GA0012

Fermilab

OCS Reference Guide

Dorota Genser
Matt Fausey

Marilyn Schweitzer
George Szmuksta

8/5/99

Version 3.2

Abstract

Operator Communications Software (OCS) is a package that performs operator assisted tape mounts, tape drive allocation and tape drive tracking/management. This document describes the available commands and subroutines.

Fermi National Accelerator Laboratory
Batavia, IL 60510

Table of Contents

1.0 OCS Subroutine Interface

This section describes the library of routines (libocs.a) that are used by application programs to allocate tape drives, issue mount requests, etc. The routines are directly callable from C and the synopsis of each routine is shown in C. A Fortran interface to these routine is provided in libocs.a and is described in the last subsection.

Version v3_0 and higher is using ftt (Fermi Tape Tools[1]) as a lower level IO package.

1.1 Handling Errors

Any error messages generated by routines in libocs.a will be displayed by calling the user-supplied routine *ocs_display_error()*. This allows error messages to be integrated into any type of application, such as X-windows. All applications that call OCS routines should supply an *ocs_display_error()* routine. If the *ocs_display_error()* routine is not supplied by the user, the default routine (found in libocs.a) is used. This simply prints the error message to stdout using a printf statement.

All routines return a zero if successful, and an error code less than zero if unsuccessful. The following error codes may be returned, and are defined in the header file **ocs.h** and **ocs.inc** for FORTRAN. The errors will be described in more detail in the error message returned in the ***errmsg*** parameter for a particular routine.

1.1.1 Error Codes

OCS_ADMIN_ERROR	– The database has not been administered to handle the request. For example, an attempt to add a tape drive that already exists or an attempt of a user to make an unauthorized administrative change.
OCS_ALREADY_ALLOCATED	– An attempt was made to allocate a tape drive that was already allocated by the same user. The only function that can return this error code is *ocs_allocate()*.
OCS_ALREADY_REQUESTED	– A mount request is already pending for the specified tapedrive. The only function that can return this error code is *ocs_mount_request()*.
OCS_BAD_PREFERENCE	– A preferred tape drive has been specified with *ocs_preference()*, but this tape does not belong to the groups specified in the call to *ocs_allocate()*, or does not have the same device type as specified in the call to *ocs_allocate()*.
OCS_CALLBACK_FAILURE	– A communications error occurred while waiting for a mount reply.
OCS_COMM_FAILURE	– The client is unable to communicate with the taped daemon.
OCS_DB_ERROR	– Indicates an error in an operation (such as an insert or fetch) involving the database.

1. Fermi Tape Tools (ftt) Library User's Guide

OCS_DB_HOST_CONFLICT — The taped daemon is not using the same database as the calling process. This error can only be returned by functions which make RPC calls to the taped daemon, such as *ocs_mount_request()*, *ocs_mount_cancel()*, and *ocs_oper_msg()*.

OCS_DEV_NOT_READY — The tape drive is not ready to read or write a tape, e.g. a tape is not in the drive. This error can only be returned by *ocs_check_label()*.

OCS_DRIVE_NOT_FOUND — The specified tape drive could not be found in the database.

OCS_HOST_GROUP_NOT_FOUND — The host group could not be found in the database. The only function that can return this error code is *ocs_allocate()*.

OCS_HOST_UNAVAILABLE — The host could be reached from the network. Starting with OCS v2_2, a tape drive will not be allowed to be allocated unless contact can be established with the host where it is located.

OCS_INIT_FAILED — The first time any of the OCS routines are called, OCS initialization is automatically performed. This relieves the user of having to call some kind of initialization routine. This error code indicates that the initialization failed.

OCS_LABEL_BAD — The tape in the drive is blank or the beginning characters do not conform to those on a standard labeled tape. This error can only be returned by *ocs_check_label()*.

OCS_LABEL_WRONG — The label of the tape in the drive does match the request label. This error can only be returned by *ocs_check_label()*.

OCS_OPER_REPLY — A subsequent action could not be taken because the operator could not mount the requested tape. This error is currently only returned by *ocs_clean_it()*.

OCS_MISC_ERROR — Any error condition not covered by any of the other errors in this list.

OCS_NO_ALLOCS_PENDING — Returned by *ocs_alloc_reply()* and *ocs_alloc_cancel()* when no outstanding queued allocation requests exist.

OCS_NOT_BROKEN — Returned by *ocs_mark_working()* when the drive is in the working condition

OCS_NO_DEV_AVAILABLE — There were no available tape drives fitting the requirements specified. The only function that can return this error code is *ocs_allocate()*.

OCS_NO_DEVICE_FILE — There is no device file in the matching the requirements specified for fixed/variable length records, byte swapping, and rewind-on-close.

OCS_NO_MEMORY — An OCS subroutine failed to allocate memory for internal data structures.

OCS_NO_MOUNTS_PENDING — No tape mount requests are currently pending for the calling process. The only function that can return this error code is *ocs_mount_reply()* and *ocs_mount_cancel()*.

OCS_NOT_ALLOCATED	– The specified tape drive was not previously allocated. This error can be returned by all functions required to have drive allocated.
OCS_NOT_WORKING	– The tape drive is not in working condition. This error can be returned by *ocs_mount_request()* and *ocs_mark_broken()*.
OCS_PGID_EXIST	– Returned by *ocs_deallocate()* when process with PID = PGID of the process which allocated drive still exists and user requested checking it.
OCS_PID_EXIST	– Returned by ocs_deallocate() when process which allocated drive still exists
OCS_PERMISSION_DENIED	– The calling process does not have permission to do something. This can happen if a process attempts to cancel a mount request that was not originally issued by a process with the same uid and login name.
OCS_PROTECT_WRONG	– The tape in the tape drive does not have the appropriate write protection. This error can only be returned by *ocs_check_label()*.
OCS_RPC_ERROR	– An error occurred when making an RPC call to the taped daemon.
OCS_STAT_ERROR	– An error occurred initializing or collecting a device file's use statistics in a call to *ocs_init_stat()*, *ocs_report_stat()*, *ocs_init_ftt()* or *ocs_report_ftt()*.
OCS_SYSTEM_ERROR	– An operating system error occurred.
OCS_TIMEOUT	– This value is returned when the time limit expires in a call to *ocs_mount_reply()* or *ocs_alloc_reply()*.
OCS_WRONG_PID	– this value is returned by *ocs_deallocate()* when the PID requested by user to compare with PID of the process which allocated this drive don't agree.

1.1.2 ocs_display_error()

The routine is used to display error messages. The user may either write his/her own version in C or use the version supplied by **libocs.a**.

Usage:

ocs_display_error(*errmsg***)**

Input Parameters:

char ****errmsg*** – Error message to display

If supplied by the user, any errors generated by routines in **libocs.a** will call *ocs_display_error()* with a formatted error message for the application to display. This allows each application to handle error messages in a uniform way. For example, an X-windows application might want to pop up a dialog displaying the error message.

If the *ocs_display_error()* routine is not supplied by the user, the default routine (found in **libocs.a**) is used. This simply prints the error message to stdout using a printf statement

1.2 Allocate Tape Drive

1.2.1 ocs_allocate()

This routine attempts to allocate a tape drive that meets the requirements for the host group, device group and device type specified by the user. If a tape drive is successfully allocated, it will return device name and host name of the allocated drive, and the function will return a value of zero. (See "ocs_preference()" on page 12. for information how to allocate specific tape drives.)

Usage:

retval **= ocs_allocate (***hostgroup***,** *devgroup***,** *devtype***,** *outhost***,** *outdev***,** *errmsg***)**

Input Parameters:

char ****hostgroup***	– Host group name or "localhost" or "any"='\0'=NULL
char ****devgroup***	– Device group name or "any"
char ****devtype***	– Device type or "any"

Output Parameters:

char ****outhost***	– Host name of the allocated drive
char ****outdev***	– Name of the allocated drive
char ****errmsg***	– Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval***	– 0 for success and negative value for error

The ***devtype*** argument specifies the type of tape drive required. The following is the list of allowed device types:

- EXB-8200
- EXB-8500
- EXB-8505
- EXB-8900
- DLT2000
- DLT4000

And generic names:

- EXB
- EXB-850X

If the generic device type 'EXB' is used, the OCS system attempts to allocate an EXB-8200. If none is available, then OCS attempts to allocate an EXB-8500 that can be run in EXB-8200 mode. If no EXB-8500 is available, then OCS attempts to allocate an EXB-8505 which can also be run in EXB-8200 mode.

If the generic parameter 'EXB-850x' is used, the OCS system will attempt to allocate an EXB-8500. If none is available, then OCS will attempt to allocate an EXB-8505.

If device type is "any", the device type will be totally ignored when the system selects a tape drive to allocate.

If ***hostgroup*** is the name of a group, then all tape drives on any of the hosts in the group are available, as long as they fit the remaining requirements. If the special string "localhost" is used, then only tape drives located on the host of the calling process are allowed to be selected.

The ***devgroup*** can be "any" or the name of a device group. If string "any" is used this parameter is ignored.

It is intentionally designed so that it is not possible to hard-code a specific host's name in a call to *ocs_allocate*(). For example, instead of specifically requesting "quark.fnal.gov", the user might instead request "johns_host", where "johns_host" is a group containing one host, namely "quark.fnal.gov". This makes it possible for a system manager to change the host allocated to John by changing the mapping between the group name and the host. The same argument holds for groups that contain more than one host.

If the string "any" is used as a ***hostgroup*** it means that this parameter is ignored.

Following is list of the errors specific to this function:

OCS_ALREADY_ALLOCATED	– the tape drive set to be allocated as a only one wanted by the user using *ocs_preference()* function is allocated to the user.
OCS_BAD_PREFERENCE	– the tape drive set to be allocated as a only one wanted by the user using *ocs_preference()* function doesn't match host group, device group or device type specified by the user.
OCS_NO_DEVICE_AVAILABLE	– no drive can be allocated
OCS_HOST_GROUP_NOT_FOUND	– The host group doesn't exist in the OCS database
OCS_HOST_UNAVAILABLE	– The requested host isn't reachable

1.2.2 ocs_no_preference()

It cancels the preference that was previously set with *ocs_preference()*.

Usage:

ocs_no_preference()

1.2.3 ocs_preference()

This routine sets a preference for a specific tapedrive for the next call to *ocs_allocate()*.

Usage:

retval = **ocs_preference** (*hostname*, *tapedrive*, *must_get*, *errmsg*)

Input Parameters:

char *	***hostname***	– Host name or "localhost" or "any"='\0'=NULL
char *	***tapedrive***	– Tape drive name
int	***must_get***	– Flag (0 or 1) if the user wants only this drive to be allocated

Output Parameters:

char *	***errmsg***	– Error message set if the error occurs. (It has to be at least 256 characters).

Return value:

int	***retval***	– 0 if success and negative value if error

The tape drive specified by the user using this function can be allocated by *ocs_allocate()* call only if it meets all requirements requested by the user in *ocs_allocate()*.

If ***must_get*** is a nonzero value, then *ocs_allocate()* will only succeed if the tape drive specified in this call can be allocated.

If ***must_get*** is set to zero, then *ocs_allocate()* may return with another tape drive if the preferred one is already allocated or if it doesn't match host group, device group or device type specified by the user in the *ocs_allocate()*.

1.3 Queue Allocate Tape Drive

1.3.1 ocs_queue_alloc()

This routine issues a request to allocate a tape drive. If no tape drives are available because they are either allocated or broken, the allocation request is queued. When a tape drive which meets the specified requirements becomes available, the OCS system allocates the tape drive on behalf of the calling process, and send a reply which can be fetched by calling *ocs_alloc_reply()*.

Usage:

bid_Id = **ocs_queue_alloc**(*hostgroup*, *devgroup*, *devtype*, *errmsg*)

Input Parameters:

char *	***hostgroup***	– Host group name or "localhost" or "any"='\0'=NULL

char **devgroup*** – Device group name or “any”

char **devtype*** – Device type or “any”

Output Parameters:

char **errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***bid_Id*** – positive value for success and negative value for error

For each successful call to this routine, there must be a matching call to *ocs_alloc_reply()* or *ocs_alloc_cancel()*. In this case, *ocs_queue_alloc()* returns a ***bid_Id*** which is an integer grater then zero. The ***bid_Id*** is returned in subsequent *ocs_alloc_reply()* calls to able to match a reply with each user's request. For example, if three successive calls are made to this routine, then there are three outstanding queued allocation requests. The first call to *ocs_alloc_reply()* can return a reply from any of the outstanding requests.

The *ocs_allocate()* (Section 1.2.1 on page 10) routine describes how ***hostgroup***, ***devgroup***, and ***devtype*** can be used to specify requirements for selecting a tape drive. This routine also works in conjunction with *ocs_preference()* and *ocs_no_preference()*.

1.3.2 ocs_alloc_reply()

This routine attempts to fetch a reply from any outstanding queued allocation requests.

The calling user must have the same UID as the user which queued request.

Usage:

retval **= ocs_alloc_reply (***bid_id***,** *outhost, outdev***,** *errmsg***,** *timeout***)**

Input Parameters:

int ***timeout*** – maximum number of seconds this call waits for drive to be allocated before it returns. If it is set to -1, the routine waits indefinitely until a reply is obtained. If it set to 0 it causes a poll for a reply.

Output Parameters:

int ***bid_Id*** – value to match allocate queue request issued by the *ocs_queue_alloc()*

char **outhost*** – Host name of the allocated drive

char **outdev*** – Name of the allocated drive

char **errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success, OCS_TIMOUT or negative value for error.

If the OCS_TIMEOUT is returned the caller can either wait for the reply again by calling *ocs_alloc_reply()* or cancel the allocation request by calling *ocs_alloc_cancel()*.

List of errors specific to this function:

OCS_NO_ALLOCS_PENDING	– there is now allocate request pending

1.3.3 ocs_alloc_cancel()

This routine cancels a queued allocation request.

The calling process must have the same UID as the process which queued request.

Usage:

retval = **ocs_alloc_cancel** (*bid_Id*, *errmsg*)

Input Parameters:

int	***bid_Id***	– number to match allocate request issued by *ocs_queue_alloc()* call

Output Parameters:

char *	***errmsg***	– Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int	***retval***	– 0 for success and negative value for error.

Because the allocation request can be satisfied after *ocs_queue_alloc()* is called but before *ocs_alloc_cancel()*, it is not considered an error if ***bid_Id*** does not exist for the calling user. In this case, ***bid_Id*** along with the tape drive that was allocated will be returned in a subsequent *ocs_alloc_reply()* call; such a tape drive should be deallocated.

1.4 Mount/Dismount Tape

1.4.1 ocs_mount_request()

This routine issues a tape mount request to the system. The routine returns immediately after sending the mount request and does not wait for a reply. The return value of this routine indicates the success or failure of sending the mount request. The reply can be obtained by calling *ocs_mount_reply()* (Section 1.4.2 on page 15).

Tape drive has to be allocated to the user.

There can be only one request for a tape drive

Usage:

retval **= ocs_mount_request(***host, tapedrive***,** *tapeset***,** *vsn***,** *protection***,** *opmsg***,** *errmsg***)**

Input Parameters:

char ****host***	**–** Host name or “localhost” or “any”=‘\0’=NULL
char ****tapedrive***	**–** Tape drive name
char ****tapeset***	**–** Name of the tape set
char ****vsn***	**–** VSN of the requested tape
char ****protection***	**–** “r” /(“w”) for the tape to read only access / (read and write access)
char ****opmsg***	**–** message to the operator or ‘\0’=NULL

Output Parameters:

char ****errmsg***	**–** Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval***	**–** 0 for success and negative value for error

The tape to be mounted is specified by ***tapeset*** and its volume serial number, ***vsn***. If the tape is not part of a set of tapes, then ***tapeset*** can be passed as a NULL pointer or a NULL character string.

Optionally, a special message (***opmsg***) can be sent with the mount request that will be displayed with the mount request on the operator’s console. If no special message is to be sent, which should be most of the time, then ***opmsg*** should be passed as a NULL pointer or a NULL character string.

The OCS system periodically checks to see if processes with pending mount requests are still in existence. If the process requesting a tape mount exits before receiving a reply, the mount request is cancelled automatically. The tape drive however, will remain allocated.

List of errors specific to this function:

OCS_NOT_WORKING	**–** Given tape drive doesn’t have “working” condition
OCS_ALREADY_REQUESTED	**–** Given tape drive has mount request pending

1.4.2 ocs_mount_reply()

This routine can be used to poll or wait for a tape mount reply.

The calling process must have the same PID as the process which queued request.

Usage:

retval **= ocs_mount_reply (***hostname***,** *tapedrive***,** *status***,** *errmsg***,** *timeout***)**

Input Parameters:

int ***timeout*** – maximum number of seconds this call waits for tape to be mounted before it returns. If it is set to -1, the routine will wait indefinitely until a reply is obtained. If it set to 0 it causes a poll for a reply.

Output Parameters:

char ****hostname*** – Host name

char ****tapedrive*** – Tape drive name

int ****status*** – Status of the mount (0 for success, 1 if the mount was canceled, -1 if there was mount error)

char ****errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success, OCS_TIMOUT or negative value for error.

If a timeout occurs, a positive value, OCS_TIMEOUT, is returned. At this point, the caller can either wait for the reply again by calling *ocs_mount_reply()* or cancel the mount request by calling *ocs_mount_cancel()*. If the mount request is canceled, a final call to *ocs_mount_reply()* must be made to fetch the reply.

This routine should be called at least once for each mount request. More specifically, if three tape mounts are requested by calling *ocs_mount_request*() three times, then *ocs_mount_reply*() should be called at least three times, to catch all replies. The replies can arrive in any order.

When *ocs_mount_reply*() returns, the name of the tape drive and its host are returned in ***tapedrive*** and ***hostname*** respectively. The success of the tape mount is returned in ***status***. A ***status*** value of zero indicates a successful tape mount, while a -1 indicates failure. The reason the tape mount failed will be returned in ***errmsg***. The ***status*** value of 1 indicates that the mount has been explicitly cancelled (by the same process).

The return value of this function indicates whether or not a reply was successfully obtained. If the return value is not zero the parameters ***hostname***, ***tapedrive***, and ***status*** are not set.

