OpenVMS Technical Journal V11

Hints and Tricks When Using Dynamic Volume Expansion (DVE) on OpenVMS Systems

Rob Eulenstein, Multivendor Systems Engineering

Overview

This article provides hints and tricks for using Dynamic Volume Expansion (DVE) on OpenVMS systems. Dynamic volume expansion, which was first available in OpenVMS Alpha Version 7.3-2 and HP OpenVMS for Integrity servers Version 8.2, allows system managers to increase the size (the number of logical blocks) of a mounted volume. Although often associated with OpenVMS Host-Based Volume Shadowing, DVE can be implemented on nonshadowed volumes.

Introduction

The early groundwork for DVE began with OpenVMS Alpha Version 7.2. Prior to that release, the size of the [000000]BITMAP.SYS file was limited to 256 blocks. In Version 7.2, the maximum size of BITMAP.SYS was increased to 65,536 blocks. The initial impact of this change was to allow disks to be initialized with much smaller disk cluster sizes than was previously possible. Prior to Version 7.2, the smallest disk cluster size was approximately equal to the total blocks on the volume divided by 1 million (255 x 4096). For large volumes that contained many small files, this limitation resulted in much wasted space. With the change in Version 7.2, volumes as large as 400 GB could have a disk cluster size of 3 blocks. As we discuss in this article, allowing for a larger BITMAP.SYS file was a key prerequisite for DVE.

At the present time, the maximum volume size supported by OpenVMS is slightly less than 1 TB. The actual number is 2,147,475,456 blocks decimal or 7FFFE000 hexadecimal. The OpenVMS file system (Extended QIO Processor, or XQP) as well as current versions of SYS$DKDRIVER enforce this limit. For the history buffs among us, in OpenVMS VAX Version 5.5-2 and earlier versions, the maximum supported volume size was 16,777,215 blocks decimal or 00FFFFFF hexadecimal. The maximum number of files on an OpenVMS volume is 16,711,679. This value is calculated as follows: 2**24 – 2**16 – 1.

Prior to OpenVMS Alpha Version 7.3-2, when a volume became full, the system manager had a couple of choices. They could delete or purge files on the volume to free up some space, or they could move all the files and directories via BACKUP or COPY to a larger volume. The output of the SHOW DEVICE/FULL command was often used to monitor free space. The output showed the total blocks on the volume plus the number of free blocks on the volume. The total blocks was the value from the unit control block (UCB) field UCB$L_MAXBLOCK. This same value was also stored in the storage control block (SCB) field SCB$L_VOLSIZE. (The SCB is virtual block number, or VBN, 1 of BITMAP.SYS.) Figure 1 is an example of the output of the pre-Version 7.3-2 SHOW/DEVICE/FULL command.

```
$ show device dsa218/full

Disk DSA218:, device type MSCP served SCSI disk, is online, mounted, file-
    oriented device, shareable, available to cluster, error logging is enabled.

    Error count                    0    Operations completed            670148
    Owner process                 ""    Owner UIC                  [SYS,SYSTEM]
    Owner process ID        00000000    Dev Prot        S:RWPL,O:RWPL,G:RW,W:RW
    Reference count                1    Default buffer size                512
    Total blocks             2050353    Sectors per track                   57
    Total cylinders             2570    Tracks per cylinder                 14

    Volume label             "DISK18"    Relative volume number              0
    Cluster size                   3    Transaction count                   1
    Free blocks              1708314    Maximum files allowed          256294
.
.
.
```

Figure 1 – Pre-Version 7.3-2 SHOW DEVICE/FULL Command Output

To help manage DVE, two new items were added to the SHOW DEVICE/FULL output in OpenVMS Alpha Version 7.3-2: Logical Volume Size and Expansion Size Limit. Logical Volume Size is the value from the SCB$L_VOLSIZE field in the SCB. Expansion Size Limit is calculated when the volume is mounted, as shown in Figure 2. The value calculated is stored in the volume control block (VCB) field VCB$L_EXPSIZE.

```
Expansion Size Limit = (blocks allocated to BITMAP.SYS – 1) *
                       disk cluster size * 4096
```

Figure 2 – Calculating Expansion Size Limit

Figure 3 shows an example of the new SHOW DEVICE/FULL output.

```
$ show device/full dsa218:

Disk DSA218:, device type MSCP served SCSI disk, is online, mounted, file-
    oriented device, shareable, available to cluster, error logging is enabled,
    device supports bitmaps (no bitmaps active).

    Error count                    0    Operations completed               228
    Owner process                 ""    Owner UIC               [FIELD,SYSTEM]
    Owner process ID        00000000    Dev Prot            S:RWPL,O:RWPL,G:R,W
    Reference count                1    Default buffer size                512
    Total blocks             2050353    Sectors per track                   57
    Total cylinders             2570    Tracks per cylinder                 14
    Logical Volume Size      2050353    Expansion Size Limit            2052096

    Volume label             "DISK18"   Relative volume number                0
    Cluster size                   3    Transaction count                     1
    Free blocks              1708314    Maximum files allowed            256294
.
.
.
```

Figure 3 – Version 7.3-2 SHOW DEVICE/FULL Output

In this version of the command output, the Total blocks value is still displayed and continues to represent UCB$L_MAXBLOCK from the UCB. The UCB$L_MAXBLOCK value represents the number of logical blocks presented to OpenVMS by the storage controller. Logical Volume Size is the number of logical blocks on the volume currently in use by the OpenVMS file system (the XQP). In Figure 3, the Total blocks value equals the Logical Volume Size value, which is often the case on disks that cannot be expanded (local SCSI disks), or on disks that have already completed DVE. Expansion Size Limit represents the largest number of logical blocks that can be mapped by the current size of the BITMAP.SYS file. In Figure 3, Expansion Size Limit is slightly higher than Total blocks and Logical Volume Size for two reasons:

- The number of blocks allocated to BITMAP.SYS must be divisible by the disk cluster size.
- Logical Volume Size might not be evenly divisible by 4096 (512 bytes x 8 bits per byte).

