

Chapter 14: Data and Tape Handling

In this chapter we introduce you to the principal data and tape handling software and facilities available at Fermilab. These include:

- **SAM** (Sequential data Access via Meta-data)
- **Enstore**
- **OCS** (Operator Communications Software)



Since many groups and experiments have customized their data and tape handling routines, some or all of the applications described here may not be available or appropriate for you. Contact your spokesperson or group leader to find out what procedures have been established for your experiment or group.

14.1 SAM (Sequential data Access via Meta-data)

SAM (Sequential data Access via Meta-data) is a file-based data management and data access program that provides an intermediate layer between data processing and data storage. A single database keeps track of metadata for every data file associated with a **SAM** installation. Documentation and information about **SAM** as implemented for D0 RunII can be found at <http://d0db.fnal.gov/sam/>.

14.2 Enstore

Enstore is the mass storage system implemented at Fermilab as the primary data store for experiments' large data sets. It provides distributed access to data on tape or other storage media both local to a user's machine and over networks. It provides a generic interface so experimenters can efficiently use mass storage systems as easily as if they were native file systems. **SAM** or another data management/access program can be made to interface to **Enstore**.

Enstore is documented at

<http://www.fnal.gov/docs/products/enstore/>. The Enstore online monitoring home page is <http://hppc.fnal.gov/enstore/>. There are currently (as of February 2003) three productions **Enstore** systems running:

- D0EN for the D0 RunII experiment
- CDFEN for the CDF RunII experiment
- STKEN for general users

14.3 OCS (Operator Communications Software)

 **OCS** is a package that performs and manages tape drive allocations, operator-assisted tape mounts and tape drive use statistics. Its logical-to-physical tape device name translation helps not only human communication, but hides many platform-specific idiosyncrasies from users.

OCS is not a tape I/O package. It works well *with* such packages (e.g., **RBIO, FMB, DAFT**).

Start-up information for running and monitoring **OCS** tape mounts is provided, and the **OCS X** interface is introduced.

OCS functions are available via three separate interfaces:

- FORTRAN/C library of subroutines
- Command line tools
- X Motif tools

The features available in each of these interfaces overlap to a great extent. However, since the different interfaces are by their nature geared towards different uses, the functionality is not 100% duplicated across them. The FORTRAN/C subroutines provide the most flexibility and functionality for users requiring multiple mount requests, the command line tools are generally used in shell scripts, and the X interfaces are very useful for monitoring tape drive statistics.

The X interfaces are fairly intuitive, so we give you enough information to bring them up (section 14.3.2 *The OCS X Interfaces*), but we do not describe them in detail.

The **OCS** functions are fully described in the *OCS Reference Guide*, document number GA0012, available on the Web. The reference guide and other documents are also available wherever **OCS** is installed and setup, in the directory `$OCS_DIR/doc`.

14.3.1 OCS Basics

Monitoring Tape Mounts

During the course of your work, you may need to monitor different aspects of the job and possibly contact the operators. **OCS** provides functionality to perform these job management activities. For your convenience, we list the appropriate command next to the function in the table below, and indicate whether you can perform the function using one of the X interfaces. You will need to see the *OCS Reference Guide* for usage information on these commands and for the associated FORTRAN/C subroutines.

| Function | Command | X interface |
|--------------------------------------------------------------------------------|----------------------------------------------------|--------------------------------|
| Mark a tape drive broken so that it will not be allocated until it is repaired | ocs_broken | xocs |
| Log statistics as to how much the drive was used | ocs_report _stat ocs_init_s tat | |
| Send an attention message to the operator | ocs_messag e | |
| Send a request to the operator to run a cleaning tape through a tape drive | ocs_clean_ it | |
| Display status of tape drives in the OCS database | ocs_tape | xocs |
| Display pending mounts | ocs_pendin g | xocs xtapevi ew |
| Display tape drive statistics | ocs_devsta t ocs_stats | xocs |
| Display tape mount log | ocs_mrlog | xocs |
| Display tape drives that may need to be cleaned | ocs_clean_ list | xocs |

Sample Tape Mounting Process

Here is a sample tape mounting process using the command line tools. Read through it to see what steps are involved and what kind of response to expect from the system at each step. Note that the **OCS** commands come with many options that are not shown here.