List of errors specific to this function:

OCS_NO_MOUNT_PENDING – there is no mount request pending

1.4.3 ocs_mount_cancel()

This function cancels a mount request.

> *The calling process must have the same PID as the process which queued request.*

Usage:

retval = **ocs_mount_cancel**(*hostname*, *tapedrive*, *errmsg*)

Input Parameters:

char ****hostname***	– Host name or “localhost” or “any”=‘\0’=NULL
char ****tapedrive***	– Tape drive name

Output Parameters:

char ****errmsg***	– Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval***	– 0 for success and negative value for error

The mount request to cancel is specified by the ***hostname*** and ***tapedrive*** parameters.

If the same process that requested the mount subsequently cancels it, a mount reply still **must** be fetched by calling *ocs_mount_reply()* after the mount request has been cancelled. In most cases, the status returned from the *ocs_mount_reply()* will indicate the ‘mount cancelled’ condition. In some cases, however, if the operator responds just as the cancellation is issued, then *ocs_mount_reply()* may contain a status indicating a successful or unsuccessful mount. This, of course, can be ignored.

List of errors specific to this function:

OCS_NO_MOUNT_PENDING	– there is no mount request pending

1.4.4 ocs_mounted()

This function updates all information about the mounted tape in the OCS database for tape which is mounted by the user. It imitates *ocs_mount_request()* call with successful reply but without any request sent.

The drive has to be allocated to the user

Usage:

retval = **ocs_mounted** (*hostname*, *tapedrive*, *tapeset*, *vsn*, *protection*, *errmsg*)

Input Parameters:

char ****hostname***	– Host name or “localhost” or “any”=‘\0’=NULL
char ****tapedrive***	– Tape drive name
char ****tapeset***	– Name of the tape set
char ****vsn***	– VSN of the requested tape
char ****protection***	– “r” /(“w”) for the tape to read only access / (read and write access)

Output Parameters:

char ****errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

1.4.5 ocs_check_label()

This function checks whether the tape in the drive specified by ***hostname*** and ***tapedrive*** has the corrected label type, label and write protection.

The drive has to be allocated to the user

Usage:

retval **= ocs_check_label (***hostname***,** *tapedrive***,** *label_type***,** *vsn***,** *protection***,** *errmsg***)**

Input Parameters:

char ****hostname*** – Host name or “localhost” or “any”=‘\0’=NULL

char ****tapedrive*** – Tape drive name

char ****label_type*** – Type of the label to be checked (‘\0’, “ansi”, “fmb”, “tar”, “cpio”, “blank”)

char ****vsn*** – VSN to check

char ****protection*** – “r” /(“w”) to check if tape is/(isn’t) write protected

Output Parameters:

char ****errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

This function checks whether the tape in the drive specified by ***hostname*** and ***tapedrive*** has the corrected label type, volume label and write protection.

If the ***vsn*** and ***label_type*** parameters are ‘\0’ only the write protection is checked.

If the ***label_type*** is ‘\0’ the label is assumed to be “ansi”.

If ***hostname*** is the string “localhost”, then the logical tape drive name, ***tapedrive***, is assumed to be on the local host. If the hostname parameter is NULL or ‘\0’ or “any” given ***tapedrive*** has to be unique in the database.

List of errors specific to this function in order how they are checked:

OCS_DEV_BUSY. – the device is used by other process

OCS_DEV_NOT_READY. – the tape is not present in the drive

OCS_LABEL_BAD. – given ***label_type*** doesn't match tape label

OCS_LABEL_WRONG. – given ***vsn*** doesn't match tape label but the label type is correct

OCS_PROTECT_WRONG. – if the protection doesn't match with one asked by user. It is returned after the label type and VSN is checked unless they are both set to '\0'.

1.4.6 ocs_check_tape()

This function was replaced by ocs_check_label(). It will became obsolete in future releases.

This function as well as its FORTRAN equivalence calls *ocs_check_label()* function (see Section 1.4.5 on page 18) with ***label_type*** parameter set to '\0'.

The drive has to be allocated to the user

Usage:

retval **= ocs_check_tape (***hostname***,** *tapedrive***,** *devfile***,** *protection***,** *vsn***,** *errmsg***)**

Input Parameters:

char ****hostname***	– Host name or "localhost" or "any"='\0'=NULL
char ****tapedrive***	– Tape drive name
char ****devfile***	– This parameter is IGNORED
char ****vsn***	– VSN to check
char ****protection***	– "r" /("w") to check if tape is/(isn't) write protected

Output Parameters:

char ****errmsg***	– Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval***	– 0 for success and negative value for an error

1.4.7 ocs_use_sigio()

This cause a SIGIO signal to be generated whenever a mount reply arrives. This function should be called prior to all other OCS function calls. It is the programmer's responsibility to establish a signal handler for the SIGIO signal. The signal handler should call *ocs_mount_reply()* to handle the reply.

Usage:

ocs_use_sigio()

1.4.8 ocs_dismount_request()

The following three routines provide interface to operator-assisted dismount feature introduced in OCS v3.2. This feature may be disabled by operator so that dismount requests are ignored and/or immediately replied to with success status without actually performing the requested dismount. Before using this feature, users should check with system administrator or operations to make sure that this feature is enabled.

This routine issues a tape dismount request to the system. The routine returns immediately after sending the dismount request and does not wait for a reply. The return value of this routine indicates the success or failure of sending the dismount request. The reply can be obtained by calling *ocs_dismount_reply()* (Section 1.4.2 on page 15).

Tape drive has to be allocated to the user.

There can be only one request for a tape drive

Usage:

retval = **ocs_dismount_request(***host, tapedrive***,** *tapeset***,** *vsn***,** *opmsg***,** *errmsg***)**

Input Parameters:

char ****host***	– Host name or “localhost” or “any”=‘\0’=NULL
char ****tapedrive***	– Tape drive name
char ****tapeset***	– Name of the tape set
char ****vsn***	– VSN of the requested tape
char ****opmsg***	– message to the operator or ‘\0’=NULL

Output Parameters:

char ****errmsg***	– Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval***	– 0 for success and negative value for error

The tape to be dismounted is specified by ***tapeset*** and its volume serial number, ***vsn***. If the tape is not part of a set of tapes, then ***tapeset*** can be passed as a NULL pointer or a NULL character string.

Optionally, a special message (***opmsg***) can be sent with the dismount request that will be displayed with the dismount request on the operator’s console. If no special message is to be sent, which should be most of the time, then ***opmsg*** should be passed as a NULL pointer or a NULL character string.

The OCS system periodically checks to see if processes with pending dismount requests are still in existence. If the process requesting a tape dismount exits before receiving a reply, the dismount request is cancelled automatically. The tape drive however, will remain allocated.

List of errors specific to this function:

OCS_NOT_WORKING	– Given tape drive doesn't have "working" condition
OCS_ALREADY_REQUESTED	– Given tape drive has dismount request pending

1.4.9 ocs_dismount_reply()

This routine can be used to poll or wait for a tape dismount reply.

The calling process must have the same PID as the process which queued request.

Usage:

retval = **ocs_dismount_reply** (*hostname*, *tapedrive*, *status*, *errmsg*, *timeout*)

Input Parameters:

int	***timeout***	– maximum number of seconds this call waits for tape to be dismounted before it returns. If it is set to -1, the routine will wait indefinitely until a reply is obtained. If it set to 0 it causes a poll for a reply.

Output Parameters:

char *	***hostname***	– Host name
char *	***tapedrive***	– Tape drive name
int	****status***	– Status of the dismount (0 for success, 1 if the dismount was canceled, -1 if there was dismount error)
char *	***errmsg***	– Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int	***retval***	– 0 for success, OCS_TIMOUT or negative value for error.

If a timeout occurs, a positive value, OCS_TIMEOUT, is returned. At this point, the caller can either wait for the reply again by calling *ocs_dismount_reply()* or cancel the dismount request by calling *ocs_dismount_cancel()*. If the dismount request is canceled, a final call to *ocs_dismount_reply()* must be made to fetch the reply.

This routine should be called at least once for each dismount request. More specifically, if three tape dismounts are requested by calling *ocs_dismount_request*() three times, then *ocs_dismount_reply*() should be called at least three times, to catch all replies. The replies can arrive in any order.

When *ocs_dismount_reply*() returns, the name of the tape drive and its host are returned in ***tapedrive*** and ***hostname*** respectively. The success of the tape dismount is returned in ***status***. A ***status*** value of zero indicates a successful tape dismount, while a -1 indicates failure. The reason the tape dismount failed will be returned in ***errmsg***. The ***status*** value of 1 indicates that the dismount has been explicitly cancelled (by the same process).

The return value of this function indicates whether or not a reply was successfully obtained. If the return value is not zero the parameters ***hostname***, ***tapedrive***, and ***status*** are not set.

List of errors specific to this function:

OCS_NO_MOUNT_PENDING	**–** there is no mount request pending

1.4.10 ocs_dismount_cancel()

This function cancels a dismount request.

> *The calling process must have the same PID as the process which queued request.*

Usage:

retval **= ocs_dismount_cancel(***hostname***,** *tapedrive***,** *errmsg***)**

Input Parameters:

char ****hostname***	**–** Host name or "localhost" or "any"='\0'=NULL
char ****tapedrive***	**–** Tape drive name

Output Parameters:

char ****errmsg***	**–** Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval***	**–** 0 for success and negative value for error

The dismount request to cancel is specified by the ***hostname*** and ***tapedrive*** parameters.

If the same process that requested the dismount subsequently cancels it, a dismount reply still **must** be fetched by calling *ocs_dismount_reply()* after the dismount request has been cancelled. In most cases, the status returned from the *ocs_dismount_reply()* will indicate the 'dismount cancelled' condition. In some cases, however, if the operator responds just as the cancellation is issued, then *ocs_dismount_reply()* may contain a status indicating a successful or unsuccessful dismount. This, of course, can be ignored.

List of errors specific to this function:

OCS_NO_MOUNT_PENDING	**–** there is no dismount request pending

1.5 Device Files

1.5.1 ocs_devfile_default()

This function resets all the device file attributes to their default values (see function *ocs_devfile_set()* Section 1.5.3 on page 23 to see all possible attributes).

Usage:

ocs_devfile_default()

1.5.2 ocs_devfile_get()

This function fetches the device file for the tape drive specified by ***hostname*** and ***tapedrive*** corresponding to the attributes set with previous calls to *ocs_devfile_set()*.

Usage:

retval **= ocs_devfile_get (***hostname***,** *tapedrive***,** *filename***,** *errmsg***)**

Input Parameters:

char ****hostname***	– Host name or “localhost” or “any”=’\0’=NULL
char ****tapedrive***	– Tape drive name

Output Parameters:

char ****filename***	– Device filename
char ****errmsg***	– Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval***	– 0 for success and negative value for an error

List of errors specific to this function:

OCS_NO_DEVICE_FILE	– there is no device file for parameters set using *ocs_devfile_set()* function for this tape drive.

1.5.3 ocs_devfile_set()

This function will set one or more local device file attributes for a subsequent call to *ocs_devfile_get()*.

Usage:

retval **= ocs_devfile_set (***attrib***,** *errmsg***)**

Input Parameters:

char **attrib* – List of the attributes to set

Output Parameters:

char **errmsg* – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

The attributes are specified in ***attrib***, and more than one attribute can be specified by separating them with colons. For example, "8500:compression:variable" would set the device for 8500 mode, data compression and variable blocked records. Multiple attributes can also be set by calling this function more than once, each time with a different attribute. The following attribute strings are recognized:

Attribute Function	***attrib*** string	**Comments**
variable block size	variable	This is the default
fixed block size	fixed	
data compression	compression	
no data compression	no_compression	This is the default
no byte swapping	no_swap	This is the default, but since byte swapping is not supported via FTT, there is no need to specify this as an ***attrib*** string.
no retension on open	no_retension	This is the default, but since byte swapping is not supported via FTT, there is no need to specify this as an ***attrib*** string.
no rewind on close	no_rewind	This is the default, but since byte swapping is not supported via FTT, there is no need to specify this as an ***attrib*** string.
tape density or mode	8200 8500 10G 20G ...	The default will be the highest density that the device supports, e.g. 8500 mode on an EXB-8500 device. Not all possible density strings for DLTs are listed; any of the ones supported by the device are allowed.

If an unrecognizable string is passed in attrib, the function will return negative value and the unrecognizable attribute will be returned in ***errmsg***. The function returns 0 if successful.

If an attribute does not apply to a particular type of device, it can be ignored.

1.5.4 ocs_device_file()

This function is removed in this release of OCS. The functions ocs_devfile_default(), ocs_devfile_set(), and ocs_devfile_get() should be used instead.

1.5.5 ocs_device_type()

This function returns the device type of the specified tape drive.

Usage:

retval **= ocs_device_type (***hostname***,** *tapedrive***,** *devtype***,** *errmsg***)**

Input Parameters:

char *	***hostname***	– Host name or "localhost" or "any"='\0'=NULL
char *	***tapedrive***	– Tape drive name

Output Parameters:

char *	***devtype***	– Device type
char *	***errmsg***	– Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int	***retval***	– 0 for success and negative value for an error

See *ocs_allocate()* function Section 1.2.1 on page 10 for list of valid device types.

1.5.6 ocs_logdev()

This function returns name of the tapedrive corresponding to given device file name and host name.

Usage:

retval **= ocs_logdev (***devfile***,** *hostname***,** *tapedrive***,** *option***,** *errmsg***)**

Input Parameters:

char *	***devfile***	– device filename
char *	***option***	– Host name or "localhost"

Output Parameters:

char *	***tapedrive***	– Tape drive name
char *	***hostname***	– Host name
char *	***errmsg***	– Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int	***retval***	– 0 for success and negative value for an error

If ***option*** is “localhost”, this function returns the logical tape drive name in ***tapedrive*** and the local host name in ***hostname*** corresponding to the specified device file, ***devfile***, on the local host.

The symbolic link to the device file can not be used in current release.

1.5.7 ocs_logical_dev()

This function was removed from this release of OCS. The function ocs_logdev() should be used instead.

1.5.8 ocs_setdev()

This command will ensure that the device file given by “ocs_devfile_get()” on page 23 will be physically altered to match the desired attributes, e.g. variable block size.

The drive has to be allocated to the user

Usage:

retval = **ocs_setdev** (*hostname*, *tapedrive*, *block_size*, *errmsg*)

Input Parameters:

char *	***hostname***	– Host name or “localhost” or “any”=‘\0’=NULL
char *	***tapedrive***	– Tape drive name
int	***block_size***	– block size

Output Parameters:

char *	***errmsg***	– Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int	***retval***	– 0 for success and negative value for an error

The process calling *ocs_setdev()* may be located on the host where the tape drive is located, or it may be on a remote host.

The tape drive is specified by ***hostname*** and ***tapedrive***. The device characteristics are specified by making function calls to *ocs_devfile_default()* (Section 1.5.1 on page 23) and *ocs_devfile_set()* (Section 1.5.3 on page 23) before calling this function. Currently the only characteristics that can be set is the fixed/variable block size and data compression. If fixed block size is selected, the block size needs to be correctly passed in ***block_size***.

1.5.9 ocs_setown()

This function ensures that the ownership and protection mode of the device files are correctly set. It should be called after a tape drive has been allocated, by the process that is going to read/write the tape.

The drive has to be allocated to the user

Usage:

retval **= ocs_setown (***hostname***,** *tapedrive***,** *errmsg***)**

Input Parameters:

char *	***hostname***	– Host name or "localhost" or "any"='\0'=NULL
char	***tapedrive***	– Tape drive name

Output Parameters:

char *	***errmsg***	– Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int	***retval***	– 0 for success and negative value for an error

In most cases, the ownership and protection of the device files is set when the tape drive is allocated. In some circumstances however, this may not be possible. For example, if the allocated tape drive is on a remote node the following can occur:

1. The remote node is inaccessible from the network, or is powered off during the allocation.
2. The OCS daemons are not running on the remote node at the time of the allocation.

In both cases, it is ok for the allocation to succeed because the problem may be resolved by the time the process intending to use the tape drive is created.

If the tape drive allocated with *ocs_allocate()* (Section 1.2.1 on page 10) or *ocs_queue_alloc()* (Section 1.3.1 on page 12) is on the local host, there is no need to call this function. The allocate fails if the device files cannot be set correctly.

1.6 Device Statistics

1.6.1 ocs_init_stat()

This function retrieves the current tape drive use statistics from the device driver and temporarily stores them in the OCS database in anticipation of a subsequent call to *ocs_report_stat()*.

The tape drive has to be allocated to the user.

Usage:

retval = **ocs_init_stat** (*hostname*, *tapedrive*, *errmsg*)

Input Parameters:

char ****hostname*** – Host name or “localhost” or “any”=‘\0’=NULL

char ****tapedrive*** – Tape drive name

Output Parameters:

char ****errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for an error

This function should be called immediately before opening a file for reading or writing. After the file is closed, *ocs_report_stat()* should be called to collect the tape drive use statistics from the device again, calculate the net changes, and to save the statistics in the OCS database.

If the getting drive statistics fails the function returns success but it prints message to the standard output. In such case when the report is stored the valid flag (see “The meaning of the valid flag:” on page 32) contains “I” character to see why not all the information is available.

List of errors specific to this function:

OCS_STAT_ERROR – is returned when statistic was already initialized

1.6.2 ocs_report_stat()

This function collects tape drive use statistics from the device driver, creates an OCS database tape drive use report record for that tape drive and updates the OCS summary statistics for the tape drive.

The tape drive has to be allocated by the user.

Usage:

retval = **ocs_report_stat** (*hostname*, *tapedrive*, *op*, *kb_usr*, *display*, *errmsg*)

Input Parameters:

char ****hostname*** – Host name or “localhost” or “any”=‘\0’=NULL

char ****tapedrive*** – Tape drive name

char ****op*** – Operation (“r” for read only and “w” for read and write)

int ***kb_user*** – Kbytes read or written calculated by the user char **display* – "yes" if user want to have statistic summary printed

Output Parameters:

char **errmsg* – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

Function *ocs_init_stat()* should be called before reading or writing to initialize the statistics and *ocs_report_stat()* should be called immediately after closing a file. The tape drive use statistics is not correctly calculated if the tape is rewound before calling this function. Thus, device drivers that rewind automatically upon closing a file should not be used in conjunction with *ocs_report_stat()*.