Figure 4 shows the relationship between Expansion Size Limit and the size of the BITMAP.SYS file.

```
$ dir/size=all dsa218:[000000]bitmap.sys

Directory DSA218:[000000]
```

```
BITMAP.SYS;1              168/168

Total of 1 file, 168/168 blocks.

$ expsize = (168 – 1) * 3 * 4096

$ show symbol expsize
  EXPSIZE = 2052096
```

Figure 4 – Relationship between Expansion Size Limit and size of BITMAP.SYS

In addition to the new items added to the SHOW DEVICE/FULL output, new qualifiers were added for the DCL commands INITIALIZE and SET VOLUME. The INIT/LIMIT, INIT/SIZE, SET VOLUME/LIMIT, and SET VOLUME/SIZE commands were added to implement DVE. For both commands, the /LIMIT qualifier affects the size of BITMAP.SYS and, ultimately, the Expansion Size Limit value for the volume. The /SIZE qualifier sets or increases (you can not decrease) the Logical Volume Size. The Logical Volume Size value can never exceed the Expansion Size Limit for the volume.

How DVE works

Now that we have some background, let's see how DVE works. If you are initializing a new disk from an OpenVMS system, preparing for a future expansion of this volume is easy. Simply add the /LIMIT qualifier to the INIT command. This results in preallocating sufficient blocks to BITMAP.SYS to map up to the current 1TB volume-size limit. The actual size of the BITMAP.SYS file created depends on the disk cluster size on the volume. In Figure 5, OpenVMS defaults to a disk cluster size of 8 blocks because 8 blocks is the smallest disk cluster size possible for a 1TB volume.

```
$ init/limit $1$dga150: scratch
%INIT-I-DEFCLUSTER, value for /CLUSTER defaulted to 8

$ mount/system $1$dga150: scratch
%MOUNT-I-MOUNTED, SCRATCH mounted on _$1$DGA150: (HSV2AL)

$ show device/full $1$dga150:

Disk $1$DGA150: (HSV2AL), device type HSV210, is online, mounted, file-oriented
    device, shareable, available to cluster, device has multiple I/O paths,
    error logging is enabled.

    Error count                    0    Operations completed              1274
    Owner process                 ""    Owner UIC                      [SYSTEM]
    Owner process ID        00000000    Dev Prot            S:RWPL,O:RWPL,G:R,W
    Reference count                1    Default buffer size                512
    Current preferred CPU Id       0    Fastpath                             1
    WWID   01000010:6005-08B4-0010-3F6B-0000-9000-052E-0000
    Total blocks            75497472    Sectors per track                  128
    Total cylinders             4608    Tracks per cylinder                128
    Logical Volume Size     75497472    Expansion Size Limit         2147450880
    Allocation class               1

    Volume label           "SCRATCH"    Relative volume number               0
    Cluster size                   8    Transaction count                    1
    Free blocks             75427792    Maximum files allowed         16711679
.
.
.

$ dir/size=all $1$dga150:[000000]bitmap.sys

Directory $1$DGA150:[000000]

BITMAP.SYS;1           2305/65536
```

```
Total of 1 file, 2305/65536 blocks.

$ expsize = (65536 - 1) * 4096 * 8

$ show symbol expsize
  EXPSIZE = 2147450880   Hex = 7FFF8000
```

Figure 5 – INIT/LIMIT Command Example

In Figure 5, if 1DGA150: becomes full, the volume can be expanded without dismounting. The system manager must first increase the size of 1DGA150: from the storage controller. Because 1DGA150: is an EVA-based disk (LUN), the system manager can use the StorageWorks utility Command View EVA to increase the size of 1DGA150:. In this example the size was increased from 36 GB to 72 GB.

Next, the system manager enters the SET VOLUME/SIZE command on the OpenVMS system. If the volume is mounted by multiple cluster nodes, it is necessary to enter the SET VOLUME/SIZE command from only one node. The other cluster nodes acknowledge the larger volume size the next time an I/O involving the XQP occurs. In Figure 6, the blocks allocated to the BITMAP.SYS file do not increase; only the "used" blocks increase.

```
At this point $1$DGA150: was grown from 36 GB to 72 GB at the controller.

$ set volume/size $1$dga150:

$ show device/full $1$dga150:

Disk $1$DGA150: (HSV2AL), device type HSV210, is online, mounted, file-oriented
    device, shareable, available to cluster, device has multiple I/O paths,
    error logging is enabled.

    Error count                    0    Operations completed              8045
    Owner process                 ""    Owner UIC                      [SYSTEM]
    Owner process ID        00000000    Dev Prot            S:RWPL,O:RWPL,G:R,W
    Reference count                1    Default buffer size                512
    Current preferred CPU Id       0    Fastpath                             1
    WWID   01000010:6005-08B4-0010-3F6B-0000-9000-052E-0000
    Total blocks           150994944    Sectors per track                  128
    Total cylinders             9216    Tracks per cylinder                128
    Logical Volume Size    150994944    Expansion Size Limit         2147450880
    Allocation class               1

    Volume label           "SCRATCH"    Relative volume number               0
    Cluster size                   8    Transaction count                    1
    Free blocks            150925264    Maximum files allowed         16711679
.
.
.

$ dir/size=all $1$dga150:[000000]bitmap.sys

Directory $1$DGA150:[000000]

BITMAP.SYS;1            4609/65536

Total of 1 file, 4609/65536 blocks.
```

Figure 6 – SET VOLUME/SIZE Command Example

As easy and straightforward as this example is, most system managers do not have the luxury of being able to start with a brand new disk. Rather, they must implement DVE on existing volumes that

already contain data. In these cases, the SET VOLUME/LIMIT command must be used instead of the INIT/LIMIT command to extend the existing BITMAP.SYS file. The one significant restriction when using SET VOLUME/LIMIT is that **the volume must be mounted privately.** Failure to do so results in the following error:

```
$ set volume/limit $1$dga150:
%SET-E-NOTMOD, $1$DGA150: not modified
-SET-W-NOTPRIVATE, device must be mounted privately
```

