Chapter 7

Creating and Administering XLV Logical Volumes

This chapter describes the procedures for creating and administering XLV logical volumes using command-line utilities. A graphical user interface for performing many of these procedures is available from the *xlvm* command. See its online help for more information about *xlvm*.

The major sections in this chapter are:

- "Verifying That Plexing Is Supported" on page 141
- "Creating Volume Objects With xlv_make" on page 142
- "Displaying Logical Volume Objects" on page 148
- "Adding a Volume Element to a Plex (Growing a Logical Volume)" on page 149
- "Adding a Plex to a Logical Volume" on page 150
- "Detaching a Plex From a Logical Volume" on page 153
- "Deleting an XLV Object" on page 154
- "Removing and Mounting a Plex" on page 155
- "Creating Plexed Logical Volumes for Root" on page 158
- "Booting the System Off an Alternate Plex" on page 160
- "Configuring the System for More Than Ten XLV Logical Volumes" on page 162
- "Converting lv Logical Volumes to XLV Logical Volumes" on page 163

Verifying That Plexing Is Supported

As discussed in Chapter 7, "Creating and Administering XLV Logical Volumes," the plexing feature of XLV, which enables the use of multiple plexes, is available only when you purchase the Disk Plexing Option software option. It requires a NetLS license which must be installed in a nonstandard location, */etc/nodelock*.

You can use the *xlv_mgr* command to verify that the plexing software and a valid license are installed. Follow these steps:

1. Invoke *xlv_mgr*:

   ```
   # xlv_mgr
   ```

2. Give the **show config** command:

   ```
   xlv_mgr> show config
   Allocated subvol locks: 30      locks in use: 6
   Plexing license: present
   Plexing support: present
   Maximum subvol block number: 0x7fffffffffffffff
   ```

   The third line of output, "Plexing support is present," indicates that plexing software is installed with a valid license.

3. Quit out of *xlv_mgr*:

   ```
   xlv_mgr> quit
   ```

Creating Volume Objects With *xlv_make*

The *xlv_make* command creates volumes, subvolumes, plexes, and volume elements from unused disk partitions. It writes the logical volume labels in the disk volume headers only; data on the disk partitions is untouched.

After you create a volume, you must make a filesystem on it if necessary and mount the filesystem so that you can use the logical volume.

Caution: When you create a logical volume and make a filesystem, all data already on the disk partitions is destroyed.

xlv_make can be run interactively or it can take commands from an input file. The remainder of this section gives two examples of using *xlv_make*; the first one is interactive and the second is noninteractive.

Example 1: A Simple Logical Volume

This example shows a simple logical volume composed of a data subvolume created from two entire option disks. The disks are on controller 0, drive addresses 2 and 3. An XFS filesystem is created and mounted at */vol1*.

1. Unmount the disks that will be used in the volume if they are mounted. For example:

```
# df
Filesystem                 Type  blocks     use   avail %use  Mounted on
/dev/root                   efs 1939714  430115 1509599  22%  /
/dev/dsk/dks0d2s7           efs 2004550      22 2004528   0%  /d2
/dev/dsk/dks0d3s7           efs 3826812      22 3826790   0%  /d3
# umount /d2
# umount /d3
```

2. Start *xlv_make*:

   ```
   # xlv_make
   xlv_make>
   ```

3. Start creating the volume by specifying its name, for example xlv0:

   ```
   xlv_make> vol xlv0
   xlv0
   ```

4. Begin creating the data subvolume:

   ```
   xlv_make> data
   xlv0.data
   ```

   xlv_make echoes the name of each object (volume, subvolume, plex, or volume element) you create.

5. Continue to move down through the hierarchy of the volume by specifying the plex:

   ```
   xlv_make> plex
   xlv0.data.0
   ```

6. Specify the volume elements (disk partitions) to be included in the volume, for example */dev/dsk/dks0d2s7* and */dev/dsk/dks0d3s7*:

   ```
   xlv_make> ve dks0d2s7
   xlv0.data.0.0
   xlv_make> ve dks0d3s7
   xlv0.data.0.1
   ```

   You can specify the last portion of the disk partition pathname (as shown) or the full pathname. *xlv_make* accepts disk partitions that are of types “xlv,” “xfs,” and “efs.” You can use other partition types, for example “lvol,” by giving the **-force** option, for example, *ve –force dks0d2s7*. *xlv_make* automatically changes the partition type to “xlv.”