If the getting drive statistics fails the report is made anyway. Because not all of the information is available the message is printed to the standard output and "valid" flag (see "The meaning of the valid flag:" on page 32) is updated.

For reports which are not initialized using *ocs_init_stat()* function all available information is stored anyway but the message is printed to the standard output and the "valid" flag is updated.

List of errors specific to the function:

OCS_STAT_ERROR	– is returned when the statistics are initialized using *ocs_init_ftt()* function instead of *ocs_init_stat()*

1.6.3 ocs_init_ftt()

This function retrieves the current tape drive use statistics from the device driver and temporarily stores them in the OCS database in anticipation of a subsequent call to *ocs_report_ftt()*. It works the same way as *ocs_init_stat()* function except it reads drive statistics using ***ftt*** calls directly on the device opened by the user.

The tape drive has to be allocated to the user and it has to be on the local host.

*This function requires linking user code with **ftt** libraries.*

Usage:

retval **= ocs_init_ftt (***fd***,** *errmsg***)**

Input Parameters:

ftt_desciptor ***fd*** – Ftt handle to the opened device

Output Parameters:

char **errmsg* – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

This function should be called after device file is opened and immediately before reading or writing.

List of errors specific to this function:

OCS_STAT_ERROR – returned when statistic was already initialized

1.6.4 ocs_report_ftt()

This function collects tape drive use statistics from the device driver, creates an OCS database tape drive use report record for that tape drive and updates the OCS summary statistics for the tape drive. It works the same way as *ocs_report_stat()* function except it reads drive statistics using ftt calls directly on the device opened by the user.

The tape drive has to be allocated by the user and it has to be on the local host.

*This function requires linking user code with **ftt** libraries.*

Usage:

retval **= ocs_report_ftt (***fd***,** *display***,** *errmsg***)**

Input Parameters:

ftt_descriptor * ***fd*** – ftt handle to opened device

char ****display*** – "yes" if user want to have statistic summary printed

Output Parameters:

char ****errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

Function *ocs_init_ftt()* should be called before *ocs_report_ftt()* to initialize the statistics and *ocs_report_ftt()* should be called immediately before closing a file.

If the getting drive statistics fails the report is made anyway. Because not all of the information is available the message is printed to the standard output and "valid" flag (see "The meaning of the valid flag:" on page 32) is updated.

For reports which are not initialized using *ocs_init_ftt()* function all available information is stored anyway but the message is printed to the standard output and the "valid" flag is updated.

List of errors specific to the function:

OCS_STAT_ERROR – is returned when the statistics were initialized using *ocs_init_stat()* function instead of *ocs_init_ftt()*

1.6.5 ocs_list_reports()

This function lists information collected using *ocs_report_stat()* and *ocs_report_ftt()* functions.

retval = **ocs_list_reports** (*host*, *dev*, *serial*, *vsn*, *bdate*, *adate*, *filename*, *opt*, *errmsg*)

Input Parameters:

- char ***host*** – Host name or “localhost” or “any”
- char ***dev*** – Device name or “any”
- char ***vsn*** – VSN or “any”
- char ***bdate*** – Date like “mm/dd/yy” of the oldest reports to list or “any”
- char ***adate*** – Date like “mm/dd/yy” of the newest reports to list or “any”
- char ***filename*** – Filename of the file to print listed reports or ‘\0’=NULL to print to STDOUT
- char ***opt*** – if it is “delete” all listed reports are deleted (only for administrators) or “short” for shorter listing

Output Parameters:

- char ***errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

- int ***retval*** – 0 for success and negative value for error

The statistics that are listed consist of (in the order they are listed):

- **Host** – Host name
- **Device** – Device name
- **Serial** – Serial number
- **Reported** – Date the report was made
- **IO** – “r” /”w” reported by the user
- **User** – User name
- **VSN** – VSN of last tape mounted before report was made
- **KB/User** – Kbytes reported by user
- **KB/dev** – Kbytes reported by device
- **Errors** – Number of errors reported by device
- **S** – Information of the way the Kbytes by device and errors were calculated by ***ftt***
- **Dens** – Density
- **C/R** – Compress ratio
- **C** – “C” if the compression was on; “.” if not

Ret – *ftt* retries
H/E – *ftt* counter of the read/write hardware errors
T/R – Track retries
S/S – Stop/Start counter
Flags – Flags string
Val – Valid flags

The information which is not available is printed as -1 or '-' and "." means that the flag is "off".

The meaning of the flags string:

T – Tape not present
r/w – Write protection is ON/OFF
C – Clean bit is on
D – Cleaned bit is on
B – Beginning of tape
E – End of tape
F – Power Failure
R – Device is not ready
S – *ftt* SCSI reset flag

The meaning of the valid flag:

I – getting initial device statistics failed
R – getting final device statistics failed
U – User didn't initialize statistics

1.6.6 ocs_list_stats()

This function lists all accumulated statistic information of given tape drive.

Usage:

retval = **ocs_list_stats (***host***,** *dev***,** *filename***,** *errmsg***)**

Input Parameters:

char ****host*** – Host name or "localhost" or "any"
char ****dev*** – Device name or "any"
char ****filename*** – Filename of the file to print listed reports or NULL='\0' to print to STDOUT

Output Parameters:

char ****errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

The statistics consist of:

- Device Type
- Device filename
- Product Id
- Vendor Id
- Serial number
- Controller
- Firmware
- Current number of power hours
- Current number of motion hours
- Installation date
- Date the drive was last cleaned
- Number of times the drive has been cleaned since installation
- Cleaning bit to see if the drive needs to be cleaned
- Since the drive was installed:
  - Number of mounts (read, write and total)
  - Number of reports received (read, write and total)
  - MBytes reported by user (read, write and total)
  - MBytes reported by device (read write and total)
  - Errors reported by device (read, write and total)
  - Power hours
  - Motion hours
  - Number of track retries
  - Number of Stop/Starts
- Since the drive was cleaned last time (the same as above)

1.6.7 ocs_stat_from_ftt()

This function extracts information form two FTT statistics buffers and fills out ocsi_db_stats structure to be passed to *ocs_update_drive_stats_from_ftt()* (see below).

Usage:

ocs_stat_from_ftt (*ftt_d, ftt_f, ocs_buf***)**

Input Parameters:

ftt_stat_buf ***ftt_d*** – Statistics accumulated over the time the drive was in use in terms of FTT deltas

ftt_stat_buf ***ftt_f*** – Statistics obtained from FTT *after* using the drive

Output Parameters:

ocsi_db_stats * ***ocs_buf***

ocsi_db_stats structure is defined in ocs.h and its definition is:

```
typedef struct {
      int power_h;            - final number of hours the drive
                              was powered on
      int motion_h;           - final number of hours the drive
                              was in motion
      short clean_bit;        - drive needs to be cleaned at the end of
                              use
      int retries;            - final number of re-tries
      int fail_retries;       - final number of failed re-tries
      char count_origin[30];  - count origin
      int compress;           - compression ratio
      char mode[30];          - density mode string
      int flags;              - flags (see above)
      int r_comp;             - read compression ratio
      int w_comp;             - write compression ratio
      int r_count;            - (δ) KBytes read as reported by FTT
      int w_count;            - (δ) KBytes written as reported by FTT
      int r_error;            - (δ) read errors
      int w_error;            - (δ) write errors
      int track_retry;        - (δ) number of track retries
      int underrun;           - (δ) number of underruns
      int user_read;          - (δ) KBytes read as reported by user
      int user_write;         - (δ) KBytes written as reported by user
} ocsi_db_stats;
```

Fields marked with (δ) are integers representing quantities accumulated only over the time when the drive was in use. Other fields are final values by the time the drive was about to be dismounted and deallocated.

1.6.8 ocsi_update_drive_stats()

This function updates tape drive usage statistics using information stored in *ocsi_db_stats* structure. *ocsi_db_stats* can be filled out using *ocs_stat_from_ftt* function (see above).

Usage:

retval **= ocs_update_drive_stats(***h,d,stat,v,op,errmsg***)**

Input Parameters:

char * ***h*** **–** Host name

char * ***h*** **–** Device name

ocsi_db_stats* ***stat*** **–** Address of ocsi_db_stats structure

char * ***v*** **–** VSN

char * *op* – Operation performed, either “r” or “w”

Output Parameters:

char * ***errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

1.6.9 ocs_update_drive_stats_from_ftt()

This function is a combination of *ocs_stat_from_ftt*() and *ocs_update_drive_stats*(). It calls first to fill out internal ocsi_db_stats structure, then calls the second to update statistics in the database.

Usage:

retval = **ocs_update_drive_stats_from_ftt (***h,d,v,op,ftt_d,ftt_f,errmsg***)**

Input Parameters:

char * ***h*** – Host name

char * ***d*** – Device name

ftt_stat_buf ***ftt_d*** – Accumulated FTT statistics over the time the drive was in use

ftt_stat_buf ***ftt_f*** – Final FTT statistics (*after* using the drive)

char * ***v*** – VSN

char * ***op*** – Operation performed, either “r” or “w”

Output Parameters:

char * ***errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

1.7 Dismounting Tape and Deallocating Tape Drives

1.7.1 ocs_deallocate()

This routine deallocates previously allocated tape drive.

Usage:

retval = **ocs_deallocate** (*hostname*, *tapedrive*, *errmsg*)

Input Parameters:

char ****hostname*** – Host name or “localhost” or “any”= ‘\0’=NULL

char ****tapedrive*** – Tape drive name

Output Parameters:

char ****errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

Only a process with the same uid and login name as the process that allocated the tape drive may deallocate it. The deallocating process need not be on the same machine as the allocating process. The super-user process may deallocate any tape drive on the local system, regardless of the identity of the allocating process.

1.7.2 ocs_deallocate_all()

This routine deallocates all tape drives that were allocated by processes with the same UID and the same login name on local host or all hosts. If this function is called by super-user it behaves differently and it deallocates all allocated tape drives but only on the local host.

Usage:

retval = **ocs_deallocate_all** (*opt*, *errmsg*)

Input Parameters:

int ***opt*** – If called by super user:

0 - deallocates drives allocated by the user on local host

1 - deallocates all allocated drives on local host

for other users:

0 - deallocates drives allocated by the user on local host

1 - deallocates drives allocated by the user on all hosts

Output Parameters:

char ****errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

1.7.3 ocs_dismount()

This function unloads or rewinds tape.

The tape drive has to be allocated to the user.

Usage:

retval = **ocs_dismount** (*hostname*, *tapedrive*, *option*, *errmsg*)

Input Parameters:

char *	***hostname***	– Host name or “localhost” or “any” = ‘\0’=NULL
char *	***tapedrive***	– Tape drive name
char *	***option***	– “unload” (any case) to rewind and unload tape; other string to rewind tape

Output Parameters:

char *	***errmsg***	– Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int	***retval***	– 0 for success and negative value for error

1.8 Miscellaneous

1.8.1 ocs_clean_it()

This function requests tape drive to be cleaned and update OCS database.

Tape drive has to be allocated to the user.

Usage:

retval = **ocs_clean_it** (*hostname*, *tapedrive*, *timeout*, *errmsg*)

Input Parameters:

char *	***hostname***	– Host name or “localhost” or “any”=’\0’=NULL
char *	***tapedrive***	– Tape drive name
int	***timeout***	– seconds to wait to satisfy request

Output Parameters:

char *	***errmsg***	– Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

This function calls *ocs_mount_request()* (see Section 1.4.1 on page 14) to request that an operator run a cleaning tape through the tape drive and reply to the request after the tape drive has been cleaned. This function calls *ocs_mount_reply()* (see Section 1.4.2 on page 15) to retrieve the reply. Due to the asynchronous nature of *ocs_mount_reply()*, no other tape mounts can be pending for the calling program.

If the operator successfully replies to the mount request (which indicates that the tape drive was cleaned), the count of the number of times the drive has been cleaned is incremented, the time of the cleaning is updated and other associated counters cleared.

List of errors specific to this function:

OCS_OPER_REPLAY – A subsequent action could not be taken because the operator could not mount the requested tape. This error is currently only returned by *ocs_clean_it()*.

1.8.2 ocs_list_tapes()

This function lists to standard output the tape drives that meet a given criteria.

Usage:

retval = **ocs_list_tapes** (*hostgroup*, *devgroup*, *devtype*, *allocation*, *status*, *opt*, *errmsg*)

Input Parameters:

char ****hostgroup*** – Host group name or “localhost” or “any”=’\0’=NULL
char ****devgroup*** – Device group name or “any”
char ****devtype*** – Device type or “any”
char ****allocation*** – “allocated” or “unallocated”
char ****status*** – “working” or “broken”
char ****opt*** – Type of output (see bellow)

Output Parameters:

char ****errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

An additional input argument, ***opt***, specifies how much information should be listed about each drive, i.e:

long – Display the host name, tape drive name, device type, allocation status, working status, name of the last vsn mounted on the drive, user name to whom the drive is allocated, user id to whom the drive is allocated, and whether the invoking user is authorized to allocate the drive.

pid – Display the host name, tape drive name, device type, allocation status, working status, name of the last vsn mounted on the drive, user name to whom the drive is allocated and PID and PGID of the process which allocated tape drive.

short – Display only the host name & tape drive name of each drive.

nolist – Do not display anything to standard output - the return values with the number of drives that meet requested criteria.

1.8.3 ocs_mark_broken()

This function marks a tapedrive broken in the OCS database and sends mail with the included information from the user, tape drive statistics and a few last reports.

The caller must either have permission to allocate the drive or have OCS administrative privileges.

Usage:

retval **= ocs_mark_broken (***hostname***,** *tapedrive***,** *reason***,** *errmsg***)**

Input Parameters:

char ****hostname*** – Host name or "localhost" or "any"= '\0'=NULL

char ****tapedrive*** – Tape drive name

char ****reason*** – Reason why the tape drive is believed to be broken

Output Parameters:

char ****errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

The input argument ***reason*** is a character string containing the reason that the tape drive is believed to be broken. If **reason** is a NULL character string or the NULL pointer, then standard input will be used to request the reason.

List of errors specific to this function:

OCS_NOT_WORKING – tape drive is already set to broken

1.8.4 ocs_mark_cleaned()

This function increments the number of times the tape drive has been cleaned since it was installed, sets the last time that the tape drive was cleaned to the current time and clears all statistics corresponding to the cleaning time.

The caller must have OCS administrative privileges.

Usage:

retval **= ocs_mark_cleaned(***hostname***,** *tapedrive***,** *errmsg***)**

Input Parameters:

char *	***hostname***	**–** Host name or "localhost" or "any"= '\0'=NULL
char *	***tapedrive***	**–** Tape drive name

Output Parameters:

char *	***errmsg***	**–** Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int	***retval***	**–** 0 for success and negative value for error

1.8.5 ocs_mark_replaced()

This function declares that a tapes drive has been physically replaced. Besides marking the tape drive working, all counters associated with the device are cleared and a new serial number and device type is inserted into the database.

The caller must have OCS administrative privileges.

Usage:

retval **= ocs_mark_replaced(***hostname***,** *tapedrive***,** *serialnum***,** *devtype***,** *errmsg***)**

Input Parameters:

char *	***hostname***	**–** Host name or "localhost" or "any"= '\0'=NULL
char *	***tapedrive***	**–** Tape drive name
char *	***serialnum***	**–** Drive serial number or '\0'=NULL
char *	***devtype***	**–** Device type or '\0'=NULL

Output Parameters:

char *	***errmsg***	**–** Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

If ***serialnum*** or ***devtype*** is a NULL character string this function uses information read from drive and fails if it is impossible.

List of errors specific to this function:

OCS_NOT_BROKEN – Returned by *ocs_mark_working()* when the drive is in the working condition

1.8.6 ocs_mark_working()

This function declares a tape drive to be working.

The caller must have OCS administrative privileges.

Usage:

retval **= ocs_mark_working(***hostname***,** *tapedrive***,** *errmsg***)**

Input Parameters:

char ****hostname*** – Host name or "localhost" or "any"= '\0'=NULL

char ****tapedrive*** – Tape drive name

Output Parameters:

char ****errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

List of errors specific to this function:

OCS_NOT_BROKEN – Returned by *ocs_mark_working()* when the drive is in the working condition

1.8.7 ocs_oper_message()

The routine sends a message to a group of operator consoles belonging to given group.

Usage:

retval **= ocs_oper_message(***opergroup***,** *message***,** *errmsg***)**

Input Parameters:

char ****opergroup*** – Operator consoles group name or "all" char ****message*** – Message to send

Output Parameters:

char ****errmsg*** – Error message set if the error occurs. (It has to be at least 256 characters).

Return Value:

int ***retval*** – 0 for success and negative value for error

1.8.8 ocs_set_db_host()

Usage:

ocs_set_db_host(*hostname***)**

Input Parameters:

char ****hostname*** – Host name

This function does not do anything in OCS v2.0. The use of the DB_HOST environment variable has been removed, and all OCS applications will obtain the location of the DB server by querying the local OCS *taped* daemon.

1.8.9 ocs_cpu_status()

Usage:

ocs_cpu_status(*hostname,option,errmsg,xint***)**

The caller must have OCS administrative privileges.

Input Parameters:

char ****hostname*** – Host name

char ****option*** – program option, (up or down)

char ****errmsg*** – string buffer for returning error message

int * ***xint*** – "X" interface flag. 0 = NO, 1 = YES

This is a new function in OCS v3.1. It will set the CPU status of any node that is in the OCS database to a logical UP or DOWN. To allocate tape drives on a node, the node's CPU status must be "up". If it is down no drive on the node will be allocated. If this function is used, make sure that xint is set to 1. This will suppress the standard out print. Return codes are 0 = success, 0 < = error.

1.9 General OCS utility

This utility has been created to help the user to do basic OCS function in simple calls. The is one function *ocs_util()* to allocate drive, mount tape and do the cleanup like dismount the tape and deallocate the drive. Two additional calls allow user to get some information back like status and device name.

