XRootD

File Residency Manager Reference

23-June-2020
Andrew Hanushevsky

1   Introduction..5

2   frm_admin Command 17

3   The frm_purged Command for Purging.............................41

4   frm_xfrd Command for the Transfer Daemon 51

1 Introduction

This document describes File Residency Manager (**FRM)** commands and configuration directives. **FRM** was designed to enhance the Open Storage System (**oss**) scalable file system, an **XRootD** feature. **FRM** can be used to administer the local file system (i.e., placing files and maintaining symbolic links to these files), and for copying files to and from other servers, including a Mass Storage System. In effect, **FRM** provides the infrastructure to fully implement Multi-Tiered Storage.

The **FRM** components are fully configurable so that virtually any file transfer mechanisms can be used. The **FRM** configuration is described by a configuration file. This configuration file must hold the configuration directives describing the **XRootD** configuration as well as a few **FRM** specific directives.

Using the **oss** component does not require that you use **FRM**. In fact, there is no reason to use **FRM** if you do not need Multi-Tiered Storage, do not need server-to-server copy functionality, nor have enabled scalable file system support using the **oss.space** configuration directive.

1.1 FRM Components

The **FRM** consists of three configurable components:
File administration via the **frm_admin** command,
- Inactive file purging via the **frm_purged** daemon. and
- File Transfer via the **frm_xfragent** command and the **frm_xfrd** daemon.

The **FRM** deals with two major mechanisms:
- local disk, and
- remote servers (e.g., Mass Storage System) that implement a Unix-like name space, a file transfer mechanism, and interface to supply metadata about stored files.

1.1.1 File Administration Using frm_admin

The **frm_admin** command allows you to manipulate the file space and file attributes for file accessed via **XRootD** on a particular server. Hence, it can only be run on an actual data server and requires that you use the standard underlying **XRootD**-

provided mechanisms to access the underlying local file system. If this is not the case, the some **frm_admin** functions may not be available.

To properly work, the command scans your **XRootD** configuration file to determine how the data server is configured. Once this occurs you can

- Verify that the name space matches the data space, that usage statistics are correct, and make any needed repairs if they are inconsistent (see **audit**)
- Calculate, reset, and vie file checksums (see **chksum**),
- Find files that have (e.g. a memory mapped or pinned) or lack (e.g. are not check-summed or migrated) certain attributes (see **find**),
- Designate one or more files as migratable or purgeable (see **mark**),
- Designate one or more files are file as capable of being memory mapped or remove such designations (see **mmap**),
- Designate one or more files are non-purgeable for a variable amount of time or remove such designation (see **pin**),
- Display various information about the name and data spaces, transfer queues, and perform name conversions (see **query**),
- Relocate files from one partition to another (see **reloc**), and
- Remove files from the data server and automatically remove any references to the files from redirector caches (see **rm**).

All of these functions can perform properly only when the total configuration exists in a single file. This is because **frm_admin** is a cross-component command and needs to determine how the components interact to provide a seamless view.

The **frm_admin** command can run as a command line tool (i.e. a single command specified on the command line), as an interactive command (i.e., prompting for sub-commands), and in batch mode (i.e. piping multiple commands to its standard input while in interactive mode). The command always exits with a zero status code if no errors occurred.

To make administration easier, the **frm_admin** command can run on a live system. There is no need to stop the data server during its execution.

1.1.2   File Purging Using frm_purged

The **XRootD** system allows you to setup data servers in file caching mode, also known as staging. In this setup, you bring in files as a clients request them, either in real-time or pre-run mode. Because the data server is constantly ingesting files the

File Residency Manager provides a tool that allows you to remove files that are no longer popular (i.e., not referenced for some time). This is accomplished by running the **frm_purged** daemon.

This daemon perform the following steps:
- Reads the **XRootD** configuration file
  - looking for exported paths that have been designated as purgeable (e.g. "**all.export** *path* **purge**"),
  - determine the purge policy (i.e., when a file should be purge) for each configured space (i.e. **dirhold**, **policy**, **polprog** directives), and
  - how often to attempt purging (i.e. **waittime** directive).
- It scans all purgeable paths and constructs a list of purgeable candidate files, ordering them in least used order.
- At each purge interval, if the amount of free space is below the desired level, candidates files still eligible for purging are removed (subject to any real-time policy as implemented by the **polprog** directive) until there is sufficient free space available.
- The above step is repeated until no candidate files remain. In this case, a new candidate list is constructed (i.e., step 2 above).

While it is normal to run **frm_purged** as a daemon, you can also run it as a one-time command (see the **–O** command line option). This allows you to make additional space available on an ad hoc basis. When you to run **frm_purged** as a one-time command and you are also running it as a daemon, you should pause the daemon during the one-time run. You can do this in a number of ways (e.g., sending it a SIGSTOP kill signal or temporarily creating a stop file). While not absolutely necessary, it does keep the log file clean of false error messages as the two processes compete for purgeable files.

With the **–T** command line option you can see which files would be removed without actually removing the files. This is useful if you wish to verify that you have correctly specified various purge options.

A file is deemed purgeable when all of the following conditions are true:
- resides in a purgeable path,
- has not been accessed for a specified amount of time,
- is not pinned (see the **frm_admin pin**),
- has been migrated if it exists in migratable space and has been modified (see **Migrating Files**), and
- the specified real-time policy, if any, allows the file to be purged.

As files are removed, it is likely that empty directories are created. Normally, **frm_purged** keeps empty directories for 40 hours and then removes them. You can specify a longer or shorter time using the **dirhold** directive.

1.1.3   Transfering Files Using frm_xfrd

The **frm_xfrd** command runs as a daemon and co-ordinates copying files out of and into a data server. The daemon, itself, does not actually copes files but instead relies on site-supplied scripts to perform the actual transfer operation. This allows you to use any copy command and any source or target location.

The notion of copying files out of a data server is known as migration. It allows you to make sure that modified files are backed up on more permanent media. Migration is completely optional.

The notion of copying files into a data server is known as staging. It allows you to automatically fetch missing files into a data server from another local (e.g. MSS) or global (e.g. remote server) location.

1.1.3.1   Migrating Files

When **frm_xfrd** is started, it reads the **XRootD** configuration file looking for paths that have been designated as migratable (i.e. "**all.export** *path* **migrate**"). If it find no such paths, the migration component of **frm_xfrd** is disabled and migration activity occurs. Migration activity can also be disabled, even though migratable paths exists, if you have not specified how files are to be copied out of the server using the **copycmd** directive with the **out** attribute.

Otherwise, the daemon sets up an internal process to look for files that should be migrated. In some sense, the use of the word "migrate" is historic and rather inaccurate. Historically, migration meant that the file is merely copied to another location but not necessarily removed (i.e. purged).

For each migratable path, the daemon constructs a candidate list of files to be copied out of the server. A file is deemed migratable if all of the following are true.
- It has not be accessed for the time specified by the **idlehold** directive (default is 10 minutes) and
- it has been modified but not yet been successfully copied out of the server.

For each file in the candidate list, the transfer command specified by the "**copycmd out**" directive is invoked to copy the file. If the copy is successful, the file is marked as migrated and will not be migrated again unless it is modified. If the copy fails, the copy will be retried on the next construction of a candidate list.

Since transfers can take a long time, **frm_xfrd** rechecks if a file is still eligible for migration prior to invoking the copy command.

Candidate lists are periodically constructed. The time interval is specified by the **waittime** directive which, by default, is 60 minutes. To avoid over-loading the server, a maximum number of transfer are allowed to occur at the same time. This is specified by the **copymax** directive (default is 2). Since migration competes with staging, **frm_xfrd** employs a fair share algorithm to prevent starvation of either activity.

You can manage migration using the **frm_admin** command with two subcommands:
- **find** to display eligible files that have not yet been migrated, and
- **mark** to force a file to be migrated or to cancel a pending migration.

You can always pause **frm_xfrd**'s migration component by creating a special stop file named **STOPMIGR**.