7. Tell *xlv_make* that you are finished specifying the objects:

```
xlv_make> end
Object specification completed
```

8. Review the objects that you've specified:

```
xlv_make> show

        Completed Objects
(1)  VOL xlv0
VE xlv0.data.0.0 [empty]
        start=0, end=2004549, (cat)grp_size=1
        /dev/dsk/dks0d2s7 (2004550 blks)
VE xlv0.data.0.1 [empty]
        start=2004550, end=5831361, (cat)grp_size=1
        /dev/dsk/dks0d3s7 (3826812 blks)
```

9. Write the volume information to the logical volume labels by exiting *xlv_make*:

```
xlv_make> exit
Newly created objects will be written to disk.
Is this what you want?(yes)  yes
Invoking xlv_assemble
```

10. Make an XFS filesystem using *mkfs*, for example:

```
# mkfs /dev/dsk/xlv/xlv0
meta-data=/dev/dsk/xlv/xlv0      isize=256    agcount=8, agsize=16094 blks
data     =                       bsize=4096   blocks=2482901
log      =internal log           bsize=4096   blocks=1000
realtime =none                   bsize=4096   blocks=0, rtextents=0
             need correct command line
```

11. Mount the filesystem, for example:

```
# mkdir /vol1
# mount /dev/dsk/xlv/xlv0 /vol1
```

12. To have the logical volume mounted automatically at system startup, add an entry for the volume to */etc/fstab*, for example:

```
/dev/dsk/xlv/xlv0 /vol1 xfs rw,raw=/dev/rdsk/xlv/xlv0 0 0
```

Example 2: A Striped, Plexed Logical Volume

This example shows the noninteractive creation of a logical volume from four equal-sized option disks (controller 0, units 2 through 5). Two plexes will be created with the data striped across the two disks in each plex. The stripe unit will be 128 KB. An XFS filesystem is created and mounted at */vol1*.

1. As in the previous example, unmount filesystems on the disks to be used, if necessary.
2. Create a file, called *xlv0.specs* for example, that contains input for *xlv_make*. For this example and a volume named xlv0, the file contains:

   ```
   vol xlv0
   data
   plex
   ve -stripe -stripe_unit 256 dks0d2s7 dks0d3s7
   plex
   ve -stripe -stripe_unit 256 dks0d4s7 dks0d5s7
   end
   show
   exit
   ```

   This script specifies the volume hierarchically: volume, subvolume (data), first plex with a striped volume element, then second plex with a striped volume element. The **ve** commands have a stripe unit argument of 256. This argument is the number of 512-byte disk blocks (sectors), so 128K/512 = 256. The **end** command signifies that the specification is complete and the (optional) **show** command causes the specification to be displayed. The logical volume label is created by the **exit** command.
3. Run *xlv_make* to create the volume. For example:

   ```
   # xlv_make xlv0.specs
   ```
4. Make an XFS filesystem with an internal 10 MB log and 1 KB block size:

   ```
   # mkfs -b size=1k -l size=10m /dev/dsk/xlv/xlv0
   ```
5. Mount the filesystem, for example:

   ```
   # mkdir /vol1
   # mount /dev/dsk/xlv/xlv0 /vol1
   ```
6. To have the logical volume mounted automatically at system startup, add an entry for the volume to */etc/fstab*, for example:

   ```
   /dev/dsk/xlv/xlv0 /vol1 xfs rw,raw=/dev/rdsk/xlv/xlv0 0 0
   ```

Example 3: A Plexed Logical Volume for an XFS Filesystem With an External Log

The following example shows how you can create an XLV logical volume that has a log subvolume that is plexed and a data subvolume that is concatenated and plexed. The volume will be used to hold an XFS filesystem with an external log.

This example uses four disks on controller 1 at drive addresses 2 through 5. The disks at drive addresses 2 and 3 are partitioned as option drives with xfslog partitions. The disks at drive addresses 4 and 5 are partitioned as option drives without xfslog partitions.

1. Invoke *xlv_make* and begin to create the volume, called xfs-mp5, by creating the log subvolume with two plexes:

```
# xlv_make
xlv_make> vol xfs-mp5
xfs-mp5
xlv_make> log
xfs-mp5.log
xlv_make> plex
xfs-mp5.log.0
xlv_make> ve dks1d2s15
xfs-mp5.log.0.0
xlv_make> plex
xfs-mp5.log.1
xlv_make> ve dks1d3s15
xfs-mp5.log.1.0
```

2. Create the data subvolume with two plexes, each of which has two volume elements:

```
xlv_make> data
xfs-mp5.data
xlv_make> plex
xfs-mp5.data.0
xlv_make> ve dks1d2s7
xfs-mp5.data.0.0
xlv_make> ve dks1d4s7
xfs-mp5.data.0.1
xlv_make> plex
xfs-mp5.data.1
xlv_make> ve dks1d3s7
xfs-mp5.data.1.0
xlv_make> ve dks1d5s7
xfs-mp5.data.1.1
```