This utility can:

- allocate the drive
- queue allocate the drive and reply or cancel the allocate request
- request the tape mount, reply or cancel mount request, check if the tape is correct and if not issue the next tape mount request
- Do the general cleanup when user decide not use this drive any more which involves cancel all requests, dismount the tape and deallocate the drive

The drive can be allocated by the user and the user can use this function only for the mount tape part but direct usage of the OCS mount function together with this function is not recommended.

1.9.1 ocs_util()

<u>Usage:</u>

retval = **ocs_util** (*id*, *string*, *errmsg*)

<u>Input Parameters:</u>

int	***id***	– ID number for the drive (it can be between 0 an 19)
char *	***string***	– String with parameters

<u>Output Parameters:</u>

char *	***errmsg***	– Error message set if the error occurs. (It has to be at least 256 characters).

<u>Return Value:</u>

int	***retval***	– 0 for success and negative value for error

The string contains parameters with the following format:

“paramter_1**:**paramter_2=value**:**paramter_3”

The list of all parameters[1]:

deb=*n*	–Level of messages to print from 0 which means that nothing is printed to 5 (n=3 is recommended for users)
<u>**allocate**</u>	–to allocate drive

1. The underlined parameters are the actions to take other are just the parameters for the OCS functions

queue	–to queue allocate drive
drive=*drive_name*	–Drive name of the preferred drive to allocate or the drive that is already allocated
host=*host_name*	–Host name of the preferred drive to allocate or the drive that is already allocated (default is *"localhost"*)
device_type=*name*	–Device type name for the allocate (default is *"any"*)
device_group=*name*	–Device group name for the allocate (default is *"any"*)
host_group=*name*	–Host group name for the allocate (default is *"any"*)
mg / nomg	–To set must get flag for allocate
reply	–Reply for the queue allocate or mount request (this is the default action taken after one of those request are made)
timeout=*seconds*	–Timeout for the reply
wait	–Reply with ***timeout***=-1
cancel	–Cancel the queue allocate or mount request
mount	–to make tape mount request
vsn=*VSN*	–Tape label to mount
internal=*VSN*	–Internal tape label if it is different then external label
blank	–The blank tape can be also accepted by the label checking procedure if the tape is requested to write
opmsg	–Message to the operator
read	–To set protection to read only (this is the default)
write	–to set protection to read/write
check / nocheck	–Check tape label and re-request if the tape is wrong
label_type=*type*	–Label type name (the default is *"ansi"*)
nolabel / label	–Do not check if the label type is the same (the default is to check it)[1]
limit	–Limit of mount requests for one tape
unload / nounload	–to unload tape when dismount is made (this is the default)[2]
done	–do the cleanup
one	–works with ***done*** to do the cleanup for only one drive

All parameters but ***timeout*** are stored when the corresponding action is taken.

The following is the list of examples of the basic usage of this function:

To allocate any exabyte drive and mount ANSI tape:

retval =ocs_util(15, "allocate:device_type=exb:mount:vsn=NIC001:wait", errmsg)

This function will return success when the requested ANSI tape is mounted and verified. If the wrong tape is mounted it will dismount it and make request again with the proper

1. This allow for example mount blank tape if the requested one is ANSI
2. Using "nounload" is not recommended unless for testing if the user wants to the tape to stay in the drive

message to the operators. The number of requests per one tape is controlled by ***limit*** parameter with default three mount requests.

When the user wants to have a limited reply time it can use following example:

```
retval = ocs_util(15, “allocate:device_type=exb:mount:vsn=NIC001”,errmsg);
do {
    retval = ocs_util(15,”reply:timout=300”,errmsg);
} while (retval == OCS_TIMEOUT);
```

Mount next tape:

retval = ocs_util(15, “mount:vsn=NIC002:wait”, errmsg)

If the VSN is different then previously requested tape the previous tape is dismounted and new request is made.

Cleanup:

retval = ocs_util(0, “done”, errmsg)

This cancels all requests, dismounts mounted tape and deallocates drive for all used tape drives. This call is strongly recommended to use in case the job is killed to avoid left allocated drives.

List of errors specific to this function:

OCS_UTIL_PARAM	– if the parameters are wrong
OCS_UTIL_ID	– wrong ID number is used
OCS_NOT_MOUNTED	– if the *ocs_mount_reply()* returned status -1

1.9.2 ocs_util_str()

It returns string information stored in the OCS utility table.

Usage:

retval **= ocs_util_str (***id***,** *string***,** *value***)**

Input Parameters:

int ***id***	– ID number for the drive (it can be between 0 an 19)
char ****string***	– Parameter

Output Parameters:

char ****value***	– Returned string

Return Value:

int ***retval*** – 0 for success and negative value for error

The parameters are:

host	–Host name of the allocated drive
drive	–Allocated drive name
basename	–Basename of the allocated tape drive
errmsg	–Error message from the mount reply or from the tape check call
devtype	–Type of the allocated drive

List of errors specific to this function:

OCS_UTIL_PARAM	– if the parameters are wrong
OCS_UTIL_ID	– wrong ID number is used

1.9.3 ocs_util_val()

It returns integer value of the information stored in the OCS utility tables.

Usage:

retval = **ocs_util_val (***id***,** *string***,** *value***)**

Input Parameters:

int ***id*** – ID number for the drive (it can be between 0 an 19)

char ****string*** – Parameter

Output Parameters:

int ****value*** – Returned integer

Return Value:

int ***retval*** – 0 for success and negative value for error

The parameters are:

allocated	–1 if the drive is allocated / 0 if it is not
mounted	–1 if the tape is mounted / 0 if it is not
queued	–1 if the allocate request was made
requested	–1 if the tape mount request was made
replied	–1 if the replied was fetched
status	–value of the error returned from the mount reply or tape check call

List of errors specific to this function:

OCS_UTIL_PARAM	– if the parameters are wrong

OCS_UTIL_ID – wrong ID number is used

1.10 Fortran Interface

A Fortran callable routine exists for each of the previously discussed C callable routines, except *ocs_display_error(), ocs_init_ftt()* and *ocs_report_ftt()*. Each Fortran callable routine handles Fortran character string conversion and passing values, when appropriate, rather than addresses to the C callable routine. From a Fortran program, the routine name and argument list will appear the same as its C counterpart.

The Fortran header file corresponding to the C header, **ocs.h**, is **ocs.inc**. Besides defining the errors codes, **ocs.inc** appropriately declares the value returned by an OCS routine to be an integer. The default Fortran implicit declaration would incorrectly interpret the return value as a floating point.

The following code segment, stripped of almost everything but OCS calls, illustrates the appropriate Fortran calling syntax:

```
#include "ocs.inc"
    integer rval, status, timeout, kbusr, kbdev, mode, nerrs, flags
    character*256 host, tdrive, emsg, devfile, vsn
    vsn = 'rk3420'
    timeout = -1
    rval = ocs_devfile_default()
    rval = ocs_devfile_set (8200, emsg)
    rval = ocs_preference('fndp1', 'xdev1', 99, emsg)
    rval = ocs_allocate('any', 'any', 'exabyte', host, tdrive, emsg)
    rval = ocs_mount_request(host, tdrive, '', vsn, 'r', '', emsg)
    rval = ocs_mount_reply(host, tdrive, status, emsg, timeout)
    rval = ocs_setdev(host, tdrive, 0, emsg)
    rval = ocs_devfile_get(host, tdrive, devfile, emsg)
    rval = ocs_check_label(host, tdrive,'ansi',vsn,'r', emsg)
    rval = ocs_init_stat(host, tdrive, emsg)
    rval = ocs_report_stat(host, tdrive, 'r', kbusr, 'y', emsg)
    rval = ocs_dismount(host, tdrive, 'unload',emsg)
    rval = ocs_deallocate(host, tdrive, emsg)
```

1.11 CPS_TAPE Functions

Following function have been removed from this release of OCS.

- cps_tape_dealloc()
- cps_tape_mount()

- cps_tape_reply()
- cps_tape_umount()

2.0 OCS Command Interface

2.1 List Tape Drives

2.1.1 ocs_tape

Print list of available tape drives that meet given criteria. It calls *ocs_list_tapes()* function (see Section 1.8.2 on page 38) with an arguments corresponding parameters of this function.

ocs_tape **[-h** *<hostgroup>***] [-t** *<devgroup>***] [-d** *<devtype>***] [-a** *<allocation>***]**
[-s *<status>***] [-o** *<option>***]**

Parameters:

parameter	default		description
[**-h** *<hostgroup>*]	"localhost"	–	Host group name
[**-t** *<devgroup>*]	"any"	–	Tape drive group name
[**-d** *<devtype>*]	"any"	–	Device type
[**-a** *<allocation>*]	"any"	–	"allocated" or "unallocated" to display allocated or unallocated tape drives
[**-s** *<status>*]	"any"	–	"working" or "broken"
[**-o** *<option>*]	"long"	–	Type of output ("long", "pid", "short" or "nolist")

The default output argument, if the **-o** option is not specified, is "long". The default host group argument, if the ***-h*** option is not specified is "localhost". If any other option is not specified, the value for the corresponding argument is "any".

If the ***-o*** option is specified with "nolist", the number of drives that met the criteria is listed on standard output.

The command exit status is 0 if *ocs_list_tapes()* succeeds and 1 if an error occurred.

2.2 Allocating Tape Drive

2.2.1 ocs_allocate

This command allocates one tape drive using *ocs_preference()*, *ocs_allocate()* or *ocs_queue_alloc()* functions (see "Allocate Tape Drive" on page 10 and "Queue Allocate Tape Drive" on page 12).

ocs_allocate **[-h** *<hostgroup>***] [-t** *<devgroup>***] [-d** *<devtype>***] [-H** *<pref_host>***]**
[-T *<pref_drive>***] [-m] [-q]**

Parameters:

parameter		default		description
[**-h** *<hostgroup>*	]	"localhost"	–	Host group name
[**-t** *<devgroup>*	]	"any"	–	Tape drive group name
[**-d** *<devtype>*	]	"any"	–	Device type
[**-H** *<pref_host>*	]	NULL	–	Host name of the preferred host
[**-T** *<pref_drive>*	]	NULL	–	Name of the preferred tape drive
[**-m**	]	0	–	Must get flag (see "ocs_preference()" on page 12)
[**-q**	]	0	–	Wait for the drive flag (see "ocs_queue_alloc()" on page 12)

A specific tapedrive can be allocated by specifying a value for the preferred tape drive and using the must-get flag, ***-m***.

When no arguments are specified, OCS will choose a tape drive on the local system. If none are available, for whatever reason, the allocate will fail.

If the ***-m*** option is specified, the ***pref_host*** and ***pref_drive*** restrictions must be satisfied. In other words, OCS cannot allocate any other tape drive other than the preferred one.

The ***-q*** option causes the allocation request to be queued if a tape drive is not immediately available. The request is only queued when tape drives which could have satisfied the request are already allocated or have been marked broken.

The exit status is 0 if the command succeeds and 1 if an error occurred.

2.3 Mount Tapes

2.3.1 ocs_request

This command issues a mount request.

The drive has to be allocated to the user

ocs_request **-t** *<tapedrive>* **-v** *<VSN>* **[-r|-w] [-h** *<hostname>***] [-s** *<tapeset>***]**
[-o *<message>***] [-n** *<seconds>***]**

Parameters:

parameter		default		description
-t *<tapedrive>*			–	Tape drive name
-v *<VSN>*			–	VSN of the requested tape
[**-r | -w**	]	-r	–	flag to mount tape write protected | read-write access
[**-h** *<hostname>*	]	NULL	–	Host name
[**-s** *<tapeset>*	]	NULL	–	Tape set name
[**-o** *<message>*	]	NULL	–	Message to the operator

[**-n** *<seconds>*]	-1	**–**	Number of seconds to wait for tape to be mounted

The required parameters are the name of the tape drive and the volume serial number (vsn) of the tape to be mounted.

Use the ***-r*** option to request that the tape be mounted write-protected, and ***-w*** for write enabled. If the vsn is part of a set of tapes, the name of the tape set can be specified with the ***-s*** option.

An optional message can be sent with the request with the ***-o*** option. This message will be displayed with the mount request on the operator's console.

If the ***-n*** option is specified, the command will wait the specified number of seconds for the mount reply to arrive. If a mount reply does not arrive within the specified number of seconds, the mount request will be canceled. If the ***-n*** option is not specified, the command will wait indefinitely until a mount reply arrives.

The ***-h*** option is only needed if two or more tape drives have the same name, but are located on different hosts.

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.3.2 ocs_pending

This command prints "yes" or "no" to standard output whether or not a mount is pending for the specified tape drive.

ocs_pending **-t** *<tapedrive>* **[-h** *<hostname>***]**

Parameters:

parameter	default		description
-t *<tapedrive>*		**–**	Tape drive name
[**-h** *<hostname>*]	NULL	**–**	Host name

The ***-h*** option is only needed if two or more tape drives have the same name, but are located on different hosts.

The exit status is 0 if the command succeeds and 1 if an error occurred.

2.3.3 ocs_cancel

This command cancels a pending mount request.(see "ocs_mount_cancel()" on page 16).

Must have the same username as the process that issued the mount request or be super-user.

ocs_cancel **-t** *<tapedrive>* **[-h** *<hostname>***]**

Parameters:

	parameter	default	description
-t	*<tapedrive>*		– Tape drive name
[**-h**	*<hostname>*]	NULL	– Host name

The super-user can cancel any mount request for a tape drive on the local host, regardless of the process that initiated the request.

The ***-h*** option is only needed if two or more tape drives have the same name, but are located on different hosts.

The exit status is 0 if the command succeeds and 1 if an error occurred.

2.3.4 ocs_mounted

This function updates all information about the mounted tape in the OCS database for tape which are mounted by user (without requesting tape to be mounted function).

The drive has to be allocated to the user

ocs_mounted -t *<tapedrive>* **-v** *<VSN>* **{-r|-w} [-h** *<hostname>***] [-s** *<tapeset>***]**

Parameters:

	parameter	default	description
-t	*<tapedrive>*		– Tape drive name
-v	*<VSN>*		– VSN
-r|-w			– lag to mount tape write protected | read-write access
[**-h**	*<hostname>*]	NULL	– Host name
[**-s**	*<tapeset>*]		– Tape set name

The exit status is 0 if the command succeeds and 1 if an error occurred.

2.3.5 ocs_check_label

This command checks whether the tape in the tape drive has the appropriate protection and, optionally, the correct tape label (see “ocs_check_label()” on page 18).

The tape drive must be allocated to the user.

ocs_check_label-t *<tapedrive>* **{-r|-w} [-h** *<hostname>***] [-v** *<VSN>***]**
[-l *<label_type>***] [-f** *<device_file>***]**

Parameters:

	parameter	default	description
-t	*<tapedrive>*		– Tape drive name

-r | -w		–	Flags to check if the tape is write protected
[-v <*VSN*>]	NULL	–	VSN to check
[-l <*label_type*>]	NULL	–	Label type ("ansi", "fmb", "tar", "cpio", "blank")
[-f <*device_file*>]		–	This parameter is IGNORED
[-h <*hostname*>]	NULL	–	Host name

If the VSN and label type parameters are not given only the write protection is checked. If the label type is not given the label is assumed to be "ansi".

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.3.6 ocs_check_tape

This command is used the same way like ocs_check_label and it will be removed in future releases of OCS.

(see "ocs_check_label" on page 52).

2.4 Device Files

2.4.1 ocs_devfile

This command will print out the name of the device file corresponding to the tape drive and the specified device attributes.

ocs_devfile **-t** <*tapedrive*> **[-h** <*hostname*>**] [-fc] [-d** <*density*>**]**

Parameters:

parameter	default		description
-t <*tapedrive*>		–	Tape drive name
[**-h** <*hostname*>]	NULL	–	Host name
[**-f** <*block_size*>]	Variable	–	Fixed length records
[**-c**]	No Compression	–	Data Compression
[**-d** <*density*>]	Highest Available	–	Density

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.4.2 ocs_logdev

For a specified device file and optionally specified hostname, this command will print out the host name, corresponding tape drive name and corresponding tape drive parameters from the database.

ocs_logdev **[-h** <*hostname*>**] -f** <*devfile*> **[-o** <*option*>**]**

Parameters:

	parameter		default	description
-t	<*tapedrive*>			– Tape drive name
[-h	<*hostname*>	]	"localhost"	– Host name
[-o	<*option*>	]		– Output options

If the ***-h*** option is not specified or specified with the argument "localhost", then the device file is expected to be on the local host. (The argument to the ***-h*** option is "any" is not accepted). Any other argument to the ***-h*** option is interpreted as a specific host name.

The device file specified by the ***-f*** option must be either the full pathname to a device file. The symbolic link is not longer accepted.

If the ***-o*** option is specified with "short" only the corresponding host and tape drive name are printed to standard output.

If the ***-o*** option is not specified or is specified with an argument other than "short", the host name, tape drive name, device file name and tape drive parameters are printed to standard output. The displayed tape drive parameters are density, rewind, retension, swap, fixed and compression.

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.4.3 ocs_setdev

This command will ensure that the device file given by "ocs_devfile" on page 53 will be physically altered to match the specified attributes.

The drive has to be allocated to the user

ocs_setdev **[-h** <*hostname*>**] -t** <*tapedrive*> **[-ncv] [-f** <*block_size*>**]**

Parameters:

	parameter		default	description
-t	<*tapedrive*>			– Tape drive name
[-h	<*hostname*>	]	"localhost"	– Host name
[-f	<*block_size*>	]		– Block size
[-v		]		– Variable block size
[-c | -n		]	-n	– Data compression or no data compression
[-d	<*density*>	]	highest available	– Density or device mode (e.g. "10G", "8500")

Three device characteristics can currently be set with this command - block size, data compression and density. The ***-c*** options turns compression on, while the ***-n*** option turns it off. The ***-v*** option select variable block size, and the ***-f*** option select a fixed block size. The ***-d*** option sets specific density or mode for the device. If the ***-d*** option is not specified, then the highest available density or mode for the device will be used.

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.4.4 ocs_setown

This command calls *ocs_setown()* (see Section 1.5.9 on page 27) to change the ownership and protection mode of the device files for the given tape drive.