This restriction is by far the biggest hurdle that system managers face when using DVE. It is the one aspect of DVE that is not truly dynamic. However, once BITMAP.SYS has been extended, the Logical Volume Size value can be increased again and again by using SET VOLUME/SIZE commands while the volume is mounted using the /SYSTEM or /CLUSTER qualifier, as shown in Figure 7.

```
$ init $1$dga150: scratch     ←——  Notice That /LIMIT Was Not Specified And
                                   Disk Cluster Size Defaulted to 145 Blocks

$ mount/system $1$dga150: scratch
%MOUNT-I-MOUNTED, SCRATCH mounted on _$1$DGA150: (HSV2AL)

$ show device/full $1$dga150:

Disk $1$DGA150: (HSV2AL), device type HSV210, is online, mounted, file-oriented
    device, shareable, available to cluster, device has multiple I/O paths,
    error logging is enabled.

    Error count                    0    Operations completed              8962
    Owner process                 ""    Owner UIC                      [SYSTEM]
    Owner process ID        00000000    Dev Prot            S:RWPL,O:RWPL,G:R,W
    Reference count                1    Default buffer size                512
    Current preferred CPU Id       0    Fastpath                             1
    WWID   01000010:6005-08B4-0010-3F6B-0000-9000-052E-0000
    Total blocks           150994944    Sectors per track                  128
    Total cylinders             9216    Tracks per cylinder                128
    Logical Volume Size    150994944    Expansion Size Limit          171642880
    Allocation class               1

    Volume label           "SCRATCH"    Relative volume number               0
    Cluster size                 145    Transaction count                    1
    Free blocks            150993575    Maximum files allowed           517105
.
.
.

At this point $1$DGA150: was grown from 72 GB to 144 GB at the controller.

$ set volume/limit $1$dga150:
%SET-E-NOTMOD, $1$DGA150: not modified
-SET-W-NOTPRIVATE, device must be mounted privately

$ set volume/size $1$dga150:

$ show device/full $1$dga150:

Disk $1$DGA150: (HSV2AL), device type HSV210, is online, mounted, file-oriented
    device, shareable, available to cluster, device has multiple I/O paths,
    error logging is enabled.

    Error count                    0    Operations completed              9121
    Owner process                 ""    Owner UIC                      [SYSTEM]
    Owner process ID        00000000    Dev Prot            S:RWPL,O:RWPL,G:R,W
    Reference count                1    Default buffer size                512
    Current preferred CPU Id       0    Fastpath                             1
    WWID   01000010:6005-08B4-0010-3F6B-0000-9000-052E-0000
    Total blocks           301989888    Sectors per track                  128
```

```
    Total cylinders            18432    Tracks per cylinder                128
    Logical Volume Size    171642880    Expansion Size Limit           171642880
    Allocation class               1

    Volume label           "SCRATCH"    Relative volume number              0
    Cluster size                 145    Transaction count                   1
    Free blocks            171641430    Maximum files allowed          517105
.
.
.

Notice that the above SET VOLUME/LIMIT command failed; however, the SET
VOLUME/SIZE command did increase the "Logical Volume Size" up to maximum number
of blocks that the current size of BITMAP.SYS could map.

$ dismount $1$dga150:

$ mount/over=id $1$dga150:
%MOUNT-I-MOUNTED, SCRATCH mounted on _$1$DGA150: (HSV2AL)

$ set volume/limit $1$dga150:  <----  Command Works; Volume Mounted Privately

$ set volume/size $1$dga150:           <----  This Bug Will Be Fixed In The
%SET-E-NOTSET, error modifying $1$DGA150:     Next Round Of F11X Remedial Kits;
-SYSTEM-F-BADPARAM, bad parameter value       Workaround Is To DISMOUNT And
                                              Then MOUNT
$ dismount $1$dga150:

$ mount/system $1$dga150: scratch
%MOUNT-I-MOUNTED, SCRATCH mounted on _$1$DGA150: (HSV2AL)

$ set volume/size $1$dga150:

$ show device/full $1$dga150:

Disk $1$DGA150: (HSV2AL), device type HSV210, is online, mounted, file-oriented
    device, shareable, available to cluster, device has multiple I/O paths,
    error logging is enabled.

    Error count                    0    Operations completed               9788
    Owner process                 ""    Owner UIC                      [SYSTEM]
    Owner process ID        00000000    Dev Prot            S:RWPL,O:RWPL,G:R,W
    Reference count                1    Default buffer size                 512
    Current preferred CPU Id       0    Fastpath                              1
    WWID   01000010:6005-08B4-0010-3F6B-0000-9000-052E-0000
    Total blocks           301989888    Sectors per track                   128
    Total cylinders            18432    Tracks per cylinder                 128
    Logical Volume Size    301989888    Expansion Size Limit          2152366080
    Allocation class               1

    Volume label           "SCRATCH"    Relative volume number              0
    Cluster size                 145    Transaction count                   1
    Free blocks            301984975    Maximum files allowed          517105
.
```

Figure 7 – Expanding an Existing Volume

Dynamic volume expansion also works in conjunction with another new feature in OpenVMS Alpha Version 7.3-2: dissimilar device shadowing (DDS). Prior to Version 7.3-2, all members of a host-based shadow set (DSA*nnnn*: *device*) were required to have the same number of logical blocks. Dissimilar device shadowing allows a larger disk to be added to an existing shadow set. Once the full copy operation completes, the smaller shadow members can be removed from the shadow set. Dynamic volume expansion can then be used on the virtual unit (the DSA*nnnn*: *device*) to increase the Logical Volume Size value.

Note: At some point the virtual unit must be mounted privately to extend BITMAP.SYS. Removing a member, performing DVE on this removed member, and then adding this removed member back into

the shadow set accomplishes nothing because the removed member becomes a full copy target when it is re-added to the shadow set. The following scenario illustrates the recommended way to migrate the data on an existing host-based shadow set to a larger volume.