3. Indicate that you have completed the volume, display it, and exit *xlv_make*:

```
xlv_make> end
Object specification completed
xlv_make> show

        Completed Objects
(1)  VOL xfs-mp5
VE xfs-mp5.log.0.0 [empty]
        start=0, end=8255, (cat)grp_size=1
        /dev/dsk/dks1d2s15 (8256 blks)
VE xfs-mp5.log.1.0 [empty]
        start=0, end=8255, (cat)grp_size=1
        /dev/dsk/dks1d3s15 (8256 blks)
VE xfs-mp5.data.0.0 [empty]
        start=0, end=3920223, (cat)grp_size=1
        /dev/dsk/dks1d2s7 (3920224 blks)
VE xfs-mp5.data.0.1 [empty]
        start=3920224, end=7848703, (cat)grp_size=1
        /dev/dsk/dks1d4s7 (3928480 blks)
VE xfs-mp5.data.1.0 [empty]
        start=0, end=3920223, (cat)grp_size=1
        /dev/dsk/dks1d3s7 (3920224 blks)
VE xfs-mp5.data.1.1 [empty]
        start=3920224, end=7848703, (cat)grp_size=1
        /dev/dsk/dks1d5s7 (3928480 blks)

xlv_make> exit
Newly created objects will be written to disk.
Is this what you want?(yes)  y
Invoking xlv_assemble
```

4. Make an XFS filesystem by running *mkfs*. Note how *mkfs* automatically uses an external log when one is present.

```
# mkfs /dev/dsk/xlv/xfs-mp5
meta-data=/dev/dsk/xlv/xfs-mp5   isize=256    agcount=8, agsize=122636 blks
data     =                       bsize=4096   blocks=981088
log      =volume log             bsize=4096   blocks=1032
realtime =none                   bsize=65536  blocks=0, rtextents=0
```

5. Mount the filesystem, for example:

```
# mkdir /v1
# mount /dev/dsk/xlv/xfs-mp5 /v1
```

6. To have the logical volume mounted automatically at system startup, add an entry for the volume to */etc/fstab*, for example:

```
/dev/dsk/xlv/xfs-mp5 /v1 xfs rw,raw=/dev/rdsk/xlv/xfs-mp5 0 0
```

Displaying Logical Volume Objects

To get a list of the top level XLV objects on a system (volumes, unattached plexes, and unattached volume elements), invoke *xlv_mgr* and give the command **show all**, for example:

```
# xlv_mgr
xlv_mgr> show all
Volume Element: SPARE_VE
Volume:         BIG_VOLUME (complete)
```

In this example, there are two top level objects, a volume element named SPARE_VE and a logical volume named BIG_VOLUME. The volume element is a top level object because it is not part of (attached to) any plex. Volume elements can be attached to a plex at a later time.

To display the complete hierarchy of a top level object, give the *xlv_mgr* command **show object** with the name of the object, for example:

```
xlv_mgr> show object BIG_VOLUME
VOL BIG_VOLUME (complete)
VE BIG_VOLUME.log.0.0   [active]
        start=0, end=8255, (cat)grp_size=1
        /dev/dsk/dks1d2s15 (8256 blks)
VE BIG_VOLUME.log.1.0   [active]
        start=0, end=8255, (cat)grp_size=1
        /dev/dsk/dks1d3s15 (8256 blks)
VE BIG_VOLUME.log.2.0   [active]
        start=0, end=8255, (cat)grp_size=1
        /dev/dsk/dks1d4s15 (8256 blks)
VE BIG_VOLUME.data.0.0  [active]
        start=0, end=3920223, (cat)grp_size=1
        /dev/dsk/dks1d2s7 (3920224 blks)
```

```
VE BIG_VOLUME.data.1.0  [active]
        start=0, end=3920223, (cat)grp_size=1
        /dev/dsk/dks1d3s7 (3920224 blks)
VE BIG_VOLUME.data.2.0  [active]
        start=0, end=3920223, (cat)grp_size=1
        /dev/dsk/dks1d4s7 (3920224 blks)
```

This output shows that BIG_VOLUME contains log and data subvolumes. Each subvolume has three plexes that have one volume element each.

Adding a Volume Element to a Plex (Growing a Logical Volume)

Growing a logical volume (increasing its size) can be done by adding one or more volume elements to the end of one or more of its plexes. (If you don't add volume elements to all plexes, data stored in the added volume elements won't be replicated in all plexes.)

The procedure below assumes that you are starting with a logical volume. If you are starting with a filesystem on a single disk partition that you want to turn into a logical volume and grow onto an additional disk partition, use the procedure in the section "Growing an XFS Filesystem Onto Another Disk" in Chapter 4 or the section "Growing an EFS Filesystem Onto Another Disk" in Chapter 4 instead.

1. If any of the volume elements you plan to add to the volume don't exist yet, create them with *xlv_make*. For example, follow this procedure to create a volume element out of a new disk, */dev/dsk/dks0d4s7*:

   ```
   # xlv_make
   xlv_make> ve spare_ve dks0d4s7
   new_ve
   xlv_make> end
   Object specification completed
   xlv_make> exit
   Newly created objects will be written to disk.
   Is this what you want?(yes)  yes
   Invoking xlv_assemble
   ```

   The **ve** command creates a volume element name, spare_ve. The name is required because the volume element is not part of a larger hierarchy; it is the top level object in this case.