The drive has to be allocated to the user

ocs_setdev **[-h** *<hostname>***] -t** *<tapedrive>*

Parameters:

	parameter	default	description
-t *<tapedrive>*		–	Tape drive name
[**-h** *<hostname>*]	“localhost”	–	Host name

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.5 Device Statistics

2.5.1 ocs_devstat

This command will query a tape drive device driver for status information and display the information on standard output.

ocs_devstat **{-f** *<devfile>***} | {-t** *<tape_drive>***} [-h** *<hostname>***]**

Parameters:

	parameter	default	description
-t *<tapedrive>*		–	Tape drive name
-f *<devfile>*		–	Device file name
[**-h** *<hostname>*]		–	Host name

If the ***-t*** option is used, tape_drive must be tape drive in the OCS database. The ***-h*** option is only needed if two or more tape drives have the same name, but are on different hosts.

The ***-f*** option and ***-t*** option are mutually exclusive. If the device filename is given default for host name is “localhost” in other case it is the name of the host.

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.5.2 ocs_init_stat

This command retrieves the current tape drive use statistics from the device and stores them in the OCS database (see function “ocs_init_stat()” on page 27)

The drive has to be allocated to the user

ocs_init_stat **[-h** *<hostname>***] -t** *<tapedrive>*

Parameters:

parameter	default	description
-t *<tapedrive>*		– Tape drive name
[**-h** *<hostname>*]	“localhost”	– Host name

It should be called immediately before opening a file for reading or writing. After the file is closed, *ocs_report_stat* should be called to retrieve the tape drive use statistics from the device again, calculate the net changes and to save the statistics in the OCS database.

The exit status is 0 if the command succeeds and 1 if an error occurred.

2.5.3 ocs_report_stat

This command collects the tape drive use statistics from the device and creates an OCS database record for that tape drive (see “ocs_report_stat()” on page 28).

The drive has to be allocated to the user

ocs_report_stat [-h *<hostname>***] -t** *<tapedrive>* **-k** *<kb_usr>* **{-r | -w} [-o** *<opt>***]**

Parameters:

parameter	default	description
-t *<tapedrive>*		– Tape drive name
-k *<kb_usr>*		– Kbytes read or written by user
-r | -w		– Read only or read-write access to tape
[**-h** *<hostname>*]	“localhost”	– Host name
[**-o** *<opt>*]	“yes”	– Display statistics information

If the ***-o*** option has an argument other than “nolist’, then the tape drive use report will be displayed to standard output as well as entered into the OCS database.

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.5.4 ocs_reports

This command lists, to standard output, the tape drive use reports that meet a given criteria. The command simply calls *ocs_list_reports()* (see Section 1.6.5 on page 31) so each option corresponds to an *ocs_list_reports()* argument.

ocs_reports [-h *<hostname>***] [-t** *<tapedrive>***] [-s** *<serial>***] [-v** *<VSN>***]**
[-a *<adate>***] [-b** *<bdate>***] [-o** *<opt>***]**

Parameters:

parameter	default	description
[**-h** *<hostname>*]	“any”	– Host name
[**-t** *<tapedrive>*]	“any”	– Tape drive name

[**-s** *<serial>*]	"any"	–	Serial number
[**-v** *<VSN>*]	"any"	–	VSN
[**-a** *<adate>*]	"any"	–	Date like "mm/dd/yy" of the oldest reports to list
[**-b** *<bdate>*]	"any"	–	Date like "mm/dd/yy" of the newest reports to list
[**-o** *<opt>*]	"none"	–	f it is "delete" all listed reports are deleted (only for administrators) or "short" for shorter listing

The command exit status is 0 if the command succeeds and 1 if an error occurred.

See *ocs_list_reports()* function for explanation of each listed field.

2.5.5 ocs_stats

This command displays, to standard output, the summary use statistics of the given tape drive. The command simply calls *ocs_list_stats()* (see Section 1.6.6 on page 32) so each option corresponds to an *ocs_list_stats()* argument.

ocs_stats **[-h** *<hostname>***] -t** *<tapedrive>*

Parameters:

parameter	default		description
-t *<tapedrive>*		–	Tape drive name
[**-h** *<hostname>*]	"localhost"	–	Host name

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.5.6 ocs_update

This command shows and updates information about given tapedrive.

The caller must have OCS administrative privileges.

ocs_update **-t** *<tapedrive>* **[-h** *<hostname>***] [-S | -s** *<serial>***]**
[-D | -d *<devtype>***] [-o <opt>]**

Parameters:

parameter	default		description
-t *<tapedrive>*		–	Tape drive name
[**-h** *<hostname>*]	NULL	–	Host name or "localhost"
[**-S**]		–	flag to change serial number in the OCS database to one read from the drive
[**-s** *<serial>*]		–	Change serial number in the OCS database to given one
[**-D**]		–	Flag to device type in the OCS database to one read from drive

[**-d** *<devtype>*]		**–**	Change device type in the OCS database to given on
[**-o** *<opt>*]	"show"	**–**	Use "update" to really update information in the database.

Information updated and shown by this utility:

- Vendor Id
- Product Id
- Controller
- Frimware
- Device type (only on specific user request using ***-D*** or ***-d*** parameters)
- Serial number (only on specific user request using ***-S*** or ***-s*** parameters)

2.6 Dismount Tape and Deallocate Tape Drive

2.6.1 ocs_dismount

This command rewinds and unloads or only rewinds tape in the given tape drive (see "ocs_dismount()" on page 37).

The drive has to be allocated to the user

ocs_dismount [-h *<hostname>* **] -t** *<tapedrive>* **[-o** *<opt>* **]**

Parameters:

parameter	default		description
-t *<tapedrive>*		**–**	Tape drive name
[**-h** *<hostname>*]	NULL	**–**	Host name
[**-o** *<opt>*]	"unload"	**–**	If a string other than "unload" is specified, the tape will only be rewound

The exit status is 0 if the command succeeds and 1 if an error occurred.

2.6.2 ocs_deallocate

This command is used to deallocate one or more tape drives.

ocs_deallocate[-t *<tapedrive>* **] [-h** *<hostname>* **] [-aAcS] [-p** *<pid>* **]**

Parameters:

parameter	default		description
[**-t** *<tapedrive>*]	NULL	**–**	Tape drive name
[**-h** *<hostname>*]	NULL	**–**	Host name

[**-a**]		–	flag to deallocate all drives on the localhost allocated by user
[**-A**]		–	flag to deallocate all drives allocated by user or if executed as root all drives on localhost
[**-c**]		–	flag to check if the process which allocated tape drive still exists in the system and do NOT deallocate such drive in this case
[**-S**]		–	flag to deallocate drive allocated by other user if the user calling it has admin privileges
[**-p** *<pid>*]	NULL	–	check if the drive is allocated to the process with given PID. If the drive is allocated to other process the drive is not deallocated

A specific tape drive can be deallocated by using the ***-t*** option. The ***-h*** option is only needed if two or more tape drives have the same name, but are located on different hosts.

The ***-a*** option will cause all tape drives on the local host that are allocated by the user to be deallocated. The ***-A*** option causes every tape drive in the database allocated by the user to be deallocated. This means that tape drives allocated to the same username on other hosts can be deallocated with this option.

The ***-A*** has different meaning when the user is root. In this case, all tape drives on the local host, regardless of ownership, are deallocated. This allows root to have supreme power on the local host, but it cannot deallocate any tape drives on other hosts.

If a tape drive with a pending mount request is deallocated, the mount request is automatically cancelled.

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.7 Miscellaneous

2.7.1 ocs_all_pending

This command lists, to standard output, the all the mount requests that are currently pending.

ocs_all_pending [-h]

Parameters:

parameter	default		description
[**-h**		–	Include a column heading over the list.

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.7.2 ocs_auth

This command can be used to add or delete a user from an authorization group.

The caller must have OCS administrative privileges.

ocs_auth {-a| -d} -l *<login>* **-u** *<uid>* **-g** *<authgroup>* **[-c]**

Parameters:

parameter	default		description
-a | -d		–	Add or delete user from a group
-l *<login>*		–	User login name
-u *<uid>*		–	User UID number
-q *<authgroup>*		–	Authorization group name
[**-c**]		–	Flag to create new group

The ***-a*** option is used to add a user to a group, and the ***-d*** option will delete the user from a group.

The ***-c*** option will cause the group to be created if it doesn't already exist. This only applies when a user is being added.

The exit status is 0 if the command succeeds and 1 if an error occurred.

2.7.3 ocs_broken

This command marks the specified tape drive broken. In addition, a mail message is sent to users administered for mail from the host containing the tape drive.

The caller must either have permission to allocate the drive
or have OCS administrative privileges.

oc.s_broken [-h *<hostname>* **] -t** *<tapedrive>*

Parameters:

parameter	default		description
-t *<tapedrive>*		–	Tape drive name
[**-h** *<hostname>*]	NULL	–	Host name

The exit status is 0 if the command succeeds and 1 if an error occurred.

2.7.4 ocs_cleaned

This command increments the number of times the tape drive has been cleaned since it was installed, sets the last time that the tape drive was cleaned to the current time and clears all statistics corresponding to the last cleaning period.

The caller must have OCS administrative privileges.

ocs_cleaned **[-h** *<hostname>***]** **-t** *<tapedrive>*

Parameters:

parameter	default	description
-t *<tapedrive>*		– Tape drive name
[**-h** *<hostname>*]	NULL	– Host name

The exit status is 0 if the command succeeds and 1 if an error occurred.

2.7.5 ocs_clean_list

To help determine whether a tape drive should be cleaned, this command lists, to standard output, the tape drives that meet a given criteria.

ocs_clean_list [-m *<mounts>***] [-d** *<mb_dev>***] [-u** *<mb_usr>***] [-c** *<cleanit>***]**
[-s *<status>***] [-a** *<allocation>***] [-o** *<option>***]**

Parameters:

parameter	default	description
[**-m** *<mounts>*]	0/MAXINT	– Minimum number of mounts (
[**-d** *<mb_dev>*]	0/MAXINT	– Minimum number of Mbytes read and written reported by the drive
[**-u** *<mb_usr>*]	0/MAXINT	– Minimum number of Mbytes read and written reported by the user
[**-c** *<cleanit>*]	“any”	– Cleaning bit “on” or “off”
[**-s** *<status>*]	“any”	– Status of the drive “working” or “broken”
[**-a** *<allocation>*]	“any”	– “allocated” or “unallocated”
[**-o** *<option>*]	“long”	– Output format - “short” to display only the host name and tape drive name without the header line.

The criteria is formed by OR-ing the conditions given by ***-m***, ***-d***, ***-u***, and ***-c*** options and by AND-ing those given by ***-s*** and ***-a*** options. In other words, a tape drive is listed if it meets not only one or more conditions specified by ***-m***, ***-d***, ***-u***, and ***-c*** options but also every condition given by ***-s*** and ***-a*** options.

Thee default for ***-m***, ***-d*** and ***-u*** is 0 (“any” for ***-c***) if none of them is specified and MAXINT (the maximum representable integer on the system) otherwise.

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.7.6 ocs_clean_it

This command allocate(s) and requests to have a cleaning tape run through one drive or list of drives. After a drive is cleaned, the drive is deallocated and the count of the number of times the drive has been cleaned is incriminated.

ocs_clean_it **[[-t** <*tapedrive*> **[-h** <*hostname*>**]] [-n** <*mtimeout*>**]**
[[-A] [-q <*qtimeout*>**] | -N]**

Parameters:

parameter	default		description
[-t <*tapedrive*> **]**		–	Tape drive name
[-h <*hostname*> **]**	NULL	–	Host name (valid only with ***-t*** parameter)
[-n <*mtimeout*> **]**	3600	–	Maximum number of seconds to wait for a mount request to be satisfied after the last drive has been allocated
[-q <*qtimeout*> **]**	14400	–	Maximum number of seconds to queue for a drive that is currently allocated (not valid if ***-N*** is specified)
[-A]		–	Flag to queue allocate for the drives allocated by the user (not valid if ***-N*** is specified)
[-N]		–	Do not queue allocate for any allocated drive

The default is to queue allocate drives for drive allocated to someone else then the user and skip drive allocated to the user (unless ***-N*** or ***-A*** flags are used). Such drive will be checked for allocation (via *ocs_alloc_reply()*) every 300 seconds (i.e. 5 minutes) until this limit is exceeded. (If this limit is less than 5 minutes, the drives are less than 5 minutes.)

If the ***-t*** option is not present, standard input should be a list tape drives to be cleaned. It can use the short output of the *ocs_clean_list* command as input. For example:

ocs_clean_list -m 5 **-o** short **|** **ocs_clean_it -q** 7200

will request a cleaning for every drive that has had more than five mounts since the last cleaning and will wait at most 2 hours for drives to be allocated.

To request a specific drive to be cleaned without queuing for allocation and waiting at most 90 minutes for the operator response enter:

ocs_clean_it **-h** cdfsga **-t** cdf100 **-m** 5400 **-N**

The success or failure of cleaning each drive is listed to standard output.

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.7.7 ocs_left_allocated

Give a list of tape drivers that are allocated but haven't received any mount requests for a given period of time.

ocs_left_allocated [-h *<hostname>***] [-t** *<tapedrive>***] [-l** *<reqtime>***]**

<u>Parameters:</u>

parameter	default	description
[**-t** *<tapedrive>*]	"any"	– Tape drive name
[**-h** *<hostname>*]	"any"	– Host name
[**-l** *<reqtime>*]	48	– Number of hours drive was allocated

In the LAST MOUNT field, the "no mount" value means that the device didn't received a mount request since it was allocated. The STATUS field's value is the mount request status: R - request pending, F - failure and S - success.

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.7.8 ocs_message

This command sends a message to one or more operator consoles.

ocs_message -m *<message>* **[-g** *<oper_group>***]**

<u>Parameters:</u>

parameter	default	description
-m *<message>*		– Message
[**-q** *<oper_group>*]	"all"	– Operator group name

If no operator group is specified, the message will be delivered to all operator consoles currently running in the OCS system. Messages are not saved. This means that if a message is sent and afterwards another operator console joins the OCS system, it will not receive the message.

A specific group of operator consoles can be targeted by specifying an operator group. This feature might be useful if different operator consoles service different tape drives. If the message concerns a specific tape drive, it would not be necessary to send the message to consoles not servicing that device.

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.7.9 ocs_mrlog

This command lists, to standard output, the mount requests use that meet a given criteria..

ocs_mrlog [-h *<hostname>***] [-t** *<tapedrive>***] [-u** *<username>***] [-v** *<VSN>***]**
[-a *<adate>***] [-b** *<bdate>***] [-o** *<opt>***]**

<u>Parameters:</u>

parameter	default	description
[**-h** *<hostname>*]	"any"	– Host name

[**-t**	*<tapedrive>*]	"any"	– Tape drive name
[**-s**	*<username>*]	"any"	– Username
[**-v**	*<VSN>*]	"any"	– VSN
[**-a**	*<adate>*]	"any"	– Date like "mm/dd/yy" of the oldest mounts to list
[**-b**	*<bdate>*]	"any"	– Date like "mm/dd/yy" of the newest mouns to list
[**-o**	*<opt>*]	"none"	– f it is "short", display "failure" for a failed mount rather than the full reply.

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.7.10 ocs_repaired

This command puts a broken tape drive back in service. In addition, a mail message is sent to users administered for mail from the host containing the tape drive.

The caller must have OCS administrative privileges.

ocs_repaired **[-h** *<hostname>***]** **-t** *<tapedrive>* **{-r | -w}**
[-s *<serial>***]** **[-d** *<device_type>***]**

Parameters:

parameter		default	description
-t	*<tapedrive>*		– Tape drive name
[**-h**	*<hostname>*]	NULL	– Host name
-r			– To replace drive by ne one
-w			– To mark broken drive working without replacing it.
[**-s**	*<serial>*]		– Serial number (not valid with ***-w*** parameter)
[**-d**	*<device_type>*]		– Device type (not valid with ***-w*** parameter)

When the tape drive was physically replaced, then the ***-r*** option must be used. In this case the installation date and associated tape drive use counters are reset, and the tape drive is marked working. The new serial number and device type are necessary but if they are not specified by the user an attempt is made to get them directly from the device and if this is impossible the error is returned.

If the tape drive was not physically replaced, the ***-w*** option must be used. In this case, the tape drive is marked working, but the installation date and associated tape drive use counters are not reset.

The exit status is 0 if the command succeeds and 1 if an error occurred.

2.7.11 ocs_terminst

This command is command line version of the X window command *ocs_install* and should only be used by root. The command will interactively allow the following commands and associated values to be specified:

ocs_terminst

Available commands:

do – dbserv | taped | tables | cpu | users

taped – rpcnum | bindir | workdir | dbhost | next | prev

dbserv – rpcnum | bindir | datadir | next | prev

show

help

exit

For example, the show command will display current OCS installation parameters for the local host.

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.7.12 ocs_xfer

This command is used to transfer records out of the OCS relational database and into a flat text file for storage. The records are then deleted from the OCS database.

The caller must have OCS administrative privileges.

ocs_xfer **[-h] -t** *mountlog* **[-a** *<after_time>* **-b** *<before_time>***]**

This command is primarily used to trim down the size of 'log' tables that accumulate information, such as the mountlog table, which records every tape mount request submitted to the OCS system.

The *ocs_xfer* command prints database records to standard output. Each database record is printed in a single line of text with each field separated by a space. If a database field is a text string that might contain spaces, the field will be printed surrounded by double quotes. The goal is to make the output easily parsable by other programs.

The ***-h*** option will cause a header containing the names of the database fields to be printed.

The command exit status is 0 if the command succeeds and 1 if an error occurred.

2.7.13 ocs_cpu_status

This command is used to set or list the logical status of a node's CPU. Valid CPU status is "up" or "down". Setting of the CPU status to "down" will cause the tape drives on this node to become ineligible for user allocation. This command can be used to take a node out of service from OCS without powering the node off. The CPU status of an OCS node must be “up” in order to allocate tape drives on that node.

The invoker must have OCS administrative privileges.

ocs_cpu_status [-h *<hostname>* **] {-u | -d | -l}**

<u>Parameters:</u>

parameter	default	description
-h		– Host name.
-u		– Change CPU status to up.
-d		– Change CPU status to down.
-l		– List CPU status.