1.1.3.2   Staging Files

The **frm_xfrd** may also co-ordinate requests to copy files into the data server. This mechanism is known as staging. It occurs automatically for stageable paths (i.e., "**all.export** *path* **stage**") and it is possible to manually request that a staging operation occur using the **frm_xfragent** command.

Staging is driven by a disk-based queue. Requests may be added to the queue by the **oss** component of **XRootD** or by the **frm_xfragent** command. The queue can be listed using **frm_admin** **query xfrq** command. The control flow is shown in the following diagram.
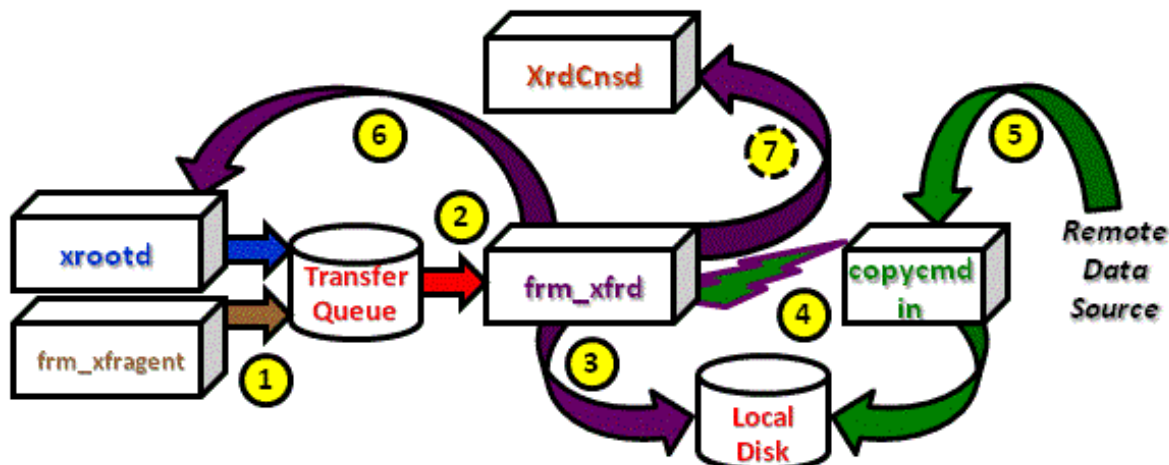
The relevant steps are

1. The **XRootD** daemon and **frm_xfragent** command adds stage-in requests to the transfer queue.
2. The **frm_xfrd** daemon reads these requests from the queue.
3. Then it normally pre-allocates a null file to mark where the file is to be placed on local disk. The pre-allocation occurs just as if the file were created via **XRootD** less any security checks since these would have been performed prior to placing the request in the transfer queue.
4. After possible file pre-allocation **frm_xfrd** launches (i.e. fork/exec) the program identified in the **copycmd** directive with the **in** attribute. The program is passed the data source and destination, plus additional information, as configured.
5. The copy program is responsible for copying the data from the remote source to the local destination, overlaying the pre-allocated file (i.e. truncate/copy).
6. For **XRootD** placed entries, **frm_xfrd** notifies **XRootD** whether the copy succeeded or failed.

The transfer queue is a set of files located in the directory identified by the **adminpath** directive (default is /tmp); possible modified by the instance name (**-n** command line option). Normally, there is a 1- to 1-correspondence between an **XRootD** instance and a transfer queue. However, you can use the **qcheck** directive to establish a single transfer queue for any number of **XRootD** instances.

The transfer queue is a set of files because the queue keeps track of requests to stage files, **frm_xfragent** initiated requests to migrate files, and **XRootD** 3rd copy file requests. Each of these requests can be individually paused using special *stop* files. Also, each request can be given a low, medium, or high priority. High priority requests execute three times as often as low priority requests; medium priority requests twice as often. By default, all requests are placed in the low priority queue.

The copy command can be any command you desire, including a script that can make arbitrary decisions on how to accomplish the copy. In all cases, the copy command must end with a 0 status code is the copy was successful. A status code of 2 if the source file was not found, and any non-zero status code for any other type of error.

The **frm_xrfd** daemon maintains numerous variables holding information to pass to the command. Perhaps the two most important variable are **$SRC** which contains the source path and **$DST** which contains the destination path. For instance, the directive

```
frm.xfr.copycmd in xrdcp root:/redirhost/$SRC $DST
```

Would copy a file whose name is contained in **$SRC** from an **XRootD** cluster control by the redirector *redirhost* to the local disk at **$DST**.

1.1.3.3   Staging Files Across A Firewall

For many sites, staging causes data to be copied from some internal source (e.g., slow file server, Mass Storage System, etc). When you wish to stage from an external source (e.g. a globally federated cluster) then the server needs to be able to establish an outgoing connection on the public internet. If there is a firewall that disallows out-going connections, then additional you need to bridge the firewall. The usual accepted solution to such a problem is to setup a border machine that has connectivity to the external internet as well as the local network. This becomes the proxy host and the actual copy must occur on the border machine since it is the only machine that has access to the inside and outside worlds.

This situation is illustrated in the following diagram.
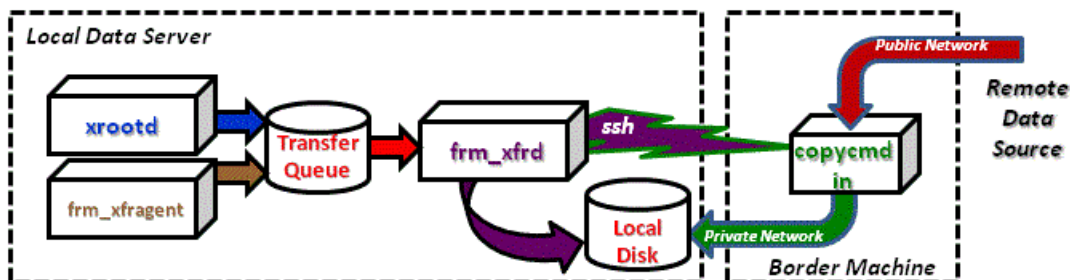
The trick is to execute the copy command on the border machine. This can be easily accomplished by simply using ssh. For instance,

```
frm.xfr.copycmd in ssh border copycmd
```

Of course, you would need to setup ssh host keys to prevent **ssh** from prompting for a password (see **ssh-keygen**).

The only problem here is that the copy command needs to be able to access the local disk that resides at the launching host. Some copy programs can do this but not all. Notably, **xrdcp** cannot be used to directly access the local disk from a border machine. There are two solutions to this problem.

1.1.3.3.1   SSH Tunneling

This solution simply sends the data from fetched by **xrdcp** back to the launching host via the established **ssh** tunnel and directs it to the destination file. The basic concept is shown below along with the **copycmd** directive that would be specified.

frm.xfr.copycmd in /bin/sh −c "ssh border `wallhop $SRC' > $DST"
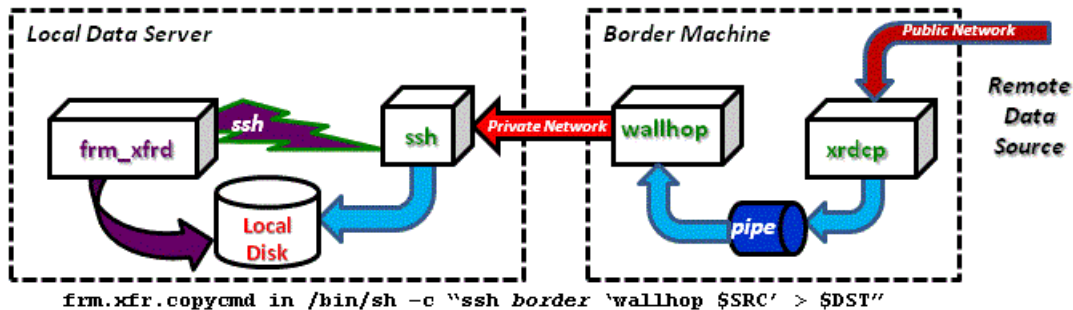
The **wallhop** script, running on the border machine, simply creates a local pipe to capture data from **xrdcp** and sends it back across the private network to the originating host where it gets piped into the destination file. The **wallhop** script is so simple that it is shown below.

```perl
#!/bin/perl
$infn = $ARGV[0];           # Input source
$fifo = "/tmp/.wallhop.$$";  # The fifo we will be using

print STDERR `mkfifo $fifo`; # Create the temporary fifo

# Copy fifo to standard out (i.e. source file -> stdout)
#
exec("/bin/cat $fifo") if !($pid = fork());

# Copy source file to the fifo, delete the fifo, and exit
#
$resp = `xrdcp -f -np $infn $fifo`;
kill(KILL,$pid) if $rc = $?;
unlink($fifo);
exit $rc;
```

