VSE Data Spaces: Part II

BY LEO J. LANGEVIN

In last month's column I showed you the basic internal workings of data spaces. This month, I'll examine what you need to know in order to write a program that actually uses data spaces. The concluding column will show you how to write this type of program.

Releasing storage from a data space allows you to release any contiguous area of storage within a data space to the data space pool. This is also the recommended method of deleting records from a data space because of the speed in which the process is performed.

CREATING A DATA SPACE

Whenever a program creates a data space its PSW key will need to be set to zero. This is done by issuing the following macro:

```
MODESET KEY=ZERO
```

Once a program has a PSW key of zero, you will need to decide how you want the data space to be used. Figure 1 presents a sample CREATE entry with DSPSERV.

Note: Of course, don't forget to put a non-blank character in column 72 in the first 10 lines to indicate a continuance.

LAYOUT OF THE DSPSERV MACRO

NAME: one- to eight-character name associated with the data space.

BLOCKS: total and initial number of blocks that will be allocated when the data space is defined. The maximum amount is 524,288 blocks of 4KB each for a maximum size of 4GB.

GENNAME: tells the system to only use the NAME provided (NO), create a unique name if the NAME is already in use (COND), or always define a unique name for the program (YES). If the system does generate a name, it will be returned in the fields defined in the GENNAME operand.

SCOPE: defines how many tasks can access this defined data space. SINGLE will indicate that only tasks running in the same partition will be able to access the data space. COMMON and ALL will allow any task to access and use the defined data space. Where they differ is COMMON will cause the ALET to be the same for every user, while ALL will cause a unique ALET to be required for every user.

KEY: contains the key required by the accessing program in order to be authorized to access this area.

CALLERKEY: specifies that the caller's PSW key will be the storage protection key for the data space (default).

FPROT: tells the system to fetch protect this area. Granted, you cannot execute a program from a data space, but you can store a program in a data space and then fetch it for execution later. A good example is defining a VSE library in a data space to eliminate real I/O while program fetching. The default is NO.

The final three operands, *STOKEN*, *ORIGIN*, and *NUMBLKS*, are names of fields where the information is to be stored after DSPSERV has completed its processing.

DELETING A DATA SPACE

Deleting a data space is easy. All you need is to know the STOKEN and then allow the system to do the rest. You would perform this process as follows:

```
DSPSERV DELETE,STOKEN=FIELD1
```

where "FIELD1" is the double word (eight-byte) field where the definition is held.

Figure 1: Sample CREATE Entry With DSPSERV

```
[NAME] DSPSERV  CREATE, NAME=dspname,
                GENNAME=YES|NO|COND,
                OUTNAME=outname,
                BLOCKS=totalblks,
                SCOPE=SINGLE|ALL|COMMON,
                CALLERKEY,
                KEY=key,
                FPROT=YES|NO,
                STOKEN=stoken,
                ORIGIN=origin,
                NUMBLKS=numbLks
```

RELEASING A DATA SPACE

Releasing storage from a data space allows you to release any contiguous area of storage within a data space to the data space pool. This is also the recommended method of deleting records from a data space because of the speed in which the process is performed. You only need the STOKEN (of course), START, and BLOCKS operands. START will be the starting address within the data space. Since all blocks are 4KB in length, this value must be on a 4KB boundary. Since full blocks are removed, any records sharing this block will also be deleted. Following is an example of deleting the first three blocks of a data space:

```
DSPSERV RELEASE,STOKEN=FIELD1,
     START=0,BLOCKS=3
```

EXTENDING DATA SPACES

If you are using a data space and you reach the end of the space but have more data to insert, you can extend the top of that data space to allow for those additional records. Note, however, that you can only extend to a maximum of 2GB of total data space and cannot exceed the amount of space available from the data space pool. Following is an example of extending the top of the data space:

```
DSPSERV EXTEND,STOKEN=FIELD1,
       BLOCKS=TOTBLKS,
       NUMBLKS=AMOUNT
```

In this example, FIELD I contains the eight-byte STOKEN value. TOTBLKS is a full-word containing the total number of blocks that you want the data space to use. AMOUNT is a full-word that will contain the number of blocks extended after DSPSERV processing has completed. This request will fail if you try to expand beyond the maximum amount defined during the creation of this data space.

As a final note, remember that you can use registers in lieu of field names when you are using the various operands within the DSPSERV and ALESERV macros.

The concluding column in this series will examine the "cookie" of data spaces — the space token, or "STOKEN." ts

Leo J. Langevin is the Lead Developer for NFS for VSE from Connectivity Systems. He has been involved with VSE Internals since its inception. He can be reached at LEO@TCPIP4VSE.COM.