A system manager has an existing 2-member shadow set, DSA100:, which is running low on free blocks. The system manager wants to use DDS and DVE to migrate the data on this shadow set to a larger volume. During the migration, down time must be kept to a minimum and availability of the data on this shadow set must be kept to a maximum. The current physical members of DSA100: are 1DGA160: and 1DGA170:. Both of these disks (LUNs) are 18 GB in size. At the storage controller, the system manager has created two new 36GB disks -- 1DGA180: and 1DGA190: -- and has presented these new disks to OpenVMS. Figure 8 shows the initial setup in this scenario.

```
$ show device dsa100:

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
DSA100:                 Mounted              0  PROD_DATA     1248084   323   3
$1$DGA160:    (HSV2AL)  ShadowSetMember      0  (member of DSA100:)
$1$DGA170:    (HSV2AL)  ShadowSetMember      0  (member of DSA100:)

$ show device/full dsa100:

Disk DSA100:, device type Generic SCSI disk, is online, mounted, file-oriented
    device, shareable, available to cluster, error logging is enabled, device
    supports bitmaps (no bitmaps active).

    Error count                    0    Operations completed           57833416
    Owner process                 ""    Owner UIC                         [SYSTEM]
    Owner process ID        00000000    Dev Prot            S:RWPL,O:RWPL,G:R,W
    Reference count              322    Default buffer size                512
    Total blocks            37748736    Sectors per track                  128
    Total cylinders             2304    Tracks per cylinder                128
    Logical Volume Size     37748736    Expansion Size Limit           39100416

    Volume label         "PROD_DATA"    Relative volume number              0
    Cluster size                  37    Transaction count                 325
    Free blocks              1248084    Maximum files allowed          496693
.
.
.

$ init $1$dga180: scratch
$ init $1$dga190: scratch

$ mount/over=id $1$dga180:
%MOUNT-I-MOUNTED, SCRATCH mounted on _$1$DGA180: (HSV2AL)
$ mount/over=id $1$dga190:
%MOUNT-I-MOUNTED, SCRATCH mounted on _$1$DGA190: (HSV2AL)

$ show device $1$dga180:

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
$1$DGA180:    (HSV2AL)  Mounted alloc        0  SCRATCH      75496527     1   1

$ show device $1$dga190:

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
$1$DGA190:    (HSV2AL)  Mounted alloc        0  SCRATCH      75496527     1   1

$ dismount $1$dga180:
$ dismount $1$dga190:
```

Figure 8 – Using DDS and DVE on an Existing Shadow Set – Initial Setup

Step 1: The first step is to add one of the larger, 36GB disks to the DSA100: shadow set, and then to allow the full copy operation to complete. The shadow set now contains three physical members, 1DGA160:, 1DGA170:, and 1DGA180:, as shown in Figure 9.

```
$ mount/system dsa100:/shadow=$1$dga180: prod_data
%MOUNT-I-MOUNTED, PROD_DATA mounted on _DSA100:
%MOUNT-I-SHDWMEMCOPY, _$1$DGA180: (HSV2AL) added to the shadow set with a copy
operation
%MOUNT-I-ISAMBR, _$1$DGA160: (HSV2AL) is a member of the shadow set
%MOUNT-I-ISAMBR, _$1$DGA170: (HSV2AL) is a member of the shadow set


$ show device dsa100:

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
DSA100:                 Mounted              0  PROD_DATA     1248084   336   3
$1$DGA160:    (HSV2AL)  ShadowSetMember      0  (member of DSA100:)
$1$DGA170:    (HSV2AL)  ShadowSetMember      0  (member of DSA100:)
$1$DGA180:    (HSV2AL)  ShadowCopying        0  (copy trgt DSA100:   8% copied)

.
.
.

$ show device dsa100:

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
DSA100:                 Mounted              0  PROD_DATA     1248084   318   3
$1$DGA160:    (HSV2AL)  ShadowSetMember      0  (member of DSA100:)
$1$DGA170:    (HSV2AL)  ShadowSetMember      0  (member of DSA100:)
$1$DGA180:    (HSV2AL)  ShadowSetMember      0  (member of DSA100:)
```

Figure 9 – Using DDS and DVE on an Existing Shadow Set – Step One

Step 2: At this point, one of the original, smaller disks (1DGA160:) can be removed from the shadow set and the second, larger disk (1DGA190:) can be added into the shadow set. As before, allow the full copy operation to complete. Figure 10 illustrates this step.

```
$ dismount $1$DGA160:

$ mount/system dsa100:/shadow=$1$dga190 prod_data
%MOUNT-I-MOUNTED, PROD_DATA mounted on _DSA100:
%MOUNT-I-SHDWMEMCOPY, _$1$DGA190: (HSV2AL) added to the shadow set with a copy
operation
%MOUNT-I-ISAMBR, _$1$DGA170: (HSV2AL) is a member of the shadow set
%MOUNT-I-ISAMBR, _$1$DGA180: (HSV2AL) is a member of the shadow set

$ show device dsa100:

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
DSA100:                 Mounted              0  PROD_DATA     1248084   297   3
$1$DGA170:    (HSV2AL)  ShadowSetMember      0  (member of DSA100:)
$1$DGA180:    (HSV2AL)  ShadowSetMember      0  (member of DSA100:)
$1$DGA190:    (HSV2AL)  ShadowCopying        0  (copy trgt DSA100:  27% copied)

.
.

$ show device dsa100:

Device                  Device           Error    Volume         Free  Trans Mnt
```

```
 Name                  Status           Count     Label       Blocks Count Cnt
DSA100:                Mounted              0  PROD_DATA     1248084   302   3
$1$DGA170:    (HSV2AL) ShadowSetMember      0  (member of DSA100:)
$1$DGA180:    (HSV2AL) ShadowSetMember      0  (member of DSA100:)
$1$DGA190:    (HSV2AL) ShadowSetMember      0  (member of DSA100:)
```

Figure 10 – Using DDS and DVE on an Existing Shadow Set – Step Two

Step 3: The third step is to remove the one remaining small disk (1DGA170:) from the shadow set. Disk DSA100: now has just 1DGA180: and 1DGA190: as its physical members; however, there is still only 18GB of usable space on the volume. The Total blocks value has increased to 36 GB, but the Logical Volume Size value is still 18 GB, as shown in Figure 11.

```
$ dismount $1$DGA170:

$ show device dsa100:

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label       Blocks Count Cnt
DSA100:                 Mounted              0  PROD_DATA     1248084   291   3
$1$DGA180:    (HSV2AL)  ShadowSetMember      0  (member of DSA100:)
$1$DGA190:    (HSV2AL)  ShadowSetMember      0  (member of DSA100:)

$ show device/full dsa100:

Disk DSA100:, device type Generic SCSI disk, is online, mounted, file-oriented
    device, shareable, available to cluster, error logging is enabled, device
    supports bitmaps (no bitmaps active).

    Error count                    0    Operations completed             892037
    Owner process                 ""    Owner UIC                      [SYSTEM]
    Owner process ID        00000000    Dev Prot            S:RWPL,O:RWPL,G:R,W
    Reference count              282    Default buffer size                 512
    Total blocks            75497472    Sectors per track                   128
    Total cylinders             4608    Tracks per cylinder                 128
    Logical Volume Size     37748736    Expansion Size Limit           39100416

    Volume label         "PROD_DATA"    Relative volume number                0
    Cluster size                  37    Transaction count                   285
    Free blocks              1248084    Maximum files allowed            496693
.
.
.
```