The wallhop Script

1.1.3.3.2   Reconnection

The second solution avoids the performance loss due to **ssh** tunneling by sending the data directly to the **XRootD** server running on the launching host, as shown below.


```
frm.xfr.copycmd in noalloc ssh border xrdcp -np root://redir/$SRC root://$HOST/$DST
```
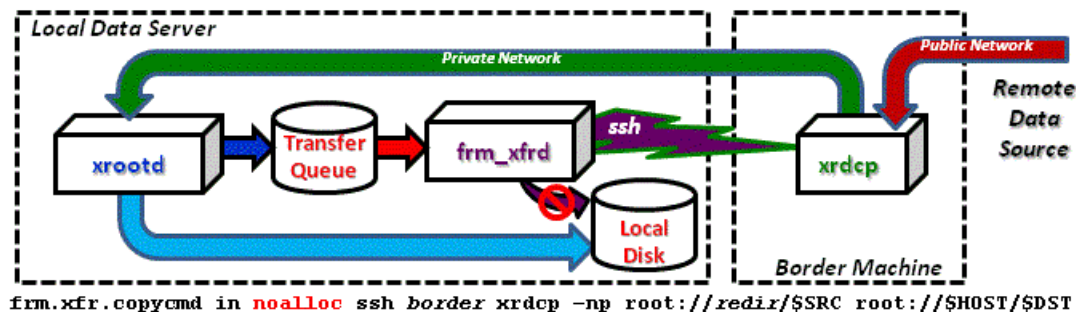
There are several things to note. First, we must disable **frm_xfrd** from pre-creating the destination file using the **noalloc** option. This is because the **XRootD** will create the file and the copy operation will fail if the file already exists.

Secondly, we need to authorize the *border* to write into directories that are typically exported as read-only. This is a bit of a complication because the only way to do this is to export stageable paths as read-write and prohibit everyone but the *border* machine from writing into those paths. If you are already implementing a complex security model, you may wish to run another instance of an unclustered **XRootD** server that can be only be used by the *border* machine and allows read-write access to stageable paths. Otherwise, the following diagram shows how to transform a simple configuration file to one that allows the *border* machine access.
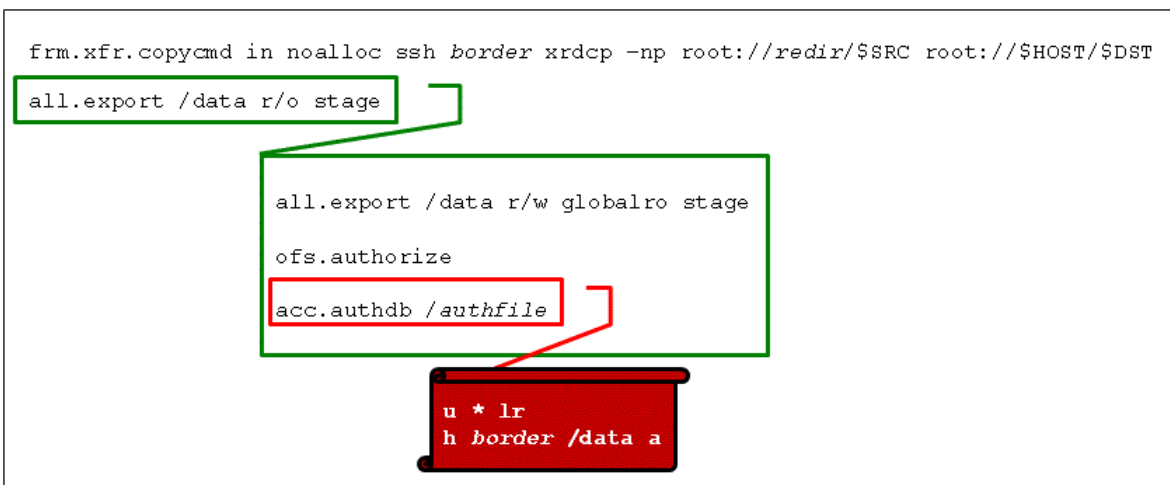
In the previous configuration file snippet,
- the **r/o** attribute in the **all.export** directive was changed to indicate that the path **/data** was now **r/w** (i.e. read-write) but it should be exported as r/o to the redirector (i.e. **globalro**). This allows data to be locally written to the path but keeps a consistent read-only view of the path in the cluster.
- The **ofs.authorize** directive was added to indicate that access privileges are to be enforced.
- The **acc.authdb** directive indicates where the authorization information is located (i.e. */authfile*).
- In the *authfile* the '**u**' record indicate that, by default, everyone has read-only access to data on this server. The '**h**' record establishes an read-write exception for any client coming from the border machine.

Because anyone on the *border* machine is given read-write access, the machine is not shareable with other **XRootD** clients. If this is too restrictive then you need to run another instance of an unclustered **XRootD** server that can be only be used by the *border* machine and allows read-write access to stageable paths.

1.1.3.4   Staging via a Global Redirector

Staging in data from a globally federated **XRootD** cluster is not much different from staging data an arbitrary remote source. Here, the data source for the **xrdcp** command points to the global redirector as the source of data and care has to be taken to prevent redirection loops. A redirection loop occurs when your cluster is federated with a global redirector and uses the global redirector to fetch a missing file using **xrd_frmd**. If the global redirector thinks that the requesting cluster may have the file, it may redirect **xrdcp** back to its cluster. However, since the file is actually missing the request proceeds to the **xrd_frmd** which simply asks the global redirector again. Hence, a redirection loop that slowly but eventually ends.

Each **XRootD** cluster is automatically assigned a globally unique cluster identifier. The **frm_xfrd** sets the variable **$CID** to contain this identifier. To avoid redirection loops via the global redirector you must tell it to ignore your cluster when looking for a file. This is done by adding additional cgi data (i.e., "tried=+$CGI") to the source path, as shown below.

```
frm.xfr.copycmd in noalloc ssh border xrdcp –np \
    xroot://globalrdr/$SRC?tried=+$CGI root://$HOST/$DST
```

1.1.3.4.1   Using The **oss.remoteroot** Directive

The **oss.remoteroot** directive allows you to automatically direct data source at a remote location and segregate them from the normal copy stream. This works only if all of the following are true
- the default **oss** plug-in to access the underlying storage is being used, and
- the **oss.namelib** directive is not specified.

If you are using a **name2name** plug-in, you can still supply the correct remote location but you will have to programmatically do so.

For instance, assume the remote location is host *globalrdr*. Then including the directive


```
oss.remoteroot xrootd://globalrdr/
```


The **oss** plug-in would automatically prefix the logical file name with "**xrootd:**//*globalrdr/*" when handing the file to be staged in to **frm_xfrd**. Consequently, the **frm.xfr.copycmd** directives can directly refer to the source of the data. For instance,


```
frm.xfr.copycmd in noalloc ssh border xrdcp -np \
     xroot://globalrdr/$SRC?tried=+$CGI root://$HOST/$DST


                         becomes

oss.remoteroot xroot://globalrdr/
frm.xfr.copycmd in noalloc url ssh border xrdcp -np \
     $SRC?tried=+$CGI root://$HOST/$DST
```


Notice that we added the "url" option to the **copycmd** directive. This is because the **frm_xfrd** will receive an actual **url** as the copy source, not a plain logical file name. A **url** formatted name needs no additional qualification as it contains all the required information.