2. Use the **attach** command of the *xlv_mgr* command to add each volume element. For example, to add the volume element from step 1 to plex 0 of the data subvolume of the volume xlv0, use this procedure:

```
# xlv_mgr
xlv_mgr> attach ve spare_ve xlv0.data.0
```

3. Quit out of *xlv_mgr*:

```
xlv_mgr> quit
```

4. If you are growing an XFS filesystem, mount the filesystem if it isn't already mounted:

```
# mount volume mountpoint
```

volume is the device name of the logical volume, for example */dev/dsk/xlv/xlv0*, and *mountpoint* is the mount point directory for the logical volume.

5. If you are growing an XFS filesystem, use *xfs_growfs* to grow the filesystem:

```
# xfs_growfs -d mountpoint
```

mountpoint is the mount point directory for the logical volume.

6. If you are growing an EFS filesystem, unmount the filesystem if it is mounted, and use *growfs* to grow the filesystem:

```
# umount mountpoint
# growfs volume
```

mountpoint is the mount point directory for the filesystem. *volume* is the device name of the logical volume, for example */dev/dsk/xlv/xlv0*.

Adding a Plex to a Logical Volume

If you have purchased the Disk Plexing Option software option and have installed a NetLS license for it (remember that its NetLS license is installed in a nonstandard location, */etc/nodelock*), you can add a plex to an existing subvolume for improved reliability in case of disk failures. The procedure to add a plex to a subvolume is described below. To add more than one plex to a subvolume or to add a plex to each of the subvolumes in a volume, repeat the procedure as necessary.

1. If the plex that you want to add to the subvolume doesn't exist yet, create it with *xlv_make*. For example, to create a plex called plex1 to add to the data subvolume of a volume called root_vol, give these commands:

```
# xlv_make
xlv_make> plex plex1
plex1
xlv_make> ve /dev/dsk/dks0d3s7
plex1.0
xlv_make> end
Object specification completed
xlv_make> exit
Newly created objects will be written to disk.
Is this what you want?(yes)  yes
Invoking xlv_assemble
```

2. Use the *xlv_mgr* command to add the plex to the volume. For example, to add the standalone plex plex1 to root_vol, use this procedure:

```
# xlv_mgr
xlv_mgr> attach plex plex1 root_vol.data
```

   xlv_mgr automatically initiates a plex revive operation to copy the contents of the original plex, root_vol.data.0, to the newly added plex.

3. You can confirm that root_vol now has two plexes by displaying the object hierarchy:

```
xlv_mgr> show object root_vol
VOL root_vol (complete)
VE root_vol.data.0.0    [active]
        start=0, end=988091, (cat)grp_size=1
        /dev/dsk/dks0d2s7 (988092 blks)
VE root_vol.data.1.0    [empty]
        start=0, end=988091, (cat)grp_size=1
        /dev/dsk/dks0d3s7 (988092 blks)
```

   The newly added plex, root_vol.data.1, is initially in the "empty" state. This is because it is newly created.

4. Exit *xlv_mgr*:

```
xlv_mgr> quit
```

The plex revive completes and the new plex switches to "active" state automatically, but if you want to check its progress and verify that the plex has become active, follow this procedure:

1. List the XLV daemons running, for example:

```
# ps -ef | grep xlv
    root    27     1  0 10:49:27 ?        0:00 /sbin/xlv_plexd -m 4
    root    35     1  0 10:49:28 ?        0:00 /sbin/xlv_labd
    root    31     1  0 10:49:27 ?        0:00 xlvd
    root   407    27  1 11:01:01 ?        0:00 xlv_plexd -v 2 -n root_vol.data
-d 50331648 -b 128 -w 0 0 1992629
    root   410   397  2 11:01:11 pts/0    0:00 grep xlv
```

   One instance of the *xlv_plexd* daemon is currently reviving root_vol.data. This daemon exits when the plex has been fully revived.

2. Later, check the XLV daemons again, for example:

```
# ps -ef | grep xlv
    root    27     1  0 10:49:27 ?        0:00 /sbin/xlv_plexd -m 4
    root    35     1  0 10:49:28 ?        0:00 /sbin/xlv_labd
    root    31     1  0 10:49:27 ?        0:03 xlvd
    root   459   397  2 11:21:10 pts/0    0:00 grep xlv
```

   The instance of *xlv_plexd* that was reviving root_vol.data is no longer running; it has completed the plex revive.

3. Check the state of the plex using *xlv_mgr*:

```
# xlv_mgr
xlv_mgr> show object root_vol
VOL root_vol (complete)
VE root_vol.data.0.0    [active]
        start=0, end=988091, (cat)grp_size=1
        /dev/dsk/dks0d2s7 (988092 blks)
VE root_vol.data.1.0    [active]
        start=0, end=988091, (cat)grp_size=1
        /dev/dsk/dks0d2s0 (988092 blks)
xlv_mgr> quit
```

   Both plexes are now in the "active" state.