If the -h option is not specified localhost is assumed. If no command option is specified “list” is assumed. The host option or an action option must be specified. If no option is specified the command help/usage line prints. In other words, at least one option must be specified in order for the command to work. A short output listing is displayed to standard out. For the “up” or “down” option the result of the change to the host is shown. For the “list” option the CPU status for all the nodes in the OCS CPU database are displayed. The “host” option is ignored by the “list” option. Also for each entry in the output listing, there is a “Y” or “N” indicator that shows if there are allocated drives on the node. The CPU status functionality also exists in the OCS “X” interface XOCS. If the CPU status of a node changes from “up” to “down” and there are allocated tape drives, these drives stay allocated until they are deallocated.

The exit status for this command is 0 for success and less then 0 if an error occurred.

3.0 OCS X Window Interface

This section summarizes the capabilities of some of the X Window interfaces that are available with OCS.

3.1 xocs

Xocs is a utility for viewing, manipulating and administering the OCS tape drives. The general display capabilities are:

- view the status of tape drives that meet a criteria, such as:
  - only tape drives that are working and unallocated
  - only tape drives that are allocated to a particular user
- view pending mount requests
- view the log of mount request
- view the available device files for a particular tape drive
- view the summary use statistics of a particular tape drive
- view the log the tape drive use reports
- view the cleaning history of tape drives

General tape drive manipulation activities are:

- allocate a tape drive
- deallocate a tape drive
- mark a tape drive broken

Activities available only to OCS administrators are:

- mark a tape drive working
- mark that a tape drive has been replaced
- mark that a tape drive has been cleaned
- add or delete a CPU from the OCS Database; CPU characteristics such as the CPU flavor or administration mail address may also be updated with this activity
- add, update or delete a tape drive from the OCS Database; tape drive characteristics such as the serial number, device type, or base device file may also be updated with this activity
- add or delete user from OCS database
- change a node's CPU status to "up" or "down".

3.2 ocs_group

Ocs_group is a utility for administering and viewing OCS authorization and device groups. General display capabilities are:

- view available authorization and device groups
- view which CPUs, tape drives and users are members of an authorization group
- view which CPUs and tape drives are members of a device group
- view the pool of CPUs, tape drives and users that could be added to an authorization group or device group

Activities available only to OCS administrators are:

- add or delete a CPU, tape drives or user from an authorization group
- add or delete a CPU and tape drives from a device group
- add or delete an authorization or device group

There are two special authorization groups: “admin” and “public”. Users who are members of the “admin” authorization group have OCS administrative privileges for all CPUs and tape drives. OCS installation initially creates the authorization group “admin” with root as the only authorized user. The other special authorization group, “public”, may be created to authorize all users access to particular tape drives. Since all users automatically are members of the authorization group “public”, *ocs_group* will not add users to that group. If a tape drive is a member of the authorization group “public”, then that tape drive may be allocated to any user. If a CPU is a member of the authorization group “public”, then all tape drives on that CPU may be allocated to any user.

3.3 xtape

Xtape is a utility for OCS operators to respond to OCS mount requests. There is a separate document titled “**XTape**” that describes how to use it.

3.4 xtapeview

Xtapeview is a version of *xtape* that allows OCS users to view, but not respond, to OCS mount requests.

3.5 ocs_install

Ocs_install is a utility for installing OCS and should only be used by root. When it is used to install a database server, the DB_HOST environment variable must be set.

3.6 xfarmdb

Xfarmdb is a utility for viewing the OCS database structure and individual database records. It is provided for OCS administrators to aide in debugging an OCS database problem. (The ***select***, ***delete***, ***insert*** and ***update*** buttons are currently non-operational.)

Since it communicates with the database server and by design does not communicate with a taped daemon, the DB_HOST environment variable must be set.

3.7 Using XOCS

See 3.1 for a summary of XOCS features.

Starting XOCS.

XOCS is started with the command “xocs”. When XOCS is started, the default tape list criteria displayed under *“Matching Tape Drives”* is based on the *“radio selection buttons”* being set to *“any”*. This is the default for XOCS startup. The tape list displayed can be modified by changing the selected *“radio buttons”* on one or more categories (Allocation, Host O/S Flavor, Tape Status, CPU Status and Device Type). There is also four “fill in boxes” in the lower left hand side of the XOCS window. These controls can be used independently or in conjunction with the “radio buttons” to refine the criteria for tape drive selection. The “fill in boxes” are “Allocated by” (tape drives allocated by a user), “Hostname” (tape drives on a specific host), “Last VSN” (tape drive with this Volume Serial Number) and “Group” (OCS group). After the tape selection criteria has been set, the “Find Matches button” must be clicked with the mouse for the changes to take effect.

The Menu Bar.

The Menu Bar is located at the top left of the XOCS window. It has three selections. View, Action and DB.

View.

See 3.1 for a summary of the View Menu functions.

Action.

The first three items from the Action menu *Allocate* (a tape drive), *Deallocate* (a tape drive) and *Mark Broken* (tape drive) are for general usage. See 3.1 for a summary of these functions.

The last five items, *Mark Working* (tape drive), *Cleaned* (tape drive), *Replaced* (tape drive), *CPU Status Up* (host) and *CPU Status Down* (host) are for OCS Administrators only. See 3.1 for a summary of all items except the CPU Status functions.

The CPU Status functions.

CPU status allows a OCS administrator to logically bring an OCS host’s CPU “up” or “down”. With OCS version 3.1 the CPU status of an OCS host must be “up” in order to allocate tape drives. The “down” status is useful when a host requires maintenance. It will no longer be necessary to “mark” all of the tape drives broken so that OCS does not allocate any drives on a particular host.

When the CPU status of a host is changed from “up” to “down”, tape drives that were previously allocated will stay allocated until they are unallocated. But there will be no new allocations until the CPU status is changed from “down” to “up”.

To change a host's CPU status, you may select a tape drive that is on the host you want to change though this is not necessary. Select the ACTION Item from the menu bar. Select CPU STATUS UP or CPU STATUS DOWN from the ACTION menu. If a tape drive was not selected from the tape list, a small window will popup and prompt you for the hostname. Click either "OK" or "CANCEL" button to continue. Regardless of how it's done, this will be followed by a small window with the message "Update display to show new cpu status?". Click the "Yes" button with the mouse or press the "ENTER" key on the keyboard. The tape list is updated with the new cpu status.

DB.

There are three items in the DB menu. They are "Host Info" (adding or deleting hosts), "User Info" (adding or deleting users) and "Tape Drive Info" (addling or deleting tape drives). You must have OCS administrator privileges to use these functions.

To Exit XOCS.

To exit XOCS click the "Dismiss" button with the mouse.

4.0 Administrate OCS

4.1 Machine Reboots

A program is provided with the OCS software, called **ocs_reboot**, which performs whatever tasks are necessary to synchronize the OCS software and database after a reboot of the local host. This is NOT an installation program and should NOT be run except after a reboot. At the present time, the actions taken are:

1. Deallocate all tape drives on the local host.
2. Remove (from the OCS database) any ***devd*** daemon entries for the local host.
3. Remove (from the OCS database) tape drive queued allocation bids from users on the local host.

4.2 Tape Client Crashes

While a tape mount request is pending, the OCS system, namely the ***taped*** daemon, periodically checks to see if the requesting process still exists. If the process crashes or exits without receiving a mount reply, the mount request is automatically cancelled and the tape drive is deallocated. The database tables are updated appropriately.

4.3 Locating the Database Server

The OCS software no longer users the DB_HOST environment variable to locate the database server. In v2.0 and greater, OCS applications will obtain the location of the DB server by querying the local ***taped*** daemon.

Multiple ***taped*** daemons can run on a single system, as long as each listens at a unique RPC number and/or version. Each ***taped*** daemon should be associated with a unique database server. If multiple ***taped*** daemons run on a single system, the OCS application can select one by setting the **TAPED_RPC_NUM** and **TAPED_RPC_VERS** environment variables.

4.4 Database Backups

A program is provided with the OCS software, called *ocs_backup*, which creates a file of SQL statements that can subsequently be used to rebuild an OCS database. Since it communicates with the database server and by design does not communicate with a ***taped*** daemon, the DB_HOST environment variable must be set.

If *ocs_backup* is invoked with no arguments, the entire database will be dumped to standard output, e.g. “ocs_backup > db.backup”. To restore a database from such a backup:

1. Kill the ***dbserv*** process and remove all the database files from the database directory.
2. Create the database tables again using *ocs_install*.
3. Feed the backup SQL script into the SQL interpreter using, e.g.“*sqlint* < db.backup”.

When making OCS backups, make sure there are not an excessive number of mount log records or tape drive use report records. If there are old records, these should be first removed and stored to text files using *ocs_xfer* and *ocs_reports*.

In the rare event that an OCS database becomes corrupted, *ocs_backup* may be used to dump selected database tables to standard output. Either tables to be included or excluded may be specified. For example, to dump only the STATLOG and MOUNTLOG tables to standard output, enter:

ocs_backup -i *statlog mountlog*

or to dump all but the STATLOG and MOUNTLOG tables to standard output, enter:

ocs_backup -e *statlog mountlog*

Valid database table names are

cpu	unixuser	tapedrive	devfile
groupid	tapedaemon	cpugroup	tapegroup
usergroup	tapeclient	mountrequest	oper
opergroup	operhost	opertape	tapebid
devd	statlog	tapestat	mountlog

5.0 Operator Console Programmer's Interface

The OCS software contains a subroutine library, liboper.a, which contains the subroutines required to build an operator console. An operator console program is both an RPC server and an RPC client. It must accept RPC calls from a system daemon notifying it of events, such as tape mount requests or replies. It must also make RPC calls to the same system daemon to send messages to the system, such as a tape mount reply. The operator console must also be able to accept keyboard or mouse input from the operator.

To accept RPC calls from the system, the programmer must write a number of routines that the OCS software will call when a certain event happens. For example, if the system sends a tape mount request to the operator, the *oper_mount_request*() routine will be called. The programmer must write a routine called *oper_mount_request*() which accepts the appropriate parameters. These predefined routines that the programmer must supply are called callback routines.

An operator console typically has the following structure. First, the library routine *oper_init*() is called to initialize the software. Next, the operator console program can call a series of routines to tell the system what tape drives it will be servicing. For example, if the operator console is going to service all tape drives on three different hosts, the library routine *oper_service_host*() would be called for each host. The library routine *oper_service_all()* could be called to service all tape drives in the entire system. Once the tape drives to be serviced have been specified, the operator console can begin accepting RPC calls.

The operator console can accept RPC calls in two ways. It can periodically call the library routine *oper_check_RPC*() which will check to see if an RPC call is pending and will service it by calling the appropriate callback routine. The other alternative is to call the library routine *oper_set_alarm*(). This will set up an alarm (SIGALRM) to go off every 5 seconds and check for pending RPC calls. For each pending RPC call found, the appropriate callback routine is called.

Two important callback routines (which the programmer must write) are *oper_add_tapedrive* and *oper_del_tapedrive*(). These routines are called when the operator console should add or remove a tape drive from the console. For example, if the operator console calls *oper_service_host*() to service all tape drives on a particular host, the system will make an RPC call to *oper_add_tapedrive*() for each tape drive on the host. Also, if a tape drive is later removed from the host, *oper_del_tapedrive*() will be called.

If an error occurs from within a subroutine in liboper.a, a programmer-supplied routine called *display_error*() will be called to display the error. This allows the operator console to display error messages in the appropriate way. For example, if the operator console uses X-windows, a dialog window with the error message might pop up.

5.1 Operator Console Library Routines (liboper.a)

These are subroutines that are called by the operator console in response to keyboard or mouse input, or during operator console startup or shutdown.

5.1.1 oper_init()

This routine must be called to initialize the OCS software.

Usage:

retval = **oper_init()**

Return Value:

int ***retval*** **–** 0 for success and negative value for error

5.1.2 oper_exit()

This routine must be called before the operator console is shutdown. There is no return value for this routine.

Usage:

oper_exit()

5.1.3 oper_service_all()

This routine declares that the operator console will service all tape drives on all hosts.

Usage:

retval = **oper_service_all()**

Return Value:

int ***retval*** **–** 0 for success and negative value for error

The system will make RPC callbacks to *oper_add_tapedrive()* for each tape drive. If *oper_service_all()* returns zero, the call was successful.

5.1.4 oper_service_host()

This routine declares that the operator console will service all tape drives on a particular host.

Usage:

retval = **oper_service_host(***hostname***)**

Input Parameters:

char ****hostname*** – Host name

Return Value:

int ***retval*** – 0 for success and negative value for error

The system will make RPC callbacks to *oper_add_tapedrive()* for each tape drive on ***hostname***.

5.1.5 oper_service_queue()

If the Distributed Batch System is running, the operator console can declare that it will service all tape drives usable by jobs on a particular batch queue. The system will make RPC callbacks to *oper_add_tapedrive()* for each tape drive.

Usage:

retval = **oper_service_queue(***queuename***)**

Input Parameters:

char ****queuename*** – Queue name

Return Value:

int ***retval*** – 0 for success and negative value for error

5.1.6 oper_service_group()

This routine declares that the operator console will service all tape drives in a particular group. The system will make RPC callbacks to oper_add_tapedrive() for each tape drive in the group specified.

Usage:

retval = oper_service_group(*groupname*)

Input Parameters:

char ****groupname*** – Group name

Return Value:

int ***retval*** – 0 for success and negative value for error

5.1.7 oper_service_tape()

This routine declares that the operator console will service a specific tape drive. The system will make an RPC callback to oper_add_tapedrive() for the tape drive specified.

Usage:

retval **= oper_service_tape(***hostname***,** *tapedrive***)**

Input Parameters:

char *	***hostname***	– Host name
char *	***tapedrive***	– Tape drive name

Return Value:

int	***retval***	– 0 for success and negative value for error

5.1.8 oper_send_reply()

This routine issues a tape mount reply.

Usage:

retval **= oper_send_reply(***hostname***,** *tapedrive***,** *status***,** *message***)**

Input Parameters:

char *	***hostname***	– Host name
char *	***tapedrive***	– Tape drive name
int	***status***	– Status of the mount
char *	***message***	– Message

Return Value:

int	***retval***	– 0 for success and negative value for error

The mount reply will be received by the process that requested the tape mount, and by all operator consoles servicing the tape drive. The first two arguments, ***hostname*** and ***tapedrive***, specify the tape drive. The third argument, *status*, should be set to 0 if the tape was mounted successfully, and -1 if no tape was mounted. The last argument, ***message***, should point to a character string explaining the reason the tape could not be mounted, or any informational message if the tape was successfully mounted. If no message is applicable, ***message*** can be passed as NULL.

The return value of this function describes the failure or success of the mechanics of sending the reply. The following values can be returned:

0 Success

1 The tape drive was already mounted by another operator. This only occurs if two or more operators respond to the same mount request nearly simultaneously.

2 No mount request was found in the database. This can happen if two or more operators respond to the same mount request nearly simultaneously, the operator consoles are running on different machines, and the machines have widely different loads causing one operator to run much slower than the other.

-1 The reply could not be sent. This can happen if the tape daemon (taped) could not be reached, or the database server (dbserv) couldn't be reached.

-2 The tape drive was not found in the database. This can happen if the operator console contains a bug and passes bad parameters to the routine.

5.1.9 oper_check_RPC()

This routine will check to see if any RPC calls from the system are pending. If so, each RPC call will be serviced by calling the appropriate callback routine. This routine is meant to be called periodically.

Usage:

oper_check_RPC()

5.1.10 oper_set_alarm()

This routine will establish a signal handler for SIGALRM and start an alarm which will go off every 5 seconds. The signal handler will check for pending RPC calls and service them by issuing the appropriate callbacks. If this method is used for servicing RPC calls, the programmer must be aware of which parts of the operator console's code is critical and cannot be interrupted. For these sections of code, the SIGALRM signal must be held and then released.

Usage:

oper_set_alarm()

5.2 Operator Console Callback Routines

These routines must be written by the operator console programmer and are called when certain events occur, such as a tape mount request. One special routine, *display_error*(), is called whenever there is an error from a routine in liboper.a.

5.2.1 display_error

This routine should display an error message on the operator's console. The programmer should write this routine to utilize whatever type of user interface is used, such as X-windows or curses.

Usage:

display_error(*message***)**

Input Parameters:

char ***message** – Message to display

5.2.2 oper_mount_request()

This routine should present a tape mount request on the operator console.

Usage:

oper_mount_request(*host*, *drive*, *tapeset*, *vsn*, *prot*, *msgflag*, *msg*, *user*, *uid*, *reqtime***)**

Input Parameters:

char *	***host***	– Host name
char *	***drive***	– Tape drive name
char *	***tapeset***	– Tape set name or “none”
char *	***vsn***	– VSN
int	***prot***	– fmount read-olny flag: 0-read/write, 1-read-only
int	***msgflag***	– Flag if the user’s message is valid (0 means that no message is send)
char *	***msg***	– Message from user
char *	***user***	– User login name
int	***uid***	– User UID
int	***reqtime***	– Time when the request was made

5.2.3 oper_dismount_request()

This routine should present a tape dismount request on the operator console.

Usage:

oper_dismount_request(*host*, *drive*, *tapeset*, *vsn*, *msgflag*, *msg*, *user*, *uid*, *reqtime***)**

Input Parameters:

char *	***host***	– Host name
char *	***drive***	– Tape drive name
char *	***tapeset***	– Tape set name or “none”
char *	***vsn***	– VSN
int	***msgflag***	– Flag if the user’s message is valid (0 means that no message is send)
char *	***msg***	– Message from user

char **user* – User login name

int *uid* – User UID

int *reqtime* – Time when the request was made

5.2.4 oper_mount_reply(), oper_dismount_reply()

These routines should remove a tape (dis)mount request from the operator console. It is called when another operator in the system satisfies a request, issues a reply, and the reply has successfully reached the requesting process.

Usage:

oper_mount_reply(*hostname***,** *tapedrive***,** *status***,** *message***)**

oper_dismount_reply(*hostname***,** *tapedrive***,** *status***,** *message)*

Input Parameters:

char **hostname* – Host name

char **tapedrive* – Tape drive name

int *status* – Status of the mount (-1 means tape couldn't be mounted)

char **message* – Included operator message

When an operator console replies to a mount or dismount request, it will also receive this callback after the requesting process has received the reply. With this mechanism, the operator console can recover if a (dis)mount reply is lost for some reason. For example, an operator console might reply to a tape (dis)mount request, but not remove the request from the screen until this callback routine is called.