To help you manage **url**-based copies, the **frm_xfrd** places them is a special input **url** queue referenced by the **copycmd** directive. If such a directive does not exist, no **url** copies are performed. This allows you to segregate **url** vs **lfn** based copy requests.

2   frm_admin Command

```
frm_admin [options] [cmd [cmdopts] cmdparms]

options: [-c cfn ] [-d] [-h] [-n name] [-v]

command: help | audit | chksum | exit | f[ind] | mark |

         mmap | pin | q[uery] | quit | reloc | rm
```

Function

Execute local file residency maintenance commands.

Options

-c *cfn* The configuration file to be used. The default configuration file, /opt/xrootd/etc/xrootd.cf, is read if it exists.

-d   Turn on debugging.

-h   Print usage information.

-n *name*

Is the name of the server instance for which the command is being performed. See the notes for more information.

-v   Prints additional messages where relevant.

Commands

cmd   Is one of the following:

    help   - provide additional usage information.
    audit   - verify file system consistency.
    chksum - calculate, set, display, or verify a file's checksum.
    exit   - terminate the program (used in interactive mode).
    find   - find existing or missing residency management file attributes.
    mmap   - set file memory mapping attributes.
    mark   - set file migration or purging attributes
    pin   - set pin attributes.
    query - display various information.
    reloc   - relocate files from one type of space to another.

      rm   - remove one or more files or directories.

cmdopts

Are command specific options. Refer to each command description for more information.

cmdparms

Are command specific parameters. Refer to each command description for more information.

Notes
1) When **frm_admin** is invoked with no parameters, it enters interactive prompt mode. In this mode you can execute a series of commands by entering new line separated functions and parameters in response to the prompt. Command-line editing tools are enabled on platforms that support them.
2) To leave interactive mode, use the **exit** or **quit** commands. Alternatively, a null line in response to the prompt causes the program to exit.

2.1 audit

```
audit [options] what

what:    names ldir | space sname[:sdir] | usage [sname]

options: [-fix] [-force] [-migratable] [-purgeable]

         [-recursive]
```

Function

Inventory the logical name space, physical data space, or usage information for validity.

Parameters

names

Audits the logical name space, *ldir*.

ldir   Is the logical name of the directory whose contents are to be audited.

space Audits the physical data space named *sname*.

usage Audits the space usage associated with physical data space named *sname*.

sname Is the name of the space to be audited for contents or usage. This is the name that is specified using the **oss.space** directive. By default, all directories associated with the named space are processed. For **usage**, if no *sname* is specified, all spaces are audited for usage.

sdir   Is a specific directory associated with *sname*, as specified using the **oss.space** directive. Only that directory is processed.

Options

-fix   Attempts to fix problems encountered during the audit.

-force When used in conjunction with **–fix**, automatically answers *yes* to every question posed. This, in effect, corrects all mistakes without intervention.

-migratable

With **–fix**, sets missing migration-purge file attributes in *ldir* to indicate that the file with the missing attributes must be migrated prior to being purged. This is the default when *ldir* is exported with the **mig** option.

-purgeable

With **–fix**, sets missing migration-purge file attributes in *ldir* to indicate that the file with the missing attributes may be purged.

-recursive

Recursively audits the names in all directories starting with *ldir*.

Notes on audit names

1) A copy-time attribute is associated with each data file to track file modifications. This attribute is automatically set when the space is exported with the **mig** or **stage** options.
2) When a names audit is requested the following occurs
    a. The directory is scanned for any files that lack a required copy-time attribute. Files in paths exported with the **mig** or **purge** and **r/w** options must have a copy-time attribute to make them eligible for migration and purging.
    b. The default fix action is to set the copy-time attribute that requires the base file to be migrated or allows it to be purged; depending on the subcommand options or defaults.
    c. Should a base file be a symbolic link to a named space, the link's consistency is verified. If the link point to a non-existent data file; the default fix action is to remove the dangling link.
    d. Additionally for XA spaces, the data file must be associated with pfn metadata. If the pfn metadata is missing or incorrect, the default fix action is to recreate the metadata.

Notes on audit space
1) Generally, space auditing is a deterministic process. Occasionally, irresolvable conflicts can arise. You should not use the **fix** option to repair spaces without first running the **audit names** subcommand.
2) When a space is audited the following occurs
   a. The directory containing the space is scanned to verify that each data file in the space is pointed to by a symbolic link in the associated name space. When a missing link is found and the physical name of the file can be determined, the default fix action is to create the symbolic link. If the physical name cannot be determined, the default fix action is to delete the data file and its associated metadata.
   b. Additionally for XA spaces, the data file must be associated with pfn metadata. If the pfn metadata is missing; the default fix action is to delete the data file. If the pfn metadata is incorrect, the default fix action is to create a new symbolic link in the logical name space pointing to the data file that is consistent with the pfn metadata.

Notes on audit usage
1) The **–fix** option is only meaningful when usage logging is enabled (see the **oss.usage** directive).
2) Space usage can be fixed while the **XRootD** server is running. Applying multiple fixes is allowed.

2.2 chksum

```
chksum [options] func path

options: [-force] [-pfn] [-type digest] [-verbose]

func:    calc | ls | set csval | unset | verify csval
```

Function

Get, set, and verify a file's checksum.

Parameters

calc   Calculates the checksum unless the file already has a valid checksum. Specifying **–force** will force the calculation of the checksum. When the checksum is calculated, the file's checksum is also updated with the calculated value. The file's checksum is also printed.

ls   Lists all the checksums associated with the file unless **–type** specifies a particular checksum.

set   Sets the file's checksum to the specified value, *csval*. The *csval* must be specified as an ASCII sequence of hexadecimal digits consistent with the associated checksum.

unset Deletes the file's default checksum information or the checksum specified with the **–type** option.

verify Effectively issues calc on the file and then compares the resulting checksum with *csval*; indicating whether the values are the same or not. The *csval* must be specified as an ASCII sequence of hexadecimal digits consistent with the associated checksum.

path   Is the path of the file to be acted upon. The *path* represents a logical name unless **–pfn** is specified, in which case *path* is taken as a physical filename.

Options

-force forces the calculation of a checksum even when the file has a valid checksum associated with it.

-pfn treats *path* as a physical file name and does not apply logical file name transformations.

-type specifies the checksum digest to be used. The **adler32**, **crc32**, and **md5** checksums are natively supported. An additional checksum may be defined with the **ofs.ckslib** directive. The default checksum corresponds to the digest specified on the **xrootd.chksum** directive. In absence of that directive, the default becomes **adler32**.

-verbose
Displays additional information when the checksum is printed. The default format is:

<div align="center">csval digest</div>

Where *csval* is the checksum value as an ASCII sequence of hexadecimal digits and digest is the algorithmic name for *csval*. When **–verbose** is specified the format is:

<div align="center">csval digest mm/dd/yy hh:mm:ss path</div>

Where *mm/dd/yy hh:mm:ss* is the date and time the checksum was calculated and *path* is the filename.

Notes
1) Checksum values are recorded in the file's extended attributes.

2.3 find

```
find [options] what ldir [ldir [. . .]]

what:    failfiles | mmapped | nochksum digest | pinned |

         unmigrated

options: [-recursive]
```

Function

   Search the logical name space for certain kinds of anomalies.

Parameters

failfiles

   Finds all fail meta-files (i.e., files suffixed with ".fail").

mmapped

   Finds all data files that have individual memory-map attributes and displays
   their memory-map properties. See the **mmap** subcommand for details.

nochksum *digest*

   Finds all data files that do not have a valid checksum for *digest* (e.g. **md5**);
   nocs is a synonym for **nochksum**.

pinned

   Finds all data files that are potentially pinned and displays their pinned
   properties. See the **pin** subcommand for details.

unmigrated

   Finds all data files whose attributes indicate an un-migrated status.

ldir   Is the logical name of the directory whose contents are to be searched. More
         than one logical directory name may be specified.