Detaching a Plex From a Logical Volume

Detaching a plex from a volume, perhaps because you want to swap disk drives, can be done while the volume is active. However, the entire address range of the subvolume must still be covered by active volume elements in the remaining plex or plexes. *xlv_mgr* does not allow you to detach the only active plex in a volume if the other plexes are not yet active. The procedure to detach a plex is:

1. Start *xlv_mgr* and display the volume that has the plex that you plan to detach, for example, root_vol:

   ```
   # xlv_mgr
   xlv_mgr> show object root
   VOL root (complete)
   VE root.data.0.0        [active]
           start=0, end=1843199, (cat)grp_size=1
           /dev/dsk/dks1d3s0 (1843200 blks)
   VE root.data.1.0        [active]
           start=0, end=1843199, (cat)grp_size=1
           /dev/dsk/dks1d4s0 (1843200 blks)
   ```

2. Detach plex 1 and give it the name plex1 by giving these commands:

   ```
   xlv_mgr> detach plex root.1 rplex1
   ```

3. To examine the volume and the detached plex, give these commands:

   ```
   xlv_mgr> show -long all
   PLEX rplex1
   VE rplex1.0     [stale]
           start=0, end=1843199, (cat)grp_size=1
           /dev/dsk/dks1d4s0 (1843200 blks)

   VOL root (complete)
   VE root.data.0.0        [active]
           start=0, end=1843199, (cat)grp_size=1
           /dev/dsk/dks1d3s0 (1843200 blks)
   ```

4. Exit *xlv_mgr*:

   ```
   xlv_mgr> quit
   ```

Deleting an XLV Object

Caution: The procedure in this section can result in the loss of data if it is not performed properly. It is recommended only for experienced IRIX system administrators.

You can delete a volume or any other XLV object by using the *xlv_mgr* command. The procedure is:

1. If you are deleting a volume, you must unmount it first. For example:

   ```
   # umount /vol1
   ```

2. Start *xlv_mgr* and list each object on the system:

   ```
   # xlv_mgr
   xlv_mgr> show -long all
   VOL root_vol (complete)
   VE root_vol.data.0.0    [active]
           start=0, end=988091, (cat)grp_size=1
           /dev/dsk/dks0d2s0 (988092 blks)
   VE root_vol.data.1.0    [active]
           start=0, end=988091, (cat)grp_size=1
           /dev/dsk/dks0d2s7 (988092 blks)
   ```

   This example shows one high-level object, a volume with two plexes in a data subvolume (root_vol.data.0 and root_vol.data.1). Each plex has one volume element.

3. If the element you want to delete is not a high-level object, you must first detach it from its high-level object. For example, to delete one of the plexes in the example, it must first be detached:

   ```
   xlv_mgr> detach plex root_vol.data.1 plex_to_be_deleted
   ```

   Detached objects must be given a name, in this case plex_to_be_deleted.

4. Delete the object, in this example the plex plex_to_be_deleted:

   ```
   xlv_mgr> delete object plex_to_be_deleted
   ```

5. Confirm that the object is gone:

   ```
   xlv_mgr> show -long all
   VOL root_vol (complete)
   VE root_vol.data.0.0    [active]
           start=0, end=988091, (cat)grp_size=1
           /dev/dsk/dks0d2s0 (988092 blks)
   ```

6. Exit *xlv_mgr*:

```
xlv_mgr> quit
```

Removing and Mounting a Plex

Caution: The procedure in this section can result in the loss of data if it is not performed properly. It is recommended only for experienced IRIX system administrators.

You can get a snapshot of a filesystem by removing a plex from a plexed volume and mounting that plex separately. Since you can only mount volumes, you must convert the plex into a volume. The following procedure shows you how to remove the plex from its original volume and make it into a separate volume:

1. Verify that the volume is currently not being revived. If there is a revive in progress, you should wait until the revive is done because the data among the plexes is not identical until after the plex revive is done.

```
# ps -ef | grep xlv_plexd
    root    35     1  0   Dec 13 ?        0:00 /sbin/xlv_plexd -m 4
```

   The output shows that just one copy of *xlv_plexd*, the master process, is running. If more than one copy is running, a plex revive is in progress.

2. Unmount the filesystem mounted on the logical volume, */projvol5* in this example:

```
# umount /projvol5
```

   Unmounting the filesystem puts it into a clean state.

3. Start *xlv_mgr* and display the logical volume, xfs-mp5 in this example:

```
# xlv_mgr
xlv_mgr> show object xfs-mp5
VOL xfs-mp5 (complete)
VE xfs-mp5.log.0.0 [active]
        start=0, end=8255, (cat)grp_size=1
        /dev/dsk/dks1d2s15 (8256 blks)
VE xfs-mp5.log.1.0 [active]
        start=0, end=8255, (cat)grp_size=1
        /dev/dsk/dks1d3s15 (8256 blks)
VE xfs-mp5.data.0.0 [active]
        start=0, end=3920223, (cat)grp_size=1
        /dev/dsk/dks1d2s7 (3920224 blks)
```

```
VE xfs-mp5.data.0.1 [active]
        start=3920224, end=7848703, (cat)grp_size=1
        /dev/dsk/dks1d4s7 (3928480 blks)
VE xfs-mp5.data.1.0 [active]
        start=0, end=3920223, (cat)grp_size=1
        /dev/dsk/dks1d3s7 (3920224 blks)
VE xfs-mp5.data.1.1 [active]
        start=3920224, end=7848703, (cat)grp_size=1
        /dev/dsk/dks1d5s7 (3928480 blks)
```