5.2.5 oper_mount_cancel(), oper_dismount_cancel()

This callback routine will be called whenever a tape (dis)mount request is cancelled. A request can be cancelled because the requesting process crashed or exited, or because the user specifically cancelled the (dis)mount.

Usage:

oper_mount_cancel(*hostname***,** *tapedrive***,** *reason***)**

oper_dismount_cancel(*hostname***,** *tapedrive***,** *reason***)**

Input Parameters:

char **hostname* – Host name

char **tapedrive* – Tape drive name

char **reason* – Reason why the request was canceled

When the operator console receives this call, it should remove the (dis)mount request from the console and notify the operator that the (dis)mount request has been cancelled.

5.2.6 oper_add_tapedrive()

When the operator console receives this callback routine it should add the tapedrive to the display.

Usage:

oper_add_tapedrive(*hostname*, *tapedrive***)**

Input Parameters:

char ****hostname*** – Host name

char ****tapedrive*** – Tape drive name

If the operator console does not display the tape drives it services, it can regard this callback as a notification that it will be receiving mount requests for the specified tape drive.

5.2.7 oper_del_tapedrive()

When the operator console receives this callback routine it should remove the tapedrive from the
display.

Usage:

oper_del_tapedrive(*hostname*, *tapedrive***)**

Input Parameters:

char ****hostname*** – Host name

char ****tapedrive*** – Tape drive name

If the operator console does not display the tape drives it services, it can regard this callback as a notification that it will no longer be receiving mount requests for the specified tape drive.

5.2.8 oper_user_message()

This callback routine is called when a message arrives from a user process.

Usage:

oper_user_message(*userhost*, *login*, *opergroup*, *message***)**

Input Parameters:

char **userhost* – User host name

char **login* – User login name

char **opergroup* – Operator group name

char **message* – Included message

Appendix: A Release Notes

A.1 Release v3_0

A.1.1 General

OCS v3_0 is **NOT** backward compatible with OCS v2_5, v2_4, v2_3 and v2_2. Users MUST requested to relink with OCS v3_0, though user code need not change

A.1.2 Modified And New Features For Users

1. *ocs_logical_dev()* and *ocs_device_file*() no longer exists.

2. The cps tape functions (*cps_tape_dealloc(), cps_tape_mount(), cps_tape_reply()*, and *cps_tape_umount()*) supplied originally for backward compatibility have been removed.

3. OCS error message string may now be up to 256 characters long.

4. *ocs_logdev()* no longer accepts a host option of “any” and the long output has been reformatted. Valid host options are “localhost” or the name of the host. If no host option is specified, the default is still “localhost”.

5. *ocs_logdev()* and **ocs_logdev** no longer will resolve the symbolic link to a character special file if one is passed in. This was original done for backward compatibility with CPSBATCH that had symbolic in */usr/cpsdev* to the character special files. On Solaris */dev* are symbolic links to rather inscrutable paths of the character special files. Also using the character special name would make FTT and OCS have illogical base device file names. Thus this functionality is being dropped.

6. *ocs_device_type()* now checks for host strings of “any” and “localhost” like most other user callable routines do. The device type string that is returned now matches the standard product id, namely “EXB-8200”, not “exabyte_8200”.

7. OCS now supports both DLT2000 and DLT4000 tape drives. OCS no longer supports 9 track, 3480 or 3490 tape drives.

8. *ocs_devfile_set()* no longer supports attributes of “swap”, “rewind” and “retension” because these attribute are not recommended to be used through FTT. The attributes “no_swap”, “no_rewind” and “no_retension” are default. These strings are not needed, but will not produce an error. Any other attribute is assumed to be a density string suitable for the device, e.g. 8200, 8500, 10G, 20G.

9. **ocs_setdev** has an additional argument for density, namely:

> **ocs_setdev** [-h <*host*>] -t <*drive*> [-n -c -v -f <*blksiz*> -d <*dens*>]

10. Either **ocs_setdev** (from the command line) or *ocs_setdev()* (from a C or Fortran program) should be called prior to doing I/O to the device regardless of the O/S platform. In previous releases, this was only necessary on AIX.

11. *ocs_check_tape()*, *ocs_dismount*(), and *ocs_setdev()* now handle host strings of "any", '\0' and NULL like most other user callable routines do. Before "any" was treated like a valid host name whereas '\0' and NULL were treated like "localhost".

12. *ocs_list_reports()*, now checks for the host strings of "localhost", '\0' and NULL like most other user callable routines do. Before "localhost" was treated like a valid host name whereas '\0' and NULL were handled at all.

13. OCS allocate calls are storing PID and PGID of the process which allocates this drive

14. There are new *ocs_init_ftt()* and *ocs_report_ftt*() functions to use the same way as *ocs_init_stat()* and *ocs_report_stat()* for applications using ***ftt*** directly (like RBIO). This allow OCS to read and store some information directly form ***ftt*** which is not available using *devd* daemon to get statistics. Those function needs to be linked with ***ftt*** libraries by the user.

15. **ocs_update** call to compare and update new drive statistic information like vendor etc., serial number and device type.

16. New output format from **ocs_devstat**, *ocs_list_reports()* and **ocs_reports**, *ocs_report_stat()* and **ocs_report_stat**, *ocs_list_stats()* and **ocs_stats**.

17. **ocs_left_allocated** prints PID and PGID of the process which allocated drive and checks if it still exists.

18. **ocs_broken** and **ocs_repaired** (*ocs_mark_broken()* and *ocs_mark_working()*) check current drive status and report if the drive is already broken or working. This prevents marking tape drive broken twice.

19. **ocs_repaired** (*ocs_mark_replaced()*) don't need device type but it has to be able to find it based on the information from the drive.

20. **ocs_mounted** (*ocs_mounted()* for C and FORTRAN) - new function to update VSN in OCS database for tapes mounted "by hand" by user or any other way.

21. **ocs_init_stat** and **ocs_report_stat** (*ocs_init_stat()* and *ocs_report_stat()*) check if the drive is allocated to the user.

22. **ocs_check_tape** (*ocs_check_tape()*) was replaced by **ocs_check_label** (*ocs_check_label()*) with label type as a additional parameter. The function still exists but the device filename is ignored (it is not needed now) and it is calling ocs_check_label with the default label type. Interactive ocs_check_tape is now soft link to ocs_check_label

23. **ocs_deallocate** has new options:
-S - to force drive to be deallocated by admin
-c - to check if the process which allocated drive or the process with PID = PGID of the process allocated drive still exits. If it does the drive is not deallocated.
-p *<pid>* - compress given *<pid>* with one of the process which allocated drive.

24. **ocs_tape -o** *pid*: new option to show PID and PGID of the process allocated drive.

25. **xocs** “Mount Request Log” and “Tape Drive Stats Log” now includes the year and a limited number of lines may be displayed (the default is 20.)

26. *ocs_mount_request()* now ensures that the VSN is neither a NULL nor an empty string.

27. *ocs_util()* is a new function that provides allocation, tape mounts, tape replies, dismounts and deallocation through a single interface.

A.1.3 Modified And New Features For Operations

1. **xtape** lookup into Tape Vault Bin database now translates the VSN to upper case.

2. **xtape** has improved default colors.

3. **xtape** has a new option, **-rows**, that will set the number of rows on the host window to be equal to the value specified. If the **-rows** option is not specified nor **-freesize** specified, **xtape** will determine the appropriate number of rows to make the window roughly square.

A.1.4 Modified And New Features For Administrators

1. xocs no longer calls ocs_setdev when adding an AIX tape drive. The user should call this routinely and we do not do this for other OS’s.

2. OCS_ROUNDROBIN is now the default. (see **templates/ocs_local_setup.csh** and **templates/ocs_local_setup.sh**)

A.1.5 Modified And New Internal Features

1. OCS now uses the Fermi library product FTT for device file manipulation. Among other things, the DEVFILE database table no longer exists and the base name of the device file is stored in the TAPEDRIVE table as a FILENAME field. OCS v3_0 should be built with FTT v1_8 or later.

2. OCS is no longer supported on IRIX 4.0.5

3. OCS is now supported on IRIX 6.2 provided all modules and user interfaces built with the -n32 option.

4. OCS is now supported on AIX 4.1.

5. New fields in STATLOG and TAPESTAT tables for new statistic.

6. PID, PGID and PIDLOCK (not used now) fields are added to TAPEDRIVE table.

7. **xocs**, **ocs_install** and **ocs_grou**p now easily accept different Xresource values.

8. Library routines and variables that are exported, but are not intended to be called by the user, are prefixed with “ocsi_” or “fl_”.

9. OCS is now under cvs source and version control.

A.1.6 Bug Fixes

1. X-screens would subsequently core dump if the window exit/close button (rather than the dismiss button) was used to close the window

2. **ocs_terminst** and **ocs_install** sometimes would core dump while inserting users. This was due to an incorrect string length and has been fixed.

3. **ocs_terminst** now has a separate entry for just update NIS maps like **ocs_install** has.

4. **ocs_install** and **ocs_terminst** gave the appearance that multiple **dbservs** could run on the same NIS domain or same host, but the underlying functionality wasn't there. In fact, only an RPC number of 211637 would work. This has been fixed.

5. **ocs_install** and **ocs_terminst** now allow for the NIS rpc file to be the default for the particular OS which may not always be */etc/rpc*.

6. The template, **ocs**, for starting/stopping ocs on initialization now sets the # of columns to be 512 so the “ps” command won't truncate it under OSF1.

7. **/var/tmp** rather than **/usr/tmp** is used as temporary storage for sending mail messages.

A.2 Release v3_0a

A.2.1 General

OCS v3_0a is a bug release and is completely compatible with OCS v3_0.

A.2.2 Bug Fixes

1. The Motion hours, Power hours and Need Cleaning Bit are updated in the database if they are available to ocs_devstat or ocs_update. Furthermore, the Installed Motion hours and Power hours will be updated if an update is done via **xocs** if they are not already set and the update is done within 24 hours of the drive being installed.

2. **ocs_check_label** will now wait up to 30 seconds if the drive is busy before returning an error.

3. **ocs_dismount** will now wait up to 20 seconds for a drive to become only before returning an error that the drive is already offline.
4. **ocs_cleaned** and the xocs cleaning action will no longer core dump
5. **ocs_allocate** now works properly with the “must get” option. The option was being ignored when the drive was broken or an unknown drive was specified.
6. Tape drive information can now be updated via **xocs** even if the drive is allocated.
7. The tape drive information will properly be refreshed when the NEXT button is pressed in the **xocs**->DB->Tape Drive Info window.
8. The **xocs**->DB->Tape Drive Info window will no longer give an error if a drive is not

selected.

A.3 Release v3_1

A.3.1 General

OCS v3_1 is NOT backward compatible with OCS v2_5, v2_4, v2_3 and v2_2. Users MUST relink with OCS v3_1. See UPGRADE_NOTES for possible user code incompatibilities.

OCS v3_1 is backward compatible with OCS v3_0 and v3_0a. Users MUST relink with OCS v3_1, though user code need not change.

OCS v3_1 is a minor release with a new administrative function and some bug fixes. It must be built with FTT v2.0 or higher.

A.3.2 Modified And New Features For Users

1. ocs_util() can request and check label of the tape that has internal label not equal external VSN. It can also check if the tape is blank or labeled which can be useful for tapes mounted to write (v3_1).
2. Devd calls ftt_setdev instead of ftt_open_dev which doesn’t wait for drive to be ready. This requires ftt v2_0 or up to build (v3_1).
3. Support for DLT700 and AIT-1 tape drives has been added.

A.3.3 New Features For Operations

1. xtape has an additional sort option on the host window to sort the mount requests by tape vault bin number,
2. The print menu item from on the main xtape window will now print the mount requests by tape vault bin number rather than the order that the mounts requests were sorted in the individual host windows.

3. A new command ,ocs_all_pending, will display all pending mount requests to standard output.
4. ocs_mrlog now has command line options similar to ocs_reports and will display the mount request log to standard output.(v3_1a)
5. error messages from failed mount now contain the tape drive and vsn. (v3_1a)

A.3.4 New Features For Administrators

1. OCS CPU Status. It is now possible to set the CPU status on an OCS node. This feature allows a given node's status to be logically set "up" or "down" to OCS. Tape drives can only be allocated on a node that has it's CPU status set to "up". This can be used to take a node out of service from OCS without "marking" all of the tape drives on the node as "broken".
   The new OCS command OCS_CPU_STATUS will list or change the CPU status of a node. This functionality is also included in the XOCS graphical interface. See the OCS V3_1 Reference Guide. There is also a callable "C" and Fortran program function called OCS_CPU_STATUS.
   OCS administrator authorization is required to use OCS_CPU_STATUS.
2. The option of changing basename for drive was added to the xocs->DB->Tape Drive window.
3. Ocs_dismount will allow anyone to attempt to dismount a tape if the tape drive is not allocated and will allow root to dismount a tape regardless of who has it allocated.

A.3.5 Modified And New Internal Features

1. OCS v3_1 must be built with FTT v2_0.
2. OCS is now supported on IRIX 6.2 provided all modules and user interfaces built with the -n32 option.

A.3.6 Bug Fixes

1. Errors corrected in ocs_ctest and ocs_btest from examples.
2. After deleting a host in xocs, the host info window would retain the host name and attempt to retrieve the host information from the database, which would cause an error. All host info window fields will be cleared after a host is deleted.
3. The "Mail Address - Service Only" field of the xocs host info window will default to "None" when adding a new host if no information is provided.
4. Xtape had unitialized varilables which caused the columns/row algoritm to fail sometimes. (v3_1a)
5. ocs_deallocate as an administrator would deallocate the drive without checking for mount requests (v3_1a)

6. the taped would hang if more than 10 connections were established within 1 second (v3_1a)

Appendix: B Upgrade notes

B.1 Upgrading To v3_1 from v2_*

This upgrade procedure works for versions v2_5, v2_4a, v2_4, v2_3 and v2_2.

1. Build and distribute OCS v3_1 to appropriate *taped* and *dbserv* hosts. If a host does not require the source, then the executable files, libraries and documentation need only to be distributed to that host, e.g.:

> **rdist -c**./{ABSTRACT,LICENSE,README,ORIGIN,SUPPORT,\
> ups,bin,lib,include,doc,examples,templates} \
> products@cdfsga:/usr/products/ocs/v3_1

Make sure *taped*, *devd*, *dbserv* are owned by root.sys. *Taped* and *devd* must have setuid bit on as well, e.g.

> **chown** root.sys *taped devd dbserv*
> **chmod** ug+s *taped devd*

2. Backup OCS old database:

a) Reduce the STATLOG to a reasonable number of records, e.g.[1]

> **ocs_reports** -o delete -b 05/01 > stats.0596

b) Reduce the MOUNTLOG to a reasonable number of records, e.g.[2]

> **ocs_xfer** -t mountlog -b 05/01 > mounts.0496

c) Do actual backup, e.g.

> **ocs_backup** > backup.old

d) Make sure that the old ocs_backup is used and that the exit status is 0.

3. Render old version of OCS useless on taped and dbserv hosts:

a) undeclare it in ups, e.g.

> **setup** ups
> **ups** undeclare -z $PRODUCTS ocs v2_5[3]

b) kill *dbserv*, *taped* and *xtape*

c) move/rename the v2_5 directories and database, e.g:

> **mv** /usr/products/ocs/v2_5 /usr/products/ocs/v2_5.old
> **mv** /usr/products/ocs/DB /usr/products/ocs/DB.v2_5
> **mkdir** /usr/products/ocs/DB

4. Declare OCS v3_1 current on taped and dbserv hosts, e.g:

1. This is not necessary, but not all of the information can be translated to the new format
2. This is not necessary.
3. or whatever version is running

setup ups
ups declare -c -z $PRODUCTS -r /usr/products/ocs/v3_1 ocs v3_1

5. Setup ocs v3_1 and make sure that new ocs executable files are now used

6. Bring up **ocs_install** as root on the dbserv host. If the pathname for the *dbserv* and *taped* binary directory has the version in the pathname (e.g. v2_5):

- **a)** change dbserv and taped to use v3_1 rather than v2_5 (press OK buttons)
- **b)** press *taped* and *dbserv* install button
- **c)** press create db tables button

Otherwise, exit **ocs_install** and either:

- **a)** modify the symbolic links to the binary directory to point to the OCS v3_1 binary directory or wherever the OCS v3_1 *dbserv* and *taped* will run.
- **b)** copy the OCS v3_1 *dbserv*, *taped* and *devd* executable files to the *dbserv* and *taped* binary directories.

7. Convert the OCS v2_5 DB backup to the OCS v3_1 DB format and populate the OCS v3_1 database, e.g.:

ocs_db_cnvrt < backup.old > backup.v3_1
sqlint < backup.v3_1 (check exit status)

Verify that the exit status is zero.

8. Bring up **ocs_install** as root on each *taped* host. If the pathname for the *taped* binary directory has the version in the pathname (e.g. v2_5):

- **a)** change *taped* to use v3_1 rather than v2_5 (press OK)
- **b)** press install taped

Otherwise, exit **ocs_install** and either:

- **a)** modify the symbolic links to the binary directory to point to the OCS v3_1 binary directory or wherever the OCS v3_1 *taped* will run.
- **b)** copy the OCS v3_1 *taped* and *devd* executables to the *taped* binary directory

9. As a user other than root, execute: (make sure that new executable is used)

ocs_tape

on the *dbserv* host and all *taped* hosts. (Make sure that the new executable is used.) If you get an error, check that steps 1 through 8were done correctly.

10. Verify that an appropriate OCS initialization script is installed on each *taped* and *dbserv* host. On AIX, the script is usually called */etc/startocs* and is called from *rc.local*. On IRIX the script is usually called */etc/init.d/ocs* and has links to it from */etc/rc0.d* and */etc/rc2.d*. See templates/startocs and templates/ocs for examples. Typically, **ocs_reboot** will need to be copied to the same binary directory in which the *devd*, *taped* and *dbserv* were placed.

