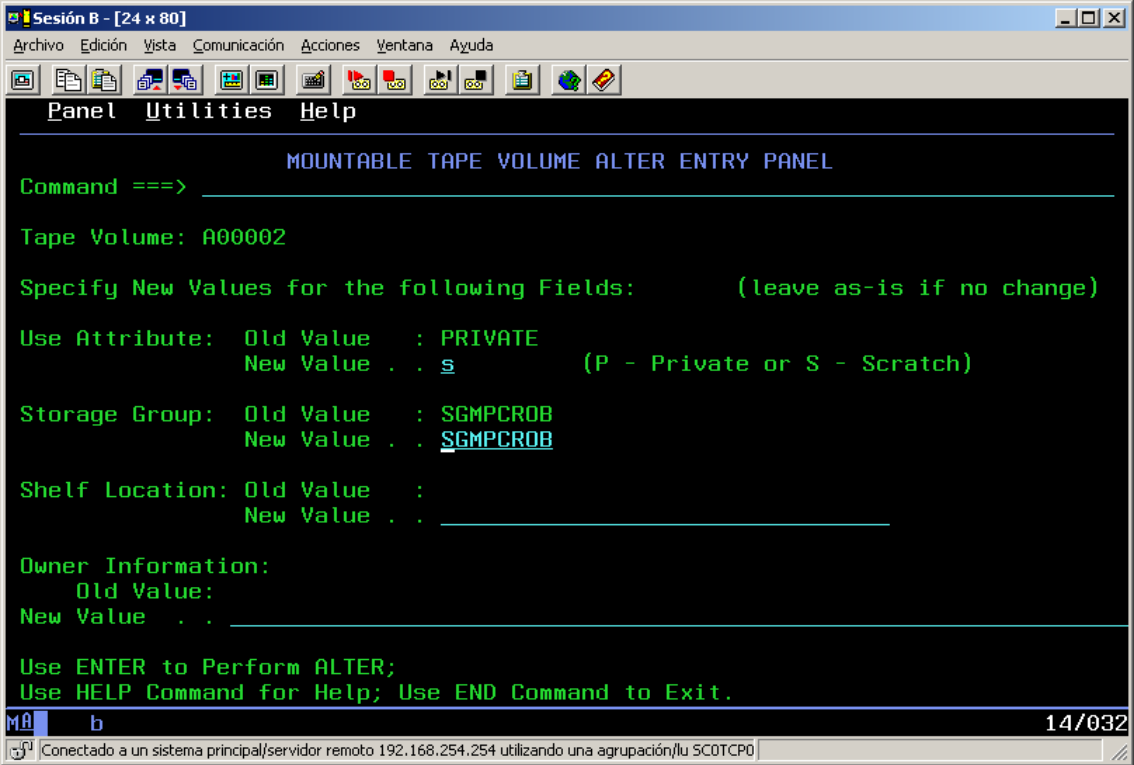


Fig. 47: Cambio de Private por Scratch

La otra opción consistiría en cambiar la definición de la robótica, de forma que cuando se inserten nuevas cintas, se hagan de forma Scratch:

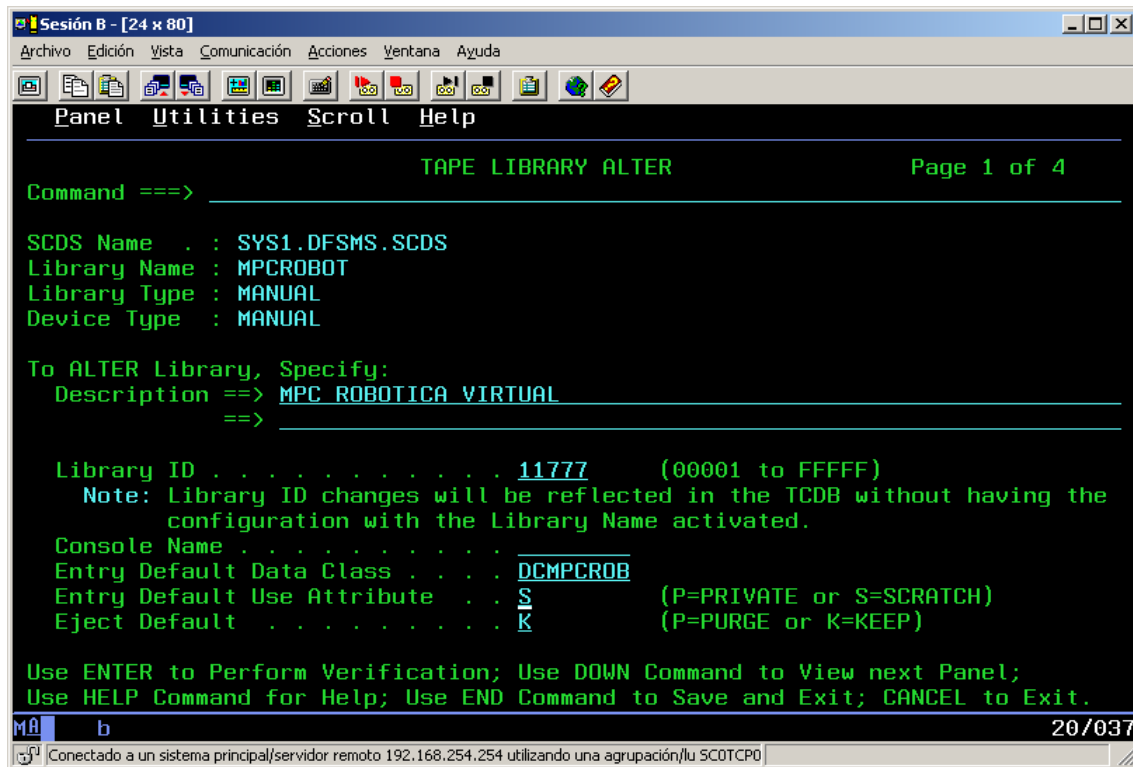


Fig. 48: Cambio de configuración Robótica Virtual

1.5.2 PONER ONLINE U OFFLINE LA ROBÓTICA VIRTUAL

Para controlar la operación de la robótica virtual, se ejecutarían los siguiente comandos de SMS:

V SMS, LIB(MPCROBOT), ONLINE para poner la robótica en línea.

V SMS, LIB(MPCROBOT), OFFLINE para poner la robótica fuera de línea.