4. Detach the second plex from the log subvolume and call it log_copy:

   ```
   xlv_mgr> detach plex xfs-mp5.log.1 log_copy
   ```

   One of the plexes from the log subvolume must be detached because the volume that will be created with one of the data plexes must have a log subvolume to go with it.

5. Detach the second plex from the data subvolume and call it data_copy:

   ```
   xlv_mgr> detach plex xfs-mp5.data.1 data_copy
   ```

6. Display all of the high-level objects to verify that there are now one volume and two plexes:

   ```
   xlv_mgr> show all
   Volume:         xfs-mp5 (complete)
   Plex:           log_copy
   Plex:           data_copy
   ```

7. Give the **delete** command for each of the detached plexes:

   ```
   xlv_mgr> delete object log_copy
   Object log_copy deleted.

   xlv_mgr> delete object data_copy
   Object data_copy deleted.
   ```

   The **delete** command changes the logical volume information in the volume headers, but doesn't touch the data in the partitions.

8. Exit *xlv_mgr*:

   ```
   xlv_mgr> quit
   ```

9. Make the partitions from the detached plexes into a volume:

```
# xlv_make
xlv_make> vol copy
copy
xlv_make> log
copy.log
xlv_make> ve dks1d3s15
copy.log.0.0
xlv_make> data
copy.data
xlv_make> ve dks1d3s7
copy.data.0.0
xlv_make> ve dks1d5s7
copy.data.0.1
xlv_make> end
Object specification completed
xlv_make> exit
Newly created objects will be written to disk.
Is this what you want?(yes)  yes
Invoking xlv_assemble
```

10. Mount the new volume. The filesystem is still intact, so *mkfs* isn't used (using *mkfs* would erase the data).

```
# mkdir /copy
# mount /dev/dsk/xlv/copy /copy
```

11. Remount the original filesystem:

```
# mount /dev/dsk/xlv/xfs-mp5 /projvol5
```

12. Use the *ls* command to confirm that the files on the original volume also appear on the new volume that you created from the removed plex.

```
# ls /copy
autoconfig  chroot      config      cron.d
chkconfig   clri        cron        fstab
# ls /projvol5
autoconfig  chroot      config      cron.d
chkconfig   clri        cron        fstab
```

Creating Plexed Logical Volumes for Root

Caution: The procedure in this section can result in the loss of data if it is not performed properly. It is recommended only for experienced IRIX system administrators.

You can put your Root filesystem on a plexed volume for greater reliability. A plexed Root volume allows your system to continue running even if one of the root disks fails. If there is a separate Usr filesystem on the system disk, it should be plexed, too. Because the swap partition may be unavailable if the root disk fails, a spare swap partition should available on a different disk. Administering the plexes of the Root and, if present, Usr volumes and the swap partitions is easiest if each disk used in the volumes is identical and is partitioned identically.

The Root volume can contain only a data subvolume. Each plex of the data subvolume can contain only a single volume element. The volume element must contain a single disk partition.

The Root filesystem can be either an EFS filesystem or an XFS filesystem with an internal log.

Use the following procedure to create a plexed Root volume. It assumes that you are starting with a working system (not a system with an empty system disk).

1. Make the root partition into an XLV volume. In this example, the XLV volume is called xlv_root:

```
# xlv_make
xlv_make> vol xlv_root
xlv_root
xlv_make> data
xlv_root.data
xlv_make> ve -force /dev/dsk/dks0d1s0
xlv_root.data.0.0
xlv_make> end
Object specification completed
xlv_make> exit
Newly created objects will be written to disk.
Is this what you want?(yes)  yes
Invoking xlv_assemble
```

The result is an XLV volume named xlv_root that contains the root partition. Since XLV preserves the data in partitions, the contents of the root partition are preserved. The **-force** option to the **ve** command was used because a mounted partition was included in the volume.