First, display the list of available tape drives:

```
% ocs_tape

  HOSTNAME  DEVICE      TYPE          ALLOCATED  STATUS  VSN
  USERNAME  UID    AUTH
bastet      dumdl4     DLT4000      allocated  working -
root        0          n
bastet      isis2      EXB-8505     unalloctd  working -
-           n
bastet      horus      EXB-8200     unalloctd  working -
-           y
```

Request allocation of a tape drive so no other user may access it (notice only one device shows authorization as “yes”):

```
% ocs_allocate
```

```
bastet horus
```

Send a request to the operations staff to mount a tape on the drive you have allocated:

```
% ocs_request -t horus -w -v FGMS04
```

```
ocs_request: success
```

Verify that the tape was mounted correctly:

```
% ocs_check_label -t horus -w -v FGMS04
```

```
ocs_check_label: Success
```

Set the tape drive characteristics according to your needs (we recommend that you always do this; don’t assume the drive has been left in any particular state):

```
% ocs_setdev -t horus
```

Request the appropriate device file for reading and/or writing the tape according to the characteristics you’ve set:

```
% ocs_devfile -t horus
```

```
/dev/rmt/tps0d3nrv
```

Perform your task on the loaded tape. Normally this involves running a program that calls tape I/O routines from **RBIO** or another I/O package. For simplicity in this example, we just use the UNIX **dd** utility to get the tape contents (we have loaded an ANSI initialized tape with no data):

```
% dd if=/dev/rmt/tps0d3nrv conv=unblock cbs=80
```

```

VOL1FGMS04
4
HDR1
0+2 records in
0+1 records out

```

A useful feature to include in this example is the device statistics function. It provides more detailed information for the end user than the X interface implementations, which were designed more for administrative purposes.

```
% ocs_devstat -t horus
```

```

-----
Collecting Tape Drive Statistic ----- Thu Oct
2 12:43:27 1997
-----
Host Name           = bastet
Device Name         = horus
Device Filename     = /dev/rmt/tps0d3
Device Type         = EXB-8200
Controller          = SCSI
Vendor Id           = EXABYTE
Product Id          = EXB-8200
Firmware Id         = 268N
Serial Number       = -
Number of Hours On =          -1   In Motion =          -1
Count/KBytes Read  =          2200   Write =          -1
Errors Read        =           0   Write =          -1
Comp Ratio Read    =           -1   Write =          -1
Compression        = NO
Density            = 8200
Media Type         = 133
Block Size         = 0
Block Total        = 2294048
Count Origin       = Exabyte_Extended_Sense
Remain Tape/KBytes = 2290569
SCSI Sense Code    = 0
SCSI ASCQ          = 0
Track Retry        = -1
Stop/Start Count   = -1
SCSI Test Unit Ready = 0
SCSI Sense Key     = NO_SENSE
-----
Tape Not Present:  N | Write Prot:    N | Clean Bit:    N |
Drive Cleaned:    N

```

```
Beginning of Tape: N | At File Mark: N | End of Media: N |
End of Tape:      N
Ready Bit:        Y | Power Fail:  N | SCSI ILI Bit: N |
-----
-----
```

Dismount the tape (i.e. rewind and unload it):

```
% ocs_dismount -t horus
```

```
ocs_dismount: Success
```

Finally, deallocate the tape drive so that someone else can use it:

```
% ocs_deallocate -t horus
```

```
ocs_deallocate: Deallocated :horus
```

Tape Mounts with Batch Jobs



Most of the time, users run tape mounts in conjunction with batch jobs. We strongly recommend that you include the tape mount allocation, mount request, dismount and deallocation *within* your batch job. If you don't, you risk preventing others from using the tape drive while your job is waiting to run and after it has finished. A **-q** option is provided for the **ocs_allocate** command to allow you to queue for tape drive allocation so that your job won't fail if a drive isn't immediately available when it starts to run.

Here is a sample batch job script that incorporates these recommendations:

```
#!/bin/csh
setup ocs
set td=`ocs_allocate -q -h localhost -d exabyte_850x
| cut -f2 -d" " -`
if ( $status != 0 ) then
    echo ocs_allocate failed with message: $td
    exit 1
endif
set drive=`echo $td | cut -f2 -d" " -`
ocs_request -t $drive -v fr3147 -r
if ( $status != 0 ) then
    echo ocs_request failed
    ocs_deallocate -t $drive
    exit 1
endif
set dfile=`ocs_devfile -t $drive`
if ( $status != 0 ) then
    echo ocs_devfile failed with message: $dfile
    ocs_dismount -t $drive
    ocs_deallocate -t $drive
    exit 1
```

```

endif
ocs_setdev -t $drive -v -d 8500
if ( $status != 0 ) then
    echo ocs_setdev failed
    ocs_dismount -t $drive
    ocs_deallocate -t $drive
    exit 1
endif
setenv MY_DEVFILE_VAR $dfile
e831job.run
ocs_dismount -t $drive
ocs_deallocate -t $drive
exit 0

```

14.3.2 The OCS X Interfaces



Remember that your *DISPLAY* environment variable needs to be set properly for X windows applications (see section 9.6 *Some Important Variables*).