Options

-recursive
Recursively processes all subdirectories of *ldir*.

Notes
1) Except for the **mmapped** option, the **find** subcommand is useful only for
   name spaces that have been exported with the **mig** or **stage** options.

2.4 mark

```
mark [options] lspec [lspec [. . .]]

options: [-force] [-migratable] [-purgeable] [-recursive]

lspec:   lfn | ldir[/*]
```

Function

Set migration-purge attributes for one or more files.

Parameters

lfn   Is the logical name of a file whose migration-purge attributes are to be set.
          Multiple *lfn*'s may be specified.

ldir   Is the logical name of a directory. When *ldir* is suffixed by "/*' then migration
          and purge metadata is created or altered for all data files in *ldir*. Specifying –
          recursive applies **mark** to all files in *ldir* and all of its descendants. Multiple
          ldir's may be specified.

Options

-force Does not ask questions regarding possible conflicts when setting or over-
          riding migration-purge attributes.

-migratable

Sets the migration-purge attributes to indicate that *lfn* must be migrated prior
to being purged. This is the default.

-purgeable

Sets the migration-purge attributes to indicate that *lfn* may be purged.

-recursive

Recursively processes all subdirectories of *ldir*.

Notes

1) The **mark** command is useful only for name spaces that have been exported with the **mig** or **stage** options.

2) You must quote or escape *ldir* when specifying *ldir* with an asterisk on the **frm_admin** command line.

2.5   mmap

```
mmap [options] lspec [lspec [. . .]]

options: [-keep] [-lock] [-off] [-recursive]

lspec:   lfn | ldir[/*]
```

Function

  Set memory mapping attributes for one or more files.

Parameters

lfn Is the logical name of a file whose memory map attributes are to be set. Multiple *lfn*'s may be specified.

ldir Is the logical name of a directory. When *ldir* is suffixed by "/*' then memory map attributes are set for all data files in *ldir*. Specifying –**recursive** applies **mmap** to all files in *ldir* and all of its descendants. Multiple *ldir*'s may be specified.

Options

-keep requests that once the memory mapping is established, it should be kept even when the file is not open.

-lock requests that memory pages be locked in memory, if possible.

-off removes all memory map attributes. This option takes precedence over any other attribute option.

-recursive

  Recursively processes all subdirectories of *ldir*.

Notes

1) The **mmap** command without merely allows the file to be mapped in memory.

2) The **mmap** command is useful only for name spaces that have been exported with the **mcheck** option that allows individual file **mmap** attributes to be honored.

3) You must quote or escape *ldir* when specifying *ldir* with an asterisk on the **frm_admin** command line.

4) The **mmap** subcommand is supported only for file systems that support extended attributes.

2.6 mv

```
mv oldlfn newlfn
```

Function

Rename an existing file.

Parameters

oldlfn Is the logical name of the file to be renamed.

newlfn Is the new logical name. The new name may not exist.

2.7 pin

```
pin [options] lspec [lspec [. . .]]

options: [-keep tspec] [-recursive]

tspec:   [+]num[d|h|m|s] | mm/dd/yy | forever

lspec:   lfn | ldir[/*]
```

Function

      Set pin attributes for one or more files to control file purging.

Parameters

lfn   Is the logical name of a file for which pin attributes are to be set. Multiple *lfn*'s may be specified.

ldir   Is the logical name of a directory. When *ldir* is suffixed by "/*' then pin attributes are set for all data files in *ldir*. Specifying **–recursive** applies **pin** to all files in *ldir* and all of its descendants. Multiple *ldir*'s may be specified.

Options

-keep Sets amount of time a file must be kept in local disk. To keep a file on disk unless it has not been accessed for a period of time, specify the period as **+***num*[**d** | **h** | **m** | **s**] where *num* is the quantity and **d** is for days, **h** for hours, **m** for minutes, and **s** for seconds (the default). To keep a file on disk for a certain amount of time past midnight, specify the time as *num*[**d** | **h** | **m** | **s**] without the leading plus sign. To keep a file on disk until a particular date, specify the time in date format (i.e., keep until this date) as *mm/dd/yy* where *mm* is the month, *dd* is the day, and *yy* is the year. To keep the file on disk indefinitely, specify the time as **forever**. To unpin a file specify a period of zero (i.e. 0).

-recursive

      Recursively processes all subdirectories of *lspec* should it be a directory.

Notes

1) The **pin** command is useful only for name spaces that have been exported with the **purge** option.
2) Default pinning attributes are based on the configuration supplied to the purge daemon. The file pin attributes over-ride the defaults.
3) When **keep** time is zero, the file pin attributes are removed.
4) You must quote or escape *ldir* when specifying *ldir* with an asterisk on the **frm_admin** command line.
5) The **pin** subcommand is supported only for file systems that support extended attributes.

2.8 query

```
query    what

what:      pfn lspec [lspec [. . .]]
         | rfn lspec [lspec [. . .]]
         | space  [[-recursive] lspec [lspec [...]]
         | usage  [sname]
         | xfrq [qtypes] [prty] [vars]

lspec:   lfn | ldir[/*]

qtypes:  {all | get | migr | put | stage} [qtypes]

vars:    {lfn | lfncgi | mode | obj | objcgi | op | prty |

          qwt | rid | tod | note | tid} [vars]
```

Function

   Display various information.

Parameters

pfn   Displays the local disk physical file name corresponding to the specified
         logical name. The name need not exist. More than one *lspec* may be specified.

rfn   Displays the remote physical file name corresponding to the specified logical
         name. The name need not exist. More than one *lspec* may be specified.

space Without arguments, displays the configured spaces as defined by the
         oss.space directive. When *lspec* is specified, displays the space in which one
         or more files reside. The *lspec* may be preceded by **–recursive** or **–r** to
         recursively processes all subdirectories of *lspec* should it be a directory.

lfn   Is the logical name to be queried. Multiple *lfn*'s may be specified.

ldir   Is the logical name of a directory to be queried. Multiple *ldir*'s may be
         specified.

ldir/*   Is the logical name of a directory. For **query space**, the query is applied to all data files in the directory. This is the default for **query space** when *lspec* is a directory name. Multiple *ldir/**'s may be specified.

usage   Displays usage information based on the usage log file for the space named *sname*. If *sname* is not specified, usage for all named spaces is displayed.

xfrq   Displays **frm_xfrd** queue information. These are files that need to be copied into or out of the server.

qtypes   by default, all queues are listed. Otherwise, *qtypes* lists particular queues, as follows:

    all   - all queues, the default.
    get   - the copy-in queue for specific client-issued request.
    migr   - the migrate queue for the FRM migration service.
    put   - the copy-out queue for specific client-issued requests.
    stage - the stage-in queue for the FRM pre-stage service.

prty   by default, all priorities are listed. Otherwise, *prty* specified the particular priority to be listed (i.e. 0, 1, or 2).

vars   is an optional list of variable names, as described below. Information is listed in the order in which the variables are listed; each separated by a space. The default variable is **lfn**. Valid variables are:

Var	Information	Var	Information	Var	Information
lfn	logical filename	**obj**	**lfn** or *url*	**qwt**	seconds in queue
lfncgi	lfn?[*cgi* string]	**objcgi**	**lfncgi** or *url*?[*cgi*]	**rid**	requestid
mode	processing opts	**op**	Operation (e.g., '<')	**tid**	traceid
note	notification string	**prty**	priority	**tod**	time of day

Notes

1) The **query space** subcommand is only useful when a symlinked file system has been defined with the **oss.space** directive.
2) The **query usage** subcommand is only useful with usage logging has been enabled with the **oss.usage** directive.
3) Usage for non-XA spaces contains limited information and is not necessarily accurate relative to other non-XA spaces.
4) Usage information is displayed in bytes with the following tags:
    a. **Space**   the name of the space.
    b. **Used**   bytes currently recognized by **XRootD** as being used.
    c. **Staged**   bytes staged by the pre-stage daemon.
    d. **Purged**   bytes purged by the purge daemon.
    e. **Adjust**   the correction factor determined by **audit usage –fix**.
    f. **Effective**   (**Used** + **Staged** – **Purged** + **Adjust**)
5) The **staged**, **purged**, and **adjust** values are periodically reset to zero when the **XRootD** daemon re-computes the quantity it considers in-use based on the previous values.
6) You must quote or escape *ldir* when specifying *ldir* with an asterisk on the **frm_admin** command line.
7) The output of **query xfrq** is largely self-explanatory. Additional details can be found in the section describing **frm_xfragent**. For the variable **op**, the following designations are used:
    < - copy file *lfn* from a remote location to local disk.
    = - copy file *lfn* to a remote location and then remove it from local disk.
    > - copy file *lfn* to a remote location.
    + - stage file *lfn* from a remote location to local disk.
    ^ - migrate file *lfn* to a remote location and then remove it from local disk.
    & - migrate file *lfn* to a remote location.