2. Reboot the system so that the system switches from running off the root partition at */dev/dsk/dks0d1s0* to running off the logical volume */dev/dsk/xlv/xlv_root*:

```
# reboot
```

3. You can confirm that the Root volume is being used by comparing the major and minor device numbers of */dev/root* and */dev/dsk/xlv/xlv_root*:

```
# ls -l /dev/root /dev/dsk/xlv/xlv_root
brw-------    2 root     sys      192,  0 Oct 31 17:58 /dev/root
brw-------    2 root     sys      192,  0 Dec 12 17:58 /dev/dsk/xlv/xlv_root
```

4. Create the second plex, for example out of */dev/dsk/dks0d2s0*, and call the plex root_plex1:

```
# xlv_make
xlv_make> plex root_plex1
root_plex1
xlv_make> ve /dev/dsk/dks0d2s0
root_plex1.0
xlv_make> end
Object specification completed
xlv_make> exit
Newly created objects will be written to disk.
Is this what you want?(yes)  yes
Invoking xlv_assemble
```

5. Add *sash* to the volume header of the disk used for the second plex. It enables booting off of the alternate plex if the primary plex fails.

```
# dvhtool -v get sash /tmp/sash /dev/rdsk/dks0d1vh
# dvhtool -v add /tmp/sash sash /dev/rdsk/dks0d2vh
```

6. Attach the second plex to the volume using *xlv_mgr* and quit out of *xlv_mgr*:

```
# xlv_mgr
xlv_mgr> attach plex root_plex1 xlv_root.data
xlv_mgr> quit
```

When the shell prompt returns, the system automatically begins a plex revive so that the two plexes contain the same data.

7. Add procedure for creating Boot volume.

Booting the System Off an Alternate Plex

Once you have plexed the Root volumes, you can boot off a secondary plex if the primary plex becomes unavailable. Because the boot PROM does not understand logical volumes, you must manually reconfigure the system to boot the system from the disk that contains the alternate plex. The procedure for booting the system off a secondary plex depends on the model of workstation or server. The first subsection below, "CHALLENGE L, CHALLENGE XL, and CHALLENGE DM" is for those systems. For all other workstations and servers you must follow the procedure in the second subsection, "All Other Models."

CHALLENGE L, CHALLENGE XL, and CHALLENGE DM

With Challenge L, XL, and DM systems, it is possible to change the drive addresses of disks using a dial or switch. If the system disk and the alternate disk are both internal disks on the same channel and are partitioned identically, you can swap the drive addresses of the two disks. (If the system doesn't meet these requirements, use the procedure in the section "All Other Models" instead.) By exchanging the drive addresses for the system disk and the alternate disk, the system automatically boots off the alternate disk, which has become the new system disk. Follow this procedure:

1. Shut down the system. For example, use this command:

   ```
   # shutdown
   ```

2. Power off the system.
3. By manipulating the switches or dials on the system disk and the alternate disk, change each disk's drive address to the other's drive address.
4. Power up the system.

All Other Models

The following procedure describes how to boot the system off the alternate Root plex and can be used on all system. In the example used in this procedure, the system is reconfigured to boot off the partition */dev/dsk/dks0d2s0* and use partition */dev/dsk/dks0d2s1* as swap as an example. Substitute the correct partitions for your system.

1. On the System Maintenance Menu, choose `Enter Command Monitor`:

   ```
   ...
   5) Enter Command Monitor

   Option? 5
   Command Monitor.  Type "exit" to return to the menu.
   ```

2. Display the PROM environment variables:

   ```
   >> printenv
   SystemPartition=dksc(0,1,8)
   OSLoadPartition=dksc(0,1,0)
   root=dks0d1s0
   ...
   ```

   The swap PROM environment variable (which is set below) is not displayed because it is not saved in NVRAM.

3. Reset the SystemPartition, OSLoadPartition, and root environment variables to have the values of the disk partition that contains the alternate plex and the swap environment variable to have the value of the alternate swap partition:

   ```
   >> setenv SystemPartition dksc(0,2,8)
   >> setenv OSLoadPartition dksc(0,2,0)
   >> setenv root dks0d2s0
   >> setenv swap /dev/dsk/dks0d2s1
   ```

4. Exit the Command Monitor and restart the system:

   ```
   >> exit
   ...
   Option? 1
                   Starting up the system...
   ...
   ```

5. Change */dev/rswap* and */dev/swap* so that they are linked to the new swap partition:

```
# cd /dev
# ls -l dsk/dks0d1s1 dsk/dks0d2s1
brw-r-----    1 root     sys      128, 17 Sep 19 10:18 dsk/dks0d1s1
brw-r-----    2 root     sys      128, 33 Sep 19 10:18 dsk/dks0d2s1
# ls -l *swap
crw-------    2 root     sys      128, 17 Nov 18  1994 rswap
brw-r-----    2 root     sys      128, 17 Sep 19 10:18 swap
# rm rswap swap
# mknod rswap c 128 33
# mknod swap b 128 33
```

6. Reboot the system to verify that it is configured correctly:

```
# reboot
```

Configuring the System for More Than Ten XLV Logical Volumes

By default, a system can have up to ten XLV logical volumes. To increase the number of XLV logical volumes supported, you modify the file */var/sysgen/master.d/xlv*. The procedure is:

1. Using any editor, open the file */var/sysgen/master.d/xlv*, for example:

```
# vi /var/sysgen/master.d/xlv
```

2. Find this line in the file:

```
#define XLV_MAXVOLS 10
```

3. Change the 10 in this line to a higher number of your choice, for example:

```
#define XLV_MAXVOLS 20
```

4. Write the file and quit the editor.
5. Generate a new kernel:

```
# /etc/autoconfig
```

6. Reboot the system to make the change take effect:

```
# reboot
```

Converting *lv* Logical Volumes to XLV Logical Volumes

This section explains the procedure for converting *lv* logical volumes to XLV logical volumes. The files on the logical volumes are not modified or dumped during the conversion. You must be superuser to perform this procedure.