Figure 11 – Using DDS and DVE on an Existing Shadow Set – Step Three

Step 4: We have maintained data availability during the migration by ensuring that there are at least two members in the shadow set at all times. This final step requires a short period of down time because the volume was not originally initialized with the /LIMIT qualifier. We have to use the SET VOLUME/LIMIT command to extend BITMAP.SYS, and this command requires that the volume be mounted privately. On each cluster node, the system manager must reduce the transaction count to 1 prior to dismounting DSA100:, as shown in Figure 12.

```
$ show device dsa100:

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label       Blocks Count Cnt
DSA100:                 Mounted              0  PROD_DATA     1248084     1   3
$1$DGA180:    (HSV2AL)  ShadowSetMember      0  (member of DSA100:)
$1$DGA190:    (HSV2AL)  ShadowSetMember      0  (member of DSA100:)

$ dismount/cluster dsa100:
```

```
$ mount/over=id DSA100:/shadow=($1$DGA180:,$1$DGA190:)
%MOUNT-I-MOUNTED, PROD_DATA mounted on _DSA100:
%MOUNT-I-SHDWMEMSUCC, _$1$DGA180: (HSV2AL) is now a valid member of the shadow
set
%MOUNT-I-SHDWMEMSUCC, _$1$DGA190: (HSV2AL) is now a valid member of the shadow
set

$ set volume/limit dsa100:

$ dismount dsa100:

$ mount/cluster DSA100:/shadow=($1$DGA180:,$1$DGA190:) prod_data
%MOUNT-I-MOUNTED, PROD_DATA mounted on _DSA100:
%MOUNT-I-SHDWMEMSUCC, _$1$DGA180: (HSV2AL) is now a valid member of the shadow
set
%MOUNT-I-SHDWMEMSUCC, _$1$DGA190: (HSV2AL) is now a valid member of the shadow
set

$ set volume/size dsa100:

$ show device/full dsa100:

Disk DSA100:, device type Generic SCSI disk, is online, mounted, file-oriented
    device, shareable, available to cluster, error logging is enabled, device
    supports bitmaps (no bitmaps active).

    Error count                    0    Operations completed             22573
    Owner process                 ""    Owner UIC                     [SYSTEM]
    Owner process ID        00000000    Dev Prot            S:RWPL,O:RWPL,G:R,W
    Reference count               47    Default buffer size                512
    Total blocks            75497472    Sectors per track                  128
    Total cylinders             4608    Tracks per cylinder                128
    Logical Volume Size     75497472    Expansion Size Limit        2147491840

    Volume label         "PROD_DATA"    Relative volume number              0
    Cluster size                  37    Transaction count                  50
    Free blocks             38982904    Maximum files allowed          496693
.
.
.
```

Figure 12 – Using DDS and DVE on an Existing Shadow Set – Final Step

In Figure 12, notice that the Expansion Size Limit for DSA100: is now 1 TB; therefore, future dynamic volume expansions on this shadow set can be done without mounting privately. Disk DSA100: can be grown to 72 GB, 144 GB, or even larger while the volume is mounted and in use clusterwide.

A Closer Look at BITMAP.SYS

The preceding scenario makes multiple references to the BITMAP.SYS file and to the SCB, which is VBN number 1 in the BITMAP.SYS file. The following discussion takes a more in-depth look at some of the fields in a SCB and at the contents of an actual bitmap block. The SCB shown in Figure 13 is from the DSA100: shadow set, which we used in the previous DDS/DVE example. I have identified some of the more important and interesting fields.

```
$ set default dsa100:[000000]

$ dump/block=count=1 bitmap.sys

Dump of file DSA100:[000000]BITMAP.SYS;1 on 30-AUG-2007 07:54:05.47
```

```
File ID (2,2,0)   End of file block 500 / Allocated 14171

Virtual block number 1 (00000001), 512 (0200) bytes

 00000080 00000001 04800000 00250201 ................ 000000
                   ^^^^^^^^ <------ SCB$L_VOLSIZE Field - 75497472 Decimal
 00000000 0000000E 00001200 00000080 ................ 000010
 21772020 20415441 445F444F 52500001 ..PROD_DATA   .. 000020
 ^^^^
 00A6CDB6 BAE76216 000000A6 CDB6B5A9 ................ 000030
 |||||||| ||||||||     ^^^^ ^^^^^^^^ <------ SCB$Q_MOUNTTIME Field
 ^^^^^^^^ ^^^^^^^^ <------ SCB$Q_GENERNUM Field
 00000000 A0A00001 12610064 00000000 ................ 000040
                   ^^^^^^^^ ^^^^^^^^ <------ SCB$Q_UNIT_ID Field - Virtual Unit
 10E100BE 00000001 10E100B4 00000001 ................ 000050
 |||||||| |||||||| ^^^^^^^^ ^^^^^^^^ <------ SCB$Q_MEMBER_IDS (Index 0)
 ^^^^^^^^ ^^^^^^^^ <------ SCB$Q_MEMBER_IDS (Index 1)
 00000202 022CF4A8 00000000 00000000 .........,..... 000060
                   ^^^^^^^^ ^^^^^^^^ <------ SCB$Q_MEMBER_IDS (Index 2)
 00000000 00000000 00000000 00000000 ................ 000070
 00000000 00000000 00000000 00000000 ................ 000080
 00000000 00000000 00000000 00000000 ................ 000090
 00000000 00000000 00000000 00000000 ................ 0000A0
 00000000 00000000 00000000 00000000 ................ 0000B0
 00000000 00000000 00000000 00000000 ................ 0000C0
 00000000 00000000 00000000 00000000 ................ 0000D0
 00000000 00000000 00000000 00000000 ................ 0000E0
 00000000 00000000 00000000 00000000 ................ 0000F0
 00000000 00000000 00000000 00000000 ................ 000100
 00000000 00000000 00000000 00000000 ................ 000110
 00000000 00000000 00000000 00000000 ................ 000120
 00000000 00000000 00000000 00000000 ................ 000130
 00000000 00000000 00000000 00000000 ................ 000140
 00000000 00000000 00000000 00000000 ................ 000150
 00000000 00000000 00000000 00000000 ................ 000160
 00000000 00000000 00000000 00000000 ................ 000170
 00000000 00000000 00000000 00000000 ................ 000180
 00000000 00000000 00000000 00000000 ................ 000190
 00000000 00000000 00000000 00000000 ................ 0001A0
 00000000 00000000 00000000 00000000 ................ 0001B0
 00000000 00000000 00000000 00000000 ................ 0001C0
 00000000 00000000 00000000 00000000 ................ 0001D0
 00000000 00000000 00000000 00000000 ................ 0001E0
 E9A10004 00000000 00000000 00000000 ..............!i 0001F0
 ||||^^^^ <------ SCB$W_SHADOWING_STATUS Field
 ^^^^ <------ SCB$W_CHECKSUM Field
```