2.9 reloc

```
reloc lfn sname[:sdir]
```

Function

Relocate a data file from one space to another.

Parameters

lfn   Is the logical name of the file to be relocated.

sname Is the name of the space in which the file is to reside.

sdir   Is a specific directory associated with *sname*, in which the file is to reside.

Notes
1) The **reloc** subcommand is only supported for XA space targets. That is, the file may reside in any space but may only be relocated to an XA space.
2) If *lfn* already exists in *sname*, then you must specify *sdir* and it must differ from the directory in which the data file resides.

2.10 rm

```
rm [options] lspec [lspec [. . .]]

options: [-echo] [-force] [-recursive]

lspec:    lfn | ldir[/*]
```

Function

Remove one or more files from local disk.

Parameters

lfn   Is the logical name of a file to be removed. Multiple *lfn*'s may be specified.

ldir   Is the logical name of a directory to be removed. The directory must be effectively empty. Multiple *ldir*'s may be specified.

ldir/* Is the logical name of a directory. All entries in the directory, but not the directory itself are removed. Removal fails if the directory contains another directory. In this case, you must use the **–recursive** option. Multiple *ldir/**'s may be specified.

Options

echo Displays the physical name of every removed file.

force Does not ask for permission prior to removing a file.

recursive

Recursively removes all files and directories starting with *ldir* before removing *ldir* itself.

Notes
1) The **recursive** option only applies to directories and is ignored if *lspec* is not a directory.
2) You must quote or escape *ldir* when specifying *ldir* with an asterisk on the **frm_admin** command line.

3   The frm_purged Command for Purging

```
frm_purged [ options ] [ args ]

args:      [sname] [path] [args]

options: [-b] [-c cfn] [-d] [-f]

           [-k {num | sz{k|m|g} | sig}] [-l [=]logfn]

           [-n name] [-O free[,hold]] [-s pfn] [-S site]

           [-T] [-v] [-z]

sig:       fifo|hup|rtmin|rtmin+1|rtmin+2|ttou|winch|xfsz
```

Function

   Manage the purging of files inactive files from local disk.

Arguments

sname Restricts purging to the indicated space. Only files residing in *sname* are potentially purged. Any number of space names may be specified but each specified name must have been defined via the **oss.space** directive.

path Restricts purging to the indicated path and all of its descendants. Only files residing on *path* and defined as purgeable via the **all.export** directive are potentially purged. Any number of path names may be specified but each specified name must have been exported via the **all.export** directive.

Options

-b Runs in true daemon mode as a background process.

-c *configfn*

   The configuration file name. The default name for the configuration file is "/opt/xrootd/etc/xrootd.cf".

-d Turns on debugging mode. Additional information is printed to describe various actions.

-f   Automatically removes (i.e., fixes) orphaned fail files.

-k *num* | *sz*{**k**|**m**|**g**} | *sig*

Keep no more than *num* old log files. If *sz* is specified, the number of log files kept (excluding the current log file) is trimmed to not exceed *sz* bytes. The *sz* must be suffixed by **k**, **m**, or **g** to indicate **k**ilobytes, **m**egabyte, or **g**igbytes, respectively. If a *sig* value is specified (i.e. **hup** etc), then an external program is expected to handle log file rotation (e.g. logrotate). Except for **fifo**, the argument specifies signal that causes the daemon to close and re-open the log file to allow rotation to occur. When fifo is specified, the daemon waits for data to appear on a fifo whose path is identical to the log file path but whose name is prefixed by a dot. Refer to the notes for manual rotation caveats.

-l *logfn*

Routes error messages and any trace output to *logfn*. By default, messages are directed to standard error. When *fn* is prefixed by an equals sign, the *fn* is not qualified by the instance name, if any. This allows log files to be handled in an arbitrary manual way. For more information see the section on fencing daemon instances.

-n *name*

Assigns *name* to the **frm_purged** instance. By default, the **frm_purged** instance is unnamed. See the notes on how to use this option.

-O *free*[**,**\ *hold*]

Runs **frm_purged** as a command and performs a one-time purge. Specify for *free* either a percentage of free space (e.g., 10%) that is required or an actual amount optionally suffixed by **k**, **m**, or **g** to indicate **k**ilobytes, **m**egabytes, or **g**igbytes, respectively. Optionally specify for *hold* the minimum amount of time must have not been accessed in order to be purged. The *time* may be suffixed by **h**, **m**, or **s** to indicate **h**ours, **m**inutes, or **s**econds (the default), respectively. The one-time policy applies to all spaces selected for purging. Unspecified values come from each configured policy.

-s *pfn* Specifies the name of the file that is to hold the process id upon start-up.

-S *site* Specifies a 1- to 15-character site name that is to be included in monitoring records. The name may only contain letters, digits and the symbols "_-:."; any other characters are converted to a period.

-T   Runs in test mode. No files are actually deleted.

-v   Increases verbosity by displaying additional information about which files were purged.

-z   provides microsecond resolution for log file message timestamps.

Notes
1) The **frm_purged** also accepts additional directives from the configuration file, *configfn*. These directives are described in the following section.
2) The **–b** and **–s** options are meant to be used by start-up scripts (e.g. **init.d**). When **–s** is specified, an additional pid-file is created.
3) If a log file is specified without a signal -k option, the file is closed at midnight, renamed to have a date suffix (i.e., *fn.yyyymmdd*) and possible sequence number (i.e. *fn.yyyymmdd.n*), and a new log file is opened.
4) When a signal value is specified, log files are not automatically renamed at midnight. Instead an external program must be used to properly rotate log files. Make sure to choose a signal that is *not* in use by *any* plug-in. If unsure, choose one of the obscure signal names and monitor for any odd behavior. Otherwise, use the **fifo** option. Be aware that on some non-Linux platforms the fifo file descriptor may leak.
5) When **fifo** is specified the fifo file name must not exists or exist as a fifo file. A simple "**echo x >>** */path/.lfn*" causes the log file to close and reopen.
6) The *sig* names, except for **fifo**, be fully capitalized as well prefixed by "**sig**" or "**SIG**" when capitalized.

3.1 Exported Environment Variables

The following table shows the environment variable exported by **frm_purged**. These may be used by external programs and plug-ins, as needed. They should never be modified.

Variable	Contents
XRDCONFIGFN	The effective administrative path used for server management files.
XRDDEBUG	Set to one when the –d command line option is specified.
XRDHOST	The current host's DNS name.
XRDLOGDIR	Is the directory where log files are written.
XRDNAME	The name specified via –n or anon if no instance name was specified.
XRDPROG	The executable's name.
XRDSITE	The site name specified either via the –s command line option or the all.sitename directive.