1. Choose new names for the logical volumes, if desired. XLV, unlike *lv*, only requires names to be valid filenames (except periods, ".", are not allowed in XLV names), so you can choose more meaningful names. For example, you can make the volume names the same as the mount points you use. If you mount logical volumes at */a*, */b*, and */c*, you can name the XLV volumes a, b, and c.
2. Unmount all *lv* logical volumes that you plan to convert to XLV logical volumes. For example:

   `# ` **`umount /a`**
3. Create an input script for *xlv_make* by using *lv_to_xlv*:

   `# ` **`lv_to_xlv -o`** *scriptfile*

   scriptfile is the name of a temporary file that *lv_to_xlv* creates, for example */usr/tmp/xlv.script*. It contains a series of *xlv_make* commands that can be used to create XLV volumes that are equivalent to the *lv* logical volumes listed in */etc/lvtab*.
4. If you want to change the volume names, edit *scriptfile* and replace the names on the lines that begin with `vol` with the new names. For example, change:

   `vol lv0`

   to:

   `vol a`

   The volume name can be any name that is a valid filename.
5. By default, all *lv* logical volumes on the system are converted to XLV. If you do not want all *lv* logical volumes converted to XLV, edit *scriptfile* and remove the *xlv_make* commands for the volumes that you do not want to change. See the section "Creating Volume Objects With xlv_make" in this chapter and the xlv_make(1M) reference page for more information.
6. Create the XLV volumes by running *xlv_make* with *scriptfile* as input:

   `# ` **`xlv_make`** *scriptfile*

7. If you converted all *lv* logical volumes to XLV, remove */etc/lvtab*:

   ```
   # rm /etc/lvtab
   ```

8. If you converted just some of the *lv* logical volumes to XLV, open */etc/lvtab* for editting to begin removing the entries for the logical volumes you converted.

   ```
   # vi /etc/lvtab
   ```

9. Edit */etc/fstab* so that it automatically mounts the XLV logical volumes at startup. These changes to */etc/fstab* are required for each XLV logical volume:

   - In the first field, insert the subdirectory `xlv` after `/dev/dsk`.
   - If you changed the name of the volume, for example from lv0 to a, make the change in the first field.
   - Insert the subdirectory `xlv` into the raw device name.
   - If you changed the name of the volume, for example from lv0 to a, make the change in the raw device.

   For example, if an original line is:

   ```
   /dev/dsk/lv0   /a efs rw,raw=/dev/rdsk/lv0 0 0
   ```

   the changed line, including the name change, is:

   ```
   /dev/dsk/xlv/a /a efs rw,raw=/dev/rdsk/xlv/a 0 0
   ```

10. Mount the XLV logical volume, for example:

    ```
    # mount /a
    ```

Creating a Record of XLV Logical Volume Configurations

Information about XLV objects, volumes, subvolumes, plexes, and volume elements, is stored in logical volume labels in the volume header of each disk that contains an XLV object (see the section "Volume Headers" in Chapter 1 for more information). If a logical volume label is removed, the system is unable to assemble the logical volume that includes that logical volume label, although the data in the object described in the logical volume label is still present. You can re-create the logical volume label with *xlv_make*, but only if you remember the exact configuration of the affected logical volume. The *xlv_mgr* command can be used to create a script that records the exact configuration of each logical volume on the system. This script can be given to *xlv_make* as input at a later time if it is ever necessary to re-create any of the XLV logical volumes on the system.

To create a record of the exact configuration of each XLV logical volume on the system, follow this procedure:

1. Start the *script* command, which begins capturing text on the screen, and put the captured text in the file */var/config/XLV.configuration*:

   ```
   # script /var/config/XLV.configuration
   Script started, file is XLV.configuration
   ```

2. Start *xlv_mgr*:

   ```
   # xlv_mgr
   ```

3. Give the **script -write** command to *xlv_mgr* with the name of a file that will contain the configuration information, for example */var/config/XLV.configuration*:

   ```
   xlv_mgr> script -write /var/config/XLV.configuration
   ```

4. Exit *xlv_mgr*:

   ```
   xlv_mgr> quit
   ```

5. Check the contents of the file that contains the configuration:

   ```
   # cat /var/config/XLV.confguration
   #
   # Create Volume proj_vol
   #
   vol proj_vol
   data
   plex
   ve   -force -start 0 /dev/dsk/dks1d3s11 /dev/dsk/dks1d3s12
   plex
   ve   -force -start 0 /dev/dsk/dks1d6s2 /dev/dsk/dks1d6s3
   end
   exit
   ```