Figure 13 – Example Storage Control Block (SCB)

The SCB$Q_UNIT_ID field and the SCB$Q_MEMBER_IDS fields have a unique format. In each of these quadword fields, the low-order longword contains the allocation class, the low-order word in the high-order longword is the unit number, and the high-order word in the high-order longword contains a special 5-bit encoding scheme for the controller designation. Let's decode the SCB$Q_UNIT_ID field and the SCB$Q_MEMBER_IDS fields.

The SCB$Q_UNIT_ID field contains 12610064 00000000. The allocation class is 0 (low-order longword), and the unit number is 100 decimal (low-order word in the high-order longword). The 1261 hex is 0001 0010 0110 0001 in binary. Starting with bit 0 on the right, group these bits into groups of 5 bits: 0 00100 10011 00001. Ignore the lone 0 on the left and translate the 5-bit groups into decimal to yield 4, 19 and 1. The corresponding letters of the alphabet are D (4th letter), S (19th letter), and A (1st letter). This yields DSA as the controller designation. The complete device name is DSA100:.

The SCB$Q_MEMBER_IDS field for index 0 contains 10E100B4 00000001. The allocation class is 1 (low-order longword), and the unit number is 180 decimal (low-order word in the high-order longword). The 10E1 hex is 0001 0000 1110 0001 in binary. Starting with bit 0 on the right, group these bits into groups of 5 bits: 0 00100 00111 00001. Ignore the lone 0 on the left and translate the 5-bit groups into decimal to yield 4, 7, and 1. The corresponding letters of the alphabet are D (4th letter), G (7th letter), and A (1st letter). This yields DGA as the controller designation. The complete device name is 1DGA180:.

The SCB$Q_MEMBER_IDS field for index 1 contains 10E100BE 00000001. The allocation class is 1 (low-order longword), and the unit number is 190 decimal (low-order word in the high-order longword). The 10E1 hex is 0001 0000 1110 0001 in binary. Starting with bit 0 on the right, group these bits into groups of 5 bits: 0 00100 00111 00001. Ignore the lone 0 on the left and translate the 5-bit groups into decimal to yield 4, 7, and 1. The corresponding letters of the alphabet are D (4th letter), G (7th letter), and A (1st letter). This yields DGA as the controller designation. The complete device name is 1DGA190:.

The SCB$Q_MEMBER_IDS field for index 2 contains 00000000 00000000 because there is no third member in this shadow set.

Figure 14 provides a closer look at the contents of an actual bitmap block. In a bitmap block, a clear bit (0) means the corresponding disk cluster is allocated. A set bit (1) means the corresponding disk cluster is free.

```
$ dump/block=(start=2,count=1) bitmap.sys

Dump of file DSA100:[000000]BITMAP.SYS;1 on 30-AUG-2007 10:50:33.45
File ID (2,2,0)   End of file block 500 / Allocated 14171

Virtual block number 2 (00000002), 512 (0200) bytes

 00000000 00000000 00000000 00000000 ................ 000000
 00000000 00000000 00000000 00000000 ................ 000010
 00000000 00000000 00000000 00000000 ................ 000020
 00000000 00000000 00000000 00000000 ................ 000030
 00000000 00000000 00000000 00000000 ................ 000040
 00000000 00000000 00000000 00000000 ................ 000050
 00000000 00000000 00000000 00000000 ................ 000060
 00000000 00000000 00000000 00000000 ................ 000070
 00000000 00000000 00000000 00000000 ................ 000080
 00000000 00000000 00000000 00000000 ................ 000090
 00000000 00000000 00000000 00000000 ................ 0000A0
 00000000 00000000 00000000 00000000 ................ 0000B0
 00000000 00000000 00000000 00000000 ................ 0000C0
 00000000 00000000 00000000 00000000 ................ 0000D0
 00000000 00000000 00000000 00000000 ................ 0000E0
 00000000 00000000 00000000 00000000 ................ 0000F0
 00000000 00000000 00000000 00000000 ................ 000100
 00000000 00000000 00000000 00000000 ................ 000110
 00000000 00000000 00000000 00000000 ................ 000120
 00000000 00000000 00000000 00000000 ................ 000130
 00000000 00000000 00000000 00000000 ................ 000140
 00000000 00000000 00000000 00000000 ................ 000150
 00000000 00000000 00000000 00000000 ................ 000160
 00000000 00000000 00000000 00000000 ................ 000170
 00000000 00000000 00000000 00000000 ................ 000180
 00000000 00000000 00000000 00000000 ................ 000190
 00000000 00000000 00000000 00000000 ................ 0001A0
 00000000 00000000 00000000 00000000 ................ 0001B0
 00000000 00000000 00000000 00000000 ................ 0001C0
 00000000 00000000 00000000 00000000 ................ 0001D0
 00000000 00000000 00000000 00000000 ................ 0001E0
 00000000 00000000 00000000 00000000 ................ 0001F0
```