3.2 The **frm_purged** Configuration directives

In addition to the directives prefixed by "**frm.purge**", **frm_purged** also uses certain **ofs**, **oss**, and **xrood**-specific directives, if already specified, for configuration. They are shown grayed-out below.

```
all.adminpath        apath

all.pidpath          ppath

frm.all.monitor      [ident sec] dest [ dest ]

frm.purge.dirhold    hold

frm.purge.policy     {* | sname} [nopurge | polargs]

frm.purge.polprog    [vars] |path [pargs]

frm.purge.waittime   wsec

ofs.osslib           libpath [ parms ]

oss.localroot        lpath

oss.namelib          npath [ parms ]

oss.remoteroot       rpath
_____

dest:    dest events host:port

events:  [migr] [purge] [stage]

vars:    atime | ctime | fname | fsize | fspace

         mtime | pfn | sname | tspace | usage

polargs: [minfree [maxfree]] [hold hold] [polprog]

hold:    forever | htime
```

Directives

frm.all.monitor [**ident** *sec*] **dest** *events dest* [**dest** *events dest*]

Enables monitoring of events (see the "Scalla Monitoring" reference for more information). Event records can be sent via UDP to one or two destinations, *dest* (i.e. *host***:***port*) You select which records are sent by specifying one more event types. Valid *events* are:

migr   files copied to remote storage by **frm_xfrd.**

purge files removed from local storage by **frm_purged.**

stage   files copied from remote storage by **frm_xfrd.**

The **ident** option is used to specify the number of seconds between each server identity transmission (i.e., the '**=**' map record). Specify a number optionally suffixed by h for hours, m for minutes, or s for seconds, the default. A value of zero transmits the identity only once at start-up time. The default is 1 hour (i.e. 3600 seconds).

frm.purge.dirhold *htime*

Specifies how long empty directories are to be kept after being created. If *htime* is the word "**forever**", empty directories are not removed. Otherwise, empty directories are removed after *htime* past the last modification. The *htime* may be suffixed by **h**, **m**, or **s** to indicate **h**ours, **m**inutes, or **s**econds (the default), respectively. The default is 40h.

frm.purge.policy {***** | *sname*} **nopurge** | [*minfree* [*maxfree*]] [**hold** *hold*] [**polprog**]

Defines a purging policy. An asterisk (*****) defines the default policy and is used to complete all subsequent policy directives; while an *sname* defines a policy for the named space. Specifying **nopurge** prevents file purging. Otherwise, specify one or more of the following:

minfree   the minimum amount of free space. Purging begins when the amount of free space falls below this value. Specify a percentage of the total space (e.g., 10%) or an absolute amount space, optionally suffixed by **k**, **m**, or **g** to indicate **k**ilobytes, **m**egabytes, or **g**igbytes, respectively. The default is 2%.

maxfree   the maximum amount of free space. Purging stops when the amount of free space goes above this value. Specify a percentage of the total space or an absolute amount space which may be optionally suffixed by **k**, **m**, or **g** to indicate **k**ilobytes, **m**egabytes, or **g**igbytes, respectively. The *maxfree* value must be greater than or equal to the *minfree* value. The default is 3% if *minfree* is not specified. Otherwise, if *minfree* is a percentage the default is *minfree*+1 and if *minfree* is an absolute value the default is *minfree**120%.

htime   the minimum amount of time the file must have not been accessed in order to be purged. The *htime* may be suffixed by **h**, **m**, or **s** to indicate **h**ours, **m**inutes, or **s**econds (the default), respectively. The default is 20h (hours).

polprog   invokes the program defined by a previous *polprog* directive in order to determine if the file can be purged once all other criteria have been met.

frm.purge.polprog [*vars*] |*path* [p*args*]

Defines an external policy. The external policy is implemented by the program named by path. The program is started with optional arguments *pargs*. Whenever a purge decision is needed the indicated variable values, terminated by a newline character ('\n'), are sent to the program's standard-in. The program must write a single character response, terminated by a newline, to standard-out (see the usage notes). The following variables may be specified:

atime   file's last access time in Unix seconds.

ctime   file's creation time in Unix seconds.

fname   file name, excluding the directory path.

fsize   size of the file in bytes.

fspace   bytes of free space available in the file's designated space.

mtime   file's last modification time in Unix seconds.

pfn   physical file name, including its full path.

sname   name of the space in which the file resides.

tspace   total bytes allocated to the file's designated space.

usage   bytes used in the file's designated space. This may be -1 if usage information is not maintained by space name.

frm.purge.waittime *wsec*

wsec is the number of seconds to wait before checking whether purging is necessary. The *wsec* may be suffixed by **h**, **m**, or **s** to indicate **h**ours, **m**inutes, or **s**econds (the default), respectively. The default is 5 minutes.

3.2.1   Directives used but documented in the "ofs/oss Reference"

ofs.osslib *libpath* [*parms*]
> *libpath* is the absolute path to the shared library that contains the
> implementation of the storage system interface that **frm_xfrd** is to use for file
> system specific storage operations (e.g., create, rename, etc). The default is to
> use a built-in mechanism that is identical to what **XRootD** uses by default.

oss.localroot *lpath*
> *lpath* is the path that must prefix any logical file name before using it as a
> local physical file name (i.e. to map *lfn* to local *pfn*). By default, no prefix is
> used. Also see the **oss.namelib** directive.

oss.namelib *npath* [*parms*]
> *npath* is the absolute path to the shared library that contains the
> implementation of the name-to-name translation interface that **frm_xfrd** is to
> use to convert logical file names to local and remote physical file names. The
> default is to use a built-in mechanism that relies on the **oss.localroot** and
> **oss.remoteroot** directives.

oss.rmoteroot *rpath*
> *rpath* is the path that must prefix any logical file name before using it as a
> remote physical file name (i.e. to map *lfn* to Mass Storage System *pfn*). By
> default, no prefix is used. Also see the **oss.namelib** directive.

3.2.2   Directives used but documented in the "xrd/xrootd Reference"

all.adminpath *apath*
> *apath* is the fully qualified administrative base path where various directories
> and special files may be created to control execution. The default is "/tmp".
> The **–n** option augments the base path.

all.pidpath *apath*
> *ppath* is the fully qualified base path where the **frm_purged.pid** file is to be
> written. The default is "/tmp". The **–n** option augments the base path.

Notes
1) The **frm_purged** program periodically performs a full file system namespace scan. The frequency of the scan is determined by the smallest policy hold time. Full namespace scans are skipped as long as sufficient files remain that meet the policy from the previous scan.
2) The **polprog** directive identifies an external policy program. The program is started at initialization time and is expected to remain active to accept input from standard-in and provide responses to standard-out. It is automatically restarted should it fail. The default set of tokens sent to the program are "**sname pfn fsize atime mtime\n**" (i.e., space name, physical file name, file size, last access time, last modification time). The program must respond with a single character followed by a newline character (i.e., "*x*\n"). There are three possible response sequences:
   a) "**n\n**" which prevents the associated file from being purged,
   b) "**y\n**" which purges the associated file, and
   c) Any other sequence which not only prevents the associated file from being purged but also stops the purging process associated with the space until the next purge cycle starts.

3.3 Created Files

The following files are created by the **frm_xfrd**:

Path	Type	Modified by	Purpose
/tmp/[*name*/]frm_purged.pid	File	**pidpath** and **–n** option	Holds the process id of the associated frm_purged server
/var/adm/frm/core/[*name*/]core	File	**–n** option	Core file via default in **StartXRD.cf**
/var/adm/frm/logs/[*name*/]xfrlog	File	**-l** option and **–n** option	Log file via default in **StartXRD.cf**

The **adminpath** directive specifies the directory where the remaining files are written. The -**n** option specifies the **frm_purged** instance name. If specified, the instance name is automatically suffixed to the **adminpath** or /tmp, as shown by "[*name*/]." A directory is also create in the current working directory for core files and the log file destination is modified by inserting "[*name*/]." in the destination specified by the **–l** option. If necessary, the directory is created.

3.4   Temporarily Stopping frm_purged

Each time **frm_purged** is about to schedule a purge scan, it check whether a *stopfile* exists indicating that purging is not allowed. The *stopfiles* is:

/*apath***/frm**[**/***name*]**/STOPPURGE**

Where

apath   comes from the **all.adminpath** directive or its default.

name   comes from the –n command line option and is empty if not specified.

The presence of the file stops purging.