xocs

xocs is an X interface that you may find useful for viewing tape statistics and history. It allows you to perform a subset of the functions available through text commands. After setting up **OCS**, enter **xocs** at the command line. You'll see the following screen (shown for version v3.0):

The screenshot shows the **xocs** window with a menu bar (View, Action, DB) and a title bar (xocs). The main content area is titled "Matching Tape Drives" and contains a table with the following data:

| Host | Device | Type | Allocation | Status | User | UID | PID | PGID | VSN |
|-------------|---------|----------|-------------|---------|------|-----|-------|-------|--------|
| baja | bj01 | EXB-8500 | Unallocated | working | - | - | - | - | VG0201 |
| baja | bj02 | EXB-8200 | allocated | working | root | 0 | 18426 | 18426 | IGNORE |
| bastet | dumdl14 | DLT4000 | Unallocated | working | - | - | - | - | - |
| bastet | horus | EXB-8200 | Unallocated | working | - | - | - | - | FGMS04 |
| bastet | isis2 | EXB-8505 | Unallocated | working | - | - | - | - | - |
| fakeacpmaps | fake1 | EXB-8500 | Unallocated | working | - | - | - | - | - |
| fakeacpmaps | fakes2 | EXB-8500 | Unallocated | working | - | - | - | - | - |

Below the table are filter options for Allocation, Host OS Flavor, Status, and Device Type, each with a diamond-shaped selection button and a list of options:

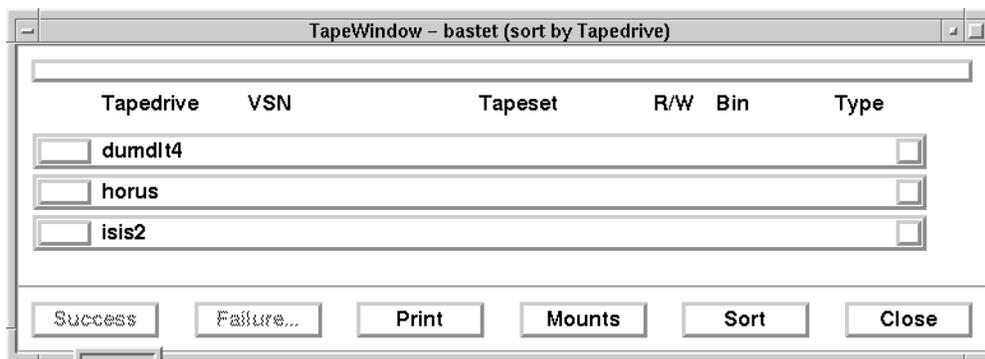
- Allocation: Allocated, Unallocated, Any
- Host OS Flavor: IRIX, AIX, SunOS, OSF1, Any
- Status: Working, Broken, Any
- Device Type: EXB-8200, DLT2000, EXB-8500, DLT4000, EXB-8505, Any, EXB-8900

At the bottom, there are input fields for "Allocated by:", "Hostname:", "Last VSN:", and "Group:", along with "Find Matches" and "Dismiss" buttons.

The menu options under **VIEW** and **ACTION** include most of the features you will need.

xtapeview

Another X interface to **OCS** which allows you to view pending mount requests is **xtapeview**. As a user, you are not permitted to respond to mount requests, only view them. Invoke this interface with the command **xtapeview**. A window will appear from which you can choose a node. Click on the node you want, then you'll see a window like the one below which shows the drives associated with that node, and any pending mounts.



If you have a color screen, the banner at the top is blue when no mounts are pending, and red if one or more are. The status button (to the left of the drive name) will be red if a mount is pending on that drive. If you don't have a color screen, there are gray-tone representations of blue and red. See `$(OCS_DIR)/doc/xtape.ps` (recall that `$(OCS_DIR)` is defined during setup) for a full description of this application, and to see how to customize the color scheme.

14.3.3 Using Provided Examples to Get Started

Examples are provided in the `$(OCS_DIR)` directory (defined during setup). You can look at `$(OCS_DIR)/doc/viewgraphs.ps` for more single-command examples and their expected output.

The directory `$(OCS_DIR)/examples` provides sample programs for FORTRAN, C and Bourne shell script which are intended to help you understand how **OCS** works. The executables are named `ocs_ctest`, `ocs_ftest`, and `ocs_btest` for C, FORTRAN and Bourne shell, respectively. Each program carries almost identical functionality. You may want to use the source of these programs (`ocs_ctest.c`, `ocs_ftest.fs` and `ocs_btest.sh`) as an aid in developing your own programs. The `$(OCS_DIR)/examples/README` file explains how to use the examples.

These example programs can actually send a mount request to operations staff: please keep this in mind if you experiment with OCS on Fermilab Computing Division systems such as FNALU.