Figure 14 – Example Bitmap Block – All Disk Clusters Allocated

The bitmap block in Figure 14 is not very interesting. All the bits are clear, which means the first 4096 disk clusters (the first 151,552 blocks) on DSA100: are allocated. The 151,552 number was calculated by multiplying 4096 times 37, the disk cluster size. What would this same bitmap block look like if a large file on DSA100: was deleted, as shown in Figure 15?

```
$ dir

Directory DSA100:[000000]

000000.DIR;1        BACKUP.SYS;1        BADBLK.SYS;1        BADLOG.SYS;1
BIG_FILE.DAT;1      BITMAP.SYS;1        CONTIN.SYS;1        CORIMG.SYS;1
INDEXF.SYS;1        SECURITY.SYS;1      VOLSET.SYS;1

Total of 11 files.

$ delete/log BIG_FILE.DAT;1
%DELETE-I-FILDEL, DSA100:[000000]BIG_FILE.DAT;1 deleted (36500019 blocks)

$ dump/block=(start=2,count=1) bitmap.sys

Dump of file DSA100:[000000]BITMAP.SYS;1 on 30-AUG-2007 10:52:56.69
File ID (2,2,0)   End of file block 500 / Allocated 14171

Virtual block number 2 (00000002), 512 (0200) bytes
 FFFFFFFF FFFFFFFF FFFFFFFF F7FFFFFC ................ 000000
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000010
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000020
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000030
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000040
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000050
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000060
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000070
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000080
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000090
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 0000A0
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 0000B0
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 0000C0
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 0000D0
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 0000E0
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 0000F0
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000100
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000110
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000120
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000130
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000140
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000150
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000160
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000170
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000180
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 000190
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 0001A0
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 0001B0
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 0001C0
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 0001D0
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 0001E0
 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF ................ 0001F0
```

Figure 15 – Example Bitmap Block – Most Disk Clusters Free

This reveals quite a different picture. Most of the disk clusters are now free. The very first longword in this block contains F7FFFFFC. All of the other longwords contain FFFFFFFF. In F7FFFFFC, bits 0, 1,

and 27 are clear. This means that out of the 151,552 blocks mapped by this bitmap block, only blocks (LBNs) 0 through 73 and 999 through 1035 are allocated.

Actual DDS/DVE Cases

The topics discussed in the next few paragraphs were taken directly from calls about DVE and DDS that were logged to the Customer Support Center. In most cases, these problems and pitfalls have been reported on more than one occasion.

When a customer implements DVE, one error they might encounter if they run ANALYZE/DISK is: %ANALDISK-I-SHORTBITMAP, the storage bitmap on RVN 1, does not cover the entire device. Although this informational error appears ominous, in almost every case it can be safely ignored. To understand this SHORTBITMAP error, we need to examine the conditions under which it occurs.

A word-length field at offset 508 decimal (1FC hex) in the SCB is used to hold status flags. The name of this field is SCB$W_SHADOWING_STATUS. As of OpenVMS Version 7.3-2, three bits are defined in this field. Bit 0 is the SCB$V_INIT_NO_ERASE bit. This bit is set when the INIT/SHADOW/NOERASE command was used to initialize the volume. Bit 1 is the SCB$V_DVE_ENABLED bit. This bit is set when either the Expansion Size Limit or the Logical Volume Size value does not equal the Total blocks value when the volume is initialized. Bit 2 is the SCB$V_HBVS_MEMBERS_MAY_DIFFER bit. Figure 16 shows the SCB$W_SHADOWING_STATUS field in the SCB.

```
$ dump/block=count=1 dsa600:[000000]bitmap.sys

Dump of file DSA600:[000000]BITMAP.SYS;1 on  8-OCT-2006 09:17:28.70
File ID (2,2,0)   End of file block 1087 / Allocated 65536

Virtual block number 1 (00000001), 512 (0200) bytes

 00000020 00000001 021EAE18 00080201 .....Z...... ... 000000
 00000000 00000006 000087AC 00000020  ...Z........... 000010
 79212020 20314B53 49445245 53550001 ..USERDISK1   !y 000020
 00A59093 1A8A34FA 000000A5 8D85D4E3 cT..#...z4....#. 000030
 .
 .
 .
 00000000 00000000 00000000 00000000 ................ 0001D0
 00000000 00000000 00000000 00000000 ................ 0001E0
 A5120002 00000000 00000000 00000000 ...............% 0001F0
     ^^^^<---- SCB$W_SHADOWING_STATUS Field, Bits 0 & 2 Are Clear, Bit 1 Is Set
```

Figure 16 - SCB$W_SHADOWING_STATUS Field in an SCB

The SHORTBITMAP error occurs only if the SCB$V_DVE_ENABLED bit (bit 1) is clear in the SCB$W_SHADOWING_STATUS field, and if the Total blocks value on the volume is greater than the current Logical Volume Size. Initializing a disk device with either the /SIZE or the /LIMIT qualifier sets the SCB$V_DVE_ENABLED bit. Using the SET VOLUME/LIMIT command does not set the SCB$V_DVE_ENABLED bit.

```
$ show device/full $6$dka200:

Disk $6$DKA200: (WSC236), device type COMPAQ BF01885A34, is online, mounted,
    file-oriented device, shareable, served to cluster via MSCP Server, error
    logging is enabled.

    Error count                    0    Operations completed            814899
    Owner process                 ""    Owner UIC                     [SYSTEM]
```

```
    Owner process ID        00000000    Dev Prot            S:RWPL,O:RWPL,G:R,W
    Reference count                1    Default buffer size                 512
    Current preferred CPU Id       0    Fastpath                              1
    Total blocks            35565080    Sectors per track                    32
    Total cylinders            34732    Tracks per cylinder                  32
    Logical Volume Size      5000000    Expansion Size Limit         2147450880
    Allocation class               6
    .
    .
    .



$dump/block=count=1 $6$dka200:[000000]bitmap.sys

Dump of file $6$DKA200:[000000]BITMAP.SYS;1 on  8-OCT-2006 09:56:23.13
File ID (2,2,0)   End of file block 154 / Allocated 65536

Virtual block number 1 (00000001), 512 (0200) bytes

 0000002F 00000001 004C4B40 00080201 ....@KL...../... 000000
 00000000 0000000E 000008D8 0000002F /...X........... 000010
 FF822020 20202048 43544152 43530001 ..SCRATCH     .. 000020
 00000000 00000000 000000A5 90A480AB +.Z.#........... 000030
 .
 .
 .
 00000000 00000000 00000000 00000000 ................ 0001D0
 00000000 00000000 00000000 00000000 ................ 0001E0
 90D00000 00000000 00000000 00000000 ..............Z. 0001F0
     ^^^^ <------  SCB$W_SHADOWING_STATUS Field, SCB$V_DVE_ENABLED Bit Is Clear


$ analyze/disk $6$dka200:
Analyze/Disk_Structure for _$6$DKA200: started on  8-OCT-2006 09:56:37.97
%ANALDISK-I-SHORTBITMAP, storage bitmap on RVN 1 does not cover the entire device
%ANALDISK-I-OPENQUOTA, error opening QUOTA.SYS
-SYSTEM-W-NOSUCHFILE, no such file
```