11. Fix any tape drive information using **ocs_update**. This utility checks and updates new drive information as well as corrects serial number and device type.

a) For help, first run:

ocs_update -?

b) To see the information that needs updating, for each tape drive run:

ocs_update -t *<drive>*

c) To correct the information for a particular tape drive, run:

ocs_update -t *<drive>*

12. Bring up **xtape**(s) and run out a test mount.

B.1.1 Possible V2_* TO v3_* User Code Incompatiblies

Users upgrading from v2_* to v3_* should check their code for the following possible incompatibilities:

1. *ocs_logical_dev()* and *ocs_device_file()* no longer exists.
2. OCS error message string may now be up to 256 characters long.
3. *ocs_log_dev()* and **ocs_log_dev** no longer will resolve the symbolink link to a character special file if one is passed in.
4. *ocs_device_type()* now returns the device type string that matche the standard product id, namely “EXB-8200”, not “exabyte_8200”.
5. *ocs_devfile_set()* no longer supports attributes of “swap”, “rewind” and “retension” because these attribute are not recommended to be used through FTT.
6. Either **ocs_setdev** (from the command line) or *ocs_setdev()* (from a C or Fortran program) should be called prior to doing I/O to the device regardless of the O/S platform. (In previous releases, this was only necessary on AIX.)
7. *ocs_check_tape()* and **ocs_check_tape** are superseded by *ocs_check_label()* and *ocs_check_label()* with label type as a additional parameter (e.g. “ansi”, “cpio”, “tar”). The old function and interactive command still exists but may be removed in future releases.

B.2 Upgrading To v3_1 from v3_0*

1. Build & distribute OCS v3_1 to appropriate taped and dbserv hosts. If a host does not require the source, then the executables libraries and documentation need only to be distributed to that host, e.g.:

```
rdist -c./{ABSTRACT,LICENSE,README,ORIGIN,SUPPORT,\
    ups,bin,lib,include,doc,examples,templates} \
    products@cdfsga:/usr/products/ocs/v3_1
```

Make sure *taped*, *devd*, *dbserv* are owned by root.sys. *Taped* and *devd* must have setuid bit on as well, e.g.

chown root.sys taped devd dbserv
chmod ug+s taped devd

2. Render old version of OCS useless on taped and *dbserv* hosts:

a) optionally, undeclare it in ups, e.g.

setup ups
ups undeclare -z $PRODUCTS ocs v3_0*

b) kill *dbserv*, *taped*(s) and *xtape*(s)

c) optionally, remove the v3_0* directories, e.g:

rm -rf /usr/products/ocs/v3_0*

3. Declare OCS v3_1 current on *taped* and *dbserv* hosts, e.g:

setup ups
ups declare -c -z $PRODUCTS -r /usr/products/ocs/v3_1 ocs v3_1

4. Setup ocs v3_1and make sure that new ocs executables are now used

5. Bring up **ocs_install** as root on the *dbserv* host. If the pathname for the dbserv and taped binary directory has the version in the pathname (e.g. v3_0*):

a) change *dbserv* and *taped* to use v3_1 rather than v3_0* (press OK buttons)

b) press taped and dbserv install button.

Otherwise, exit **ocs_instal**l and either:

c) modify the symbolic links to the binary directory to point to the OCS v3_1 binary directory or wherever the OCS v3_1 *dbserv* and *taped* will run.

d) copy the OCS v3_1 *dbserv*, *taped* and *devd* executables to the *dbserv* and *taped* binary directories.

6. Bring up **ocs_install** as root on each *taped* host. If the pathname for the *taped* binary directory has the version in the pathname (e.g. v3_0*):

a) change *taped* to use v3_1 rather than v3_0* (press OK)

b) press install taped

Otherwise, exit **ocs_install** and either:

c) modify the symbolic links to the binary directory to point to the OCS v3_1 binary directory or wherever the OCS v3_1 *taped* will run.

d) copy the OCS v3_1 *taped* and *devd* executables to the taped binary directory

7. As a user other than root, execute:

ocs_tape

on the *dbserv* host and all *taped* hosts. (Make sure that the new executable is used.) If you get an error, check that steps 1 through 6 were done correctly.

8. Fix any tape drive information using **ocs_update**. This utility checks and updates new drive information as well as corrects serial number and device type.

   a) For help, first run:

   ocs_update -?

   b) To see the information that needs updating, for each tape drive run:

   ocs_update -t <drive>

   c) To correct the information for a particular tape drive, run:

   ocs_update -t <drive> -o update

9. Bring up **xtape**(s) and run out a test mount.

Appendix: C FTT in OCS

Ftt is used by OCS as a lower level package to do all operation on devices. There is no OCS calls to access device files only ***ftt*** calls are used. Since all of the device operation are done by device daemon *devd* the user code doesn't have to be linked with ftt package unless *ocs_init_ftt()* and *ocs_report_ftt()* functions are used.

C.1 FTT Drive Statistic vs OCS

All information from the drive is done by the **ftt** statistic function. The Table 1 on page 97 shows all available information and which of the fields are available in OCS v2_5 and current version.

TABLE 1. FTT statistic vs. OCS tables

Name	Data type	In	Description	OLD[A]	LOG[B]	STAT[C]	SHOW[D]	NO[E]	Notes
FTT_VENDOR_ID	string	dev	SCSI Inquiry vendor id data			X	X		
FTT_PRODUCT_ID	string	dev	SCSI Inquiry product id			X	X		
FTT_FIRMWARE	string	dev	SCSI Inquiry firmware rev level			X	X		
FTT_SERIAL_NUM	string	dev	Device serial number	X	X	X	X		
FTT_POWER_HOURS	counter	dev	Number of hours device has been on			X	X		undef for 8500
FTT_CLEANING_BIT	flag	dev	Cleaning bit	X	F	X	X		undef DLT4000
FTT_READ_COUNT	counter	dev /ftt	Decimal number of blocks read since rewind/reset (in Kbytes)		X	X	X		To calculate Kbytes by device
FTT_WRITE_COUNT	counter	dev /ftt	Decimal number of blocks written since last rewind/reset (in Kbytes)		X	X	X		
FTT_READ_ERRORS	counter	dev	Read errors	X	X	X	X		STAT has only
FTT_WIRTE_ERRORS	counter	dev	Write errors	X	X	X	X		one
FTT_DENSITY	numeric	dev	Mode sense tape density					N	
FTT_READ_COMP	numeric	dev	Compression factor for read		X		X		Use it for to find
FTT_WRITE_COMP	numeric	dev	Compression factor for write		X		X		if the tape was compressed
FTT_FILE_NUMBER	numeric	dev	File number from BOT					N	
FTT_BLOCK_NUMBER	numeric	dev	Block number in current file					N	
FTT_BOT	flag	dev	at beginning of the tape	X	F		X		
FTT_READY	flag	dev	Device ready status		F		X		
FTT_WRITE_PROT	flag	dev	Write protection flag	X	F		X		
FTT_FMK	flag	dev	At filemark	X			X		
FTT_EOM	flag	dev	At end of media (early warning)	X			X		
FTT_PEOT	flag	dev	At physical end of tape	X	F		X		
FTT_MEDIA_TYPE	numeric	dev	SCSI media type				X		

TABLE 1. FTT statistic vs. OCS tables

Name	Data type	In	Description	OLD[A]	LOG[B]	STAT[C]	SHOW[D]	NO[E]	Notes
FTT_BLOCK_SIZE	numeric	dev	Tape block size				X		
FTT_BLOCK_TOTAL	numeric	dev	Number of blocks on tape				X		
FTT_TRANS_DENSITY	numeric	dev /ftt	Translated density			X	X		
FTT_TRANS_COMPRESS	numeric	dev /ftt	Translated compression			X	X		
FTT_REMAIN_TAPE	counter	dev	Remaining KB of tape	X			X		used to calculate KB_DEV
FTT_USER_READ	counter	ftt	Number of KBytes read since open/unload		X	X			Use this as a KB user for RBIO (**ftt** only)
FTT_USER_WRITE	counter	ftt	Number of KBytes written by user since open/unload		X	X			
FTT_CONTROLLER	string	dev	Controller type			X	X		
FTT_ILI	flag	dev	SCSI ILI bit from Req Sense	X			X		
FTT_SCSI_ASC	numeric	dev	SCSI additional sense code				X		
FTT_SCSI_ASCQ	numeric	dev	SCSI additional sense code qualifier				X		
FTT_PF	flag	dev	SCSI PF flag	X	F		X		
FTT_CLEANED_BIT	flag	dev	SCSI cleaned bit	X	F		X		undef for DLT
FTT_TRACK_RETRY	numeric	dev	Track retry error		X	X	X		undef fro DLT and 8500
FTT_UNDERRUN	numeric	dev	stop/start counter		X	X	X		undef fro DLT and 8500
FTT_MOTION_HOURS	counter	dev	hours in motion			X	X		DLT
FTT_TUR_STATUS	numeric	dev	SCSI test Unit Ready response	X			X		
FTT_COUNT_ORIGIN	string	ftt	Origin of read/write KBytes counter		X		X		one character only
FTT_N_READS	counter	ftt	Number of read calls					N	only via **ftt**
FTT_N_WRITES	counter	ftt	Number of write calls					N	only via **ftt**
FTT_TNP	flag	dev	Exabyte tape “not present” flag	X	F		X		
FTT_SENSE_KEY	numeric	dev	SCSI request sense key					N	
FTT_TRANS_SENSE_KEY	string	dev	Above translated to string	X			X		
FTT_RETRIES	num	ftt	**ftt** retries		X				only via **ftt**
FTT_FAIL_RETRIES	num	ftt	**ftt** retries which failed						only via **ftt**
FTT_HARD_ERRORS	num	ftt	Number of hardware errors		X				only via **ftt**
FTT_RESETS	flag	ftt	SCSI reset flag		X				only via **ftt**

a. fields existing in the OCS v2_5

b. included in the STATLOG table (F - means bart of the flags field)

c. included in the TAPESTAT table

d. visible by the ocs_devstat

e. Not useful for the OCS statistic

C.2 Collecting Drive Statistic in OCS

In the previous version of OCS all access to the drive was done via **devd** (device daemon). Following is the list of OCS functions and shell calls to show and collect statistic from device using **devd** process:

- ocs_devstat - shows all available information
- ocs_init_stat - gets statistic from device and store some of them in OCS database
- ocs_report_stat - gets statistic from device and updated STATLOG and TAPESTAT tables using information stored in database by the initial function to calculate delta

There new pair of OCS statistic functions design to use in RBIO or other products using ***ftt*** which are getting the statistic directory from device not via *devd*. Those functions have to be linked with ***ftt*** by the user.

New OCS functions to collect statistic:

- *ocs_init_ftt()* - takes ftt_descriptor and initialize ***ftt*** statistic buffer and stores some of the information in the OCS DB
- *ocs_report_ftt()* - takes ftt_descriptor and gets final statistic from ***ftt*** and updates OCS database tables

There is verification of the reports added to all OCS statistic functions:

- verification of allocation
  - drive have to be allocated
  - the same UID
- the I_VALID variable from the TAPESTAT table is used to check if the information stored is valid and can be used to collect statistic and add them to OCS database
  - I_VALID = 1 set in the *ocs_init_stat()* call and then checked in the *ocs_report_stat()* function
  - I_VALID = 2 set in the *ocs_init_ftt()* call and verified in the *ocs_report_ftt()* function
  - I_VALID = 0 set in the *ocs_deallocate()*, *ocs_report_stat()*, and *ocs_report_ftt()*
- There are a few new errors returned by the those functions
  - OCS_NOT_ALLOCATED = when the drive is not allocated
  - OCS_PERMISION_DENIED= when the drive is allocated to different user
  - OCS_STAT_ERROR=
    - by *ocs_init_stat()* and *ocs_init_ftt()* when the statistic was already initialized (I_VALID nor equal 0)

-by *ocs_report_stat()* if the statistic was initialized using *ocs_init_ftt()* function

-by *ocs_report_ftt()* if the statistic was initialized using *ocs_init_stat()* functions

- For other errors the statistic will be collected with information about the error coded in the VALID field:
  - U = when the *ocs_report_stat()* or *ocs_report_ftt()* was called without the initial call first
  - I = when getting statistic failed during initial call
  - R = when getting statistic failed during the report call

Appendix: D Changes in the OCS database

The DEVFILE table is removed from the DB and TAPEDRIVE, TAPESTAT and STATLOG tables have some fields added visible in the following tables as rows with grey background color.

D.1 Release v2_5 vs v3_0

The grey area in the following tables marks fields that are changed.

D.1.1 TAPEDRIVE table

TABLE 2. TAPEDRIVE table

Name	Data type	Description	Notes
DEVNAME	string	Drive name	
HOSTNAME	string	Host name	
FILENAME	string	Base device name	
MEDIA	string	Media Name	DELETE
DEVTYPE	string	Device Type	
STATUS	string	Status of this device	
VSN	string	VSN from the mount request	
PROTECTED	int	Write protection	
ALLOCTIME	date	Date of the last allocation	
ALLOCATED	int	Allocated flag	
LOGIN	string	Login name of the user who allocated drive	
UNIXUID	int	UID of the user who allocated drive	
PID	int	PID of the process which allocated drive	
PGID	int	PGID of the process which allocated drive	
PIDLOCK	int	Lock flag	(not used[a])

a. The PIDLOCK won't be used for the next release. The PID and PGID locking for allocation needs some more testing and discussions.

D.1.2 TAPESTAT table

TABLE 3. TAPESTAT table

Name	Data type	Description	REP[A]	UP[B]	STAT[C]	Notes
DEVNAME	string	Device name				
HOSTNAME	string	Host				
SERIAL	string	Device serial number	X	X		install time
VENDOR	string	Vendor ID	X	X		install time
PRODUCT	string	Product ID	X	X		install time
FIRMWARE	string	Firmware	X	X		install time
CONTROLLER	string	Controller	X	X		install time
POWER	int	Number of hour ON	X		X	current
MOTION	int	Number of hours in motion	X		X	current
INSTALLTIME	date	Installation date	X			
CLEANTIME	date	Cleaning date	X			
CLEANINGS	int	Number of cleanings				
POWER_I/C	int	Number of hour ON at time it was installed (cleaning time)	X			install time (cleaning time)
MOTION_I/C	int	Number of hours in motion at the time it was installed (cleaning time)	X			install time (cleaning time)
TRACK_I/C	int	Track retry error at time the drive was installed (cleaned)			Σ	
UNDERRUN_I/C	int	Stop/start counter at time the drive was installed (cleaned)			Σ	
MOUNTS_R_I MOUNTS_W_I MOUNTS_R_C MOUNTS_W_C	int	Number of mounts for read (write) after it was installed (last cleaned)			Σ	
REPORTS_R/W_*I/C	int	Number of reports			+1	
MB_USR_R/W_I/C	int	Number of MBytes reported by user			Σ	
MB_DEV_R/W_I/C	int	Number of MBytes reported by device			Σ	
ERRORS_R/W_I/C	int	Number of errors			Σ	
CLEANIT	int	Clean bit			X	
I_VALID	int	Valid flag of all initial values				not used in v2_5
I_TAPELEFT	int					DELETE
I_ERRORS	int					DELETE
I_BLOCKS	int					DELETE
I_FLAGS	int					DELETE
I_MODE	string					DELETE
I_ERROR_R	int	Initial read errors counter				

TABLE 3. TAPESTAT table

Name	Data type	Description	REP[A]	UP[B]	STAT[C]	Notes
I_ERROR_W	int	Initial write errors counter				
I_COUNT_R	int	Initial read count (Kbytes)				
I_COUNT_W	int	Initial write count (Kbytes)				
I_TRACK_RETRY	int					
I_UNDERRUN	int					

a. When the new drive is added or the drive is replaced. All other fields are reset at this time

b. Information can be updated without drive change

c. Set (marked as X) or add (marked as Σ) values at OCS report time

D.1.3 STATLOG table

TABLE 4. STATLOG table

Name	Data type	Description	Notes
DEVNAME	string	Name of device	
HOSTNAME	string	Host name	
SERIAL	string	Device serial number	
REPORTTIME	date	Date of the report	
LOGIN	string	User name	
VALID	string	Valid flag (see page 100)	
VSN	string	VSN	
OPERATION	string	read/write operation from user	
KB_USER	int	KBytes read/write reported by user	
KB_DEV	int	KBytes reported by device - difference in tape position	
ERRORS	int	number of errors per kilobyte block calculated by device	
MODE	string	8200,8500,8500c	DELETE
DENSITY	string	Density - 8200, 8500...	
COMPR	string	compression “c”	
COMP_RATIO	int	Compression ratio	
ORIGIN	string	First character of the FTT_COUNT_ORIGIN	
FLAGS	int	(see flags table)	The whole flags table is changed to have two bits per one field
TRCK_RETRY	int	Track retry error	
UNDERRUN	int	stop/start counter	
FTT_RETRIES	int	Number of read/write retries	only **ftt** direct call
FTT_FAIL_RET	int	Number of hardware errors	only **ftt** direct call

TABLE 5. Old Flags

Digit	Bit 3	Bit 2	Bit 1	Bit 0
0	Error Counter Overflow	Tape Motion error	Tape Not Present	BOT
1	Power Fail	SCSI bus Parity Error	Formatted Buffer Parity Error	Media Error
2	Under Run Error	Write Error 1	Servo System Error	Formatted Error
3	Reserved	Tape Mark Detected Error	Write Protect	Filemark Error

TABLE 5. Old Flags

Digit	Bit 3	Bit 2	Bit 1	Bit 0
4	Cleaning	PEOT	Write Splice Error (blank)	Write Splice Error (position)
5	Reserved	Reserved	Reverse Retries Required	Cleaned
6	FMK	EOM	ILI	Reserved

TABLE 6. New flags

Digit	Digit 1 Bits 0-1	Digit 1 Bits 2-3	Digit 2 Bites 0-1	Digit2 Bites 2-3
1-2	TNP	WriteProt	Clean	Cleaned
3-4	BOT	PEOT	PF	Ready
5-6	FTT_resets			

The new flags filed is using two bits per each field:

01- for it the bit is set

00- if the bit is not set

10- if the field is not available

Index

R

S

T

U

V

X