Figure 17 – Example of the %ANALDISK-I-SHORTBITMAP Error

There is no corruption on this volume. The conditions described are likely to occur if the system manager has grown the volume from the storage controller and has used the SET VOLUME/LIMIT command, but has not yet used the SET VOLUME/SIZE command.

Another pitfall when attempting to expand a volume is the dreaded SYSTEM-W-DEVICEFULL error. Often the volume is being expanded because free space is insufficient. Why, then, do you get this error (shown in Figure 18), which essentially tells you what you already know?

```
$ set volume/limit $1$dga300:
%SET-E-NOTSET, error modifying $1$DGA300:
-SYSTEM-W-DEVICEFULL, device full; allocation failure
```

Figure 18 – Example SYSTEM-W-DEVICEFULL Error

The "device full" error occurs when there is insufficient contiguous free space on the volume to create the new, larger BITMAP.SYS file. If the volume in question is totally full, you have no choice but to free up some disk space and then re-enter the SET VOLUME/LIMIT command. If there is some free space on the volume, try a small increase in volume size first. Then, using this newly created space (which, by definition, has to be contiguous), enter the SET VOLUME/LIMIT command again to extend BITMAP.SYS to map up to 1 TB. In Figure 19, the volume in question was originally 36 GB and was nearly out of free space. From the storage controller, the system manager had grown the volume to 72 GB.

```
$ mount/over=id $1$dga300:
%MOUNT-I-MOUNTED, SCRATCH mounted on _$1$dga300: (HSV2AL)

$ set volume/limit $1$dga300:
%SET-E-NOTSET, error modifying $1$DGA300:
-SYSTEM-W-DEVICEFULL, device full; allocation failure

$ set volume/limit=80000000 $1$dga300:

$ dismount $1$dga300:

$ mount/over=id $1$dga300:
%MOUNT-I-MOUNTED, SCRATCH mounted on _$1$dga300: (HSV2AL)

$ set volume/size $1$dga300:

$ set volume/limit $1$dga300:

$ dismount $1$dga300:

$ mount/system $1$dga300: scratch
%MOUNT-I-MOUNTED, SCRATCH mounted on _$1$dga300: (HSV2AL)

$ set volume/size $1$dga300:
```

Figure 19 – Workaround to the SYSTEM-W-DEVICEFULL Error

Do not try to expand a volume by just a few blocks. If the expansion requested in the SET VOLUME/SIZE command is less than 256 times the disk cluster size, the expansion requested is ignored; however, no error is returned.

Note: The SET VOLUME/SIZE command cannot be used to decrease the size of a volume. If the new Logical Volume Size is less than the present Logical Volume Size, an SS$_UNSUPPORTED error is returned, as shown in Figure 20.

```
$ show device/full dsa100:

Disk DSA100:, device type Generic SCSI disk, is online, mounted, file-oriented
    device, shareable, available to cluster, error logging is enabled, device
    supports bitmaps (no bitmaps active).

    Error count                    0    Operations completed            297277
    Owner process                 ""    Owner UIC                      [SYSTEM]
    Owner process ID        00000000    Dev Prot            S:RWPL,O:RWPL,G:R,W
    Reference count              251    Default buffer size                512
    Total blocks             37748736    Sectors per track                  128
    Total cylinders              2304    Tracks per cylinder                128
    Logical Volume Size      37748736    Expansion Size Limit          39100416

    Volume label            "SCRATCH"    Relative volume number               0
    Cluster size                   37    Transaction count                  254
    Free blocks              37748103    Maximum files allowed           496693
.
.
.

$ set volume/size=17000000 dsa100:
%SET-E-NOTSET, error modifying _DSA100:
-SYSTEM-E-UNSUPPORTED, unsupported operation or function
```

Figure 20 – SS$_UNSUPPORTED Error

Finally, the exact number displayed for the Expansion Size Limit in the output of the SHOW DEVICE/FULL command depends on the version of the VMSxxx_MOUNT96 patch kit that is installed

on the system. As mentioned previously the Expansion Size Limit for a volume is calculated as follows:

```
Expansion Size Limit = (blocks allocated to BITMAP.SYS – 1) *
                       disk cluster size * 4096
```

Based on this formula, systems with older or no MOUNT96 kit installed might display a number larger than 2,147,475,456 decimal for Expansion Size Limit. The display of this errant number is only cosmetic. The latest versions of VMS*xxx*_MOUNT96 kits properly limit the number displayed to 2,147,475,456 decimal blocks.

Summary

Dynamic volume expansion and dissimilar device shadowing are two of the most useful new features in OpenVMS Alpha Version 7.3-2 and HP OpenVMS for Integrity servers Version 8.2. When using the INIT or SET VOLUME commands, the /LIMIT qualifier is used to preallocate space to or extend BITMAP.SYS. The /SIZE qualifier is used to set or to increase the Logical Volume Size. Remember that when you use the SET VOLUME/LIMIT command, the volume in question must be mounted privately.