4   frm_xfrd Command for the Transfer Daemon

```
frm_xfrd [ options ]

options: [-b] [-c cfn] [-d] [-k {num | sz{k|m|g} | sig}]

         [-l [=]logfn] [-n name] [-s pfn] [-S site] [-T]

         [-v] [-z]

sig:     fifo|hup|rtmin|rtmin+1|rtmin+2|ttou|winch|xfsz
```

Function

Manage the copying of files from a Mass Storage System to local disk.

Options

-b   Runs in true daemon mode as a background process.

-c *configfn*

The configuration file name. The default name for the configuration file is
"/opt/xrootd/etc/xrootd.cf".

-d   Turns on debugging mode. Additional information is printed to describe
various actions.

-k *num | sz{**k**|**m**|**g**} | sig*

Keep no more than *num* old log files. If *sz* is specified, the number of log files
kept (excluding the current log file) is trimmed to not exceed *sz* bytes. The *sz*
must be suffixed by **k**, **m**, or **g** to indicate **k**ilobytes, **m**egabyte, or **g**igbytes,
respectively. If a *sig* value is specified (i.e. **hup** etc), then an external program
is expected to handle log file rotation (e.g. logrotate). Except for **fifo**, the
argument specifies signal that causes the daemon to close and re-open the log
file to allow rotation to occur. When fifo is specified, the daemon waits for
data to appear on a fifo whose path is identical to the log file path but whose
name is prefixed by a dot. Refer to the notes for manual rotation caveats.

-l *logfn*

Routes error messages and any trace output to *logfn*. By default, messages are directed to standard error. When *fn* is prefixed by an equals sign, the *fn* is not qualified by the instance name, if any. This allows log files to be handled in an arbitrary manual way. For more information see the section on fencing daemon instances.

-n *name*

Assigns *name* to the **frm_xfrd** instance. By default, the **frm_xfrd** instance is unnamed. See the notes on how to use this option.

-s *pfn*   Specifies the name of the file that is to hold the process id upon start-up.

-S *site*   Specifies a 1- to 15-character site name that is to be included in monitoring records. The name may only contain letters, digits and the symbols "_-:."; any other characters are converted to a period.

-T   Runs in test mode. In test mode, no destructive actions (e.g., file removal) are taken.

-v   Produces additional details during executions (i.e., verbose mode).

Notes

1) The **frm_xfrd** also accepts additional directives from the configuration file, *configfn*. These directives are described in the following section.
2) The **–n** option allows you to run multiple instances of **frm_xfrd** with a common configuration file. This is possible because the specified name is used to modify various file system paths **frm_xfrd** uses for output files. By automatically differentiating such paths by instance name prevents two **frm_xfrd** processes from interfering with each other.
3) The **frm_xfrd** verifies that it is the only one running with a given instance name. If another **frm_xfrd** is discovered, the program exits.
4) The chosen instance name must be the same as assigned to the **frm_xfragent** that supplies **frm_xfrd** work.
5) The **cmsd** and **xrootd** have built-in mechanisms to communicate with **frm_xfrd**.
6) The **–b** and **–s** options are meant to be used by start-up scripts (e.g. **init.d**). When **–s** is specified, an additional pid-file is created.

7) If a log file is specified without a signal **-k** option, the file is closed at midnight, renamed to have a date suffix (i.e., *fn.yyyymmdd*) and possible sequence number (i.e. *fn.yyyymmdd.n*), and a new log file is opened.

8) When a signal value is specified, log files are not automatically renamed at midnight. Instead an external program must be used to properly rotate log files. Make sure to choose a signal that is *not* in use by *any* plug-in. If unsure, choose one of the obscure signal names and monitor for any odd behavior. Otherwise, use the **fifo** option. Be aware that on some non-Linux platforms the fifo file descriptor may leak.

9) When **fifo** is specified the fifo file name must not exists or exist as a fifo file. A simple "**echo x >>** */path/.lfn*" causes the log file to close and reopen.

10) The *sig* names, except for **fifo**, be fully capitalized as well prefixed by "**sig**" or "**SIG**" when capitalized.

4.1 Exported Environment Variables

The following table shows the environment variable exported by **frm_xfrd**. These may be used by external programs and plug-ins, as needed. They should never be modified.

Variable	Contents
XRDCONFIGFN	The effective administrative path used for server management files.
XRDDEBUG	Set to one when the –d command line option is specified.
XRDHOST	The current host's DNS name.
XRDLOGDIR	Is the directory where log files are written.
XRDNAME	The name specified via –n or anon if no instance name was specified.
XRDPROG	The executable's name.
XRDSITE	The site name specified either via the –s command line option or the all.sitename directive.

4.2 The frm_xfrd Configuration directives

In addition to the directives prefixed by "**frm.xfr**" and "**frm.xfr.migr**" (for the automatic migration component), **frm_xfrd** also uses certain **ofs**, **oss**, and **XRootD**-specific directives, if present, for configuration. They are shown grayed-out below. Refer to the corresponding manuals for their full descriptions.

```
all.adminpath          apath

all.pidpath            ppath

frm.all.monitor        [ident sec] dest [ dest ]

frm.xfr.copycmd        [opts] cmd [ args ]

frm.xfr.copymax        cmax | split inmax outmax

frm.xfr.qcheck         [qsec] [qpath]

frm.xfr.migr.idlehold  isec

frm.xfr.migr.waittime  wsec

ofs.osslib             libpath [ parms ]

oss.localroot          lpath

oss.namelib            npath [ parms ]

oss.remoteroot         rpath

oss.xfr                deny dt

xrootd.monitor         dest stage host:port
```

```
    dest:     dest events host:port

    events:  [migr] [purge] [stage]

    opts:    in | noalloc | out | stats | timeout tsec | url |

             xpd [opts]

    args:    [text] [var] [args]

    var:     $CID | $CGI | $DST | $INS | $LFN | $PFN | $RFN |

             $NOTIFY | $MDP | $OFLAG | $PRTY | $RID | $SRC |

             $eVar
```

Directives

frm.all.monitor [**ident** *sec*] **dest** *events dest* [**dest** *events dest*]
> Enables monitoring of events (see the "Scalla Monitoring" reference for more information). Event records can be sent via UDP to one or two destinations, *dest* (i.e. *host***:***port*) You select which records are sent by specifying one more event types. Valid *events* are:
> migr   files copied to remote storage by **frm_xfrd.**
> purge files removed from local storage by **frm_purged.**
> stage   files copied from remote storage by **frm_xfrd.**
>
> The **ident** option is used to specify the number of seconds between each server identity transmission (i.e., the '=' map record). Specify a number optionally suffixed by h for hours, m for minutes, or s for seconds, the default. A value of zero transmits the identity only once at start-up time. The default is 1 hour (i.e. 3600 seconds).

frm.xfr.copycmd [*opts*] *cmd* [*args*]

        cmd is the absolute path of the program that **frm_xfrd** is to use to transfer a file. One or more commands may be specified, depending on opts:

in   *cmd* is only used to transfer files into the server. If **copycmd in** is defined but a **copycmd out** is not defined; outgoing transfers will fail. If *cmd* is not qualified with **in** or **out**, *cmd* is used to transfer a file regardless of direction.

noalloc

        does not create a placeholder in the filesystem for incoming files.

out   *cmd* is only used to transfer files out of the server. If **copycmd out** is defined but an **copycmd in** is not defined; incoming transfers will fail. If *cmd* is not qualified with **in** or **out**, *cmd* is used to transfer a file regardless of direction.

stats transfer statistics are written to the log file. See the next section for the format and meaning of these statistics.

timeout *tsec*

        cmd is allowed up to *tsec* seconds to complete before it is killed. The *tsec* value can suffixed by **h**, **m**, or **s** to indicate **h**ours, **m**inutes, or **s**econds (the default), respectively. The default is zero, preventing any timeout from occurring.

url   *cmd* is only used if the source or destination filename is in url-format. If **copycmd url** is not defined, any url-based transfers will fail.

xpd   includes the first line of output from the copy program into the monitoring record tagged as '**pd=**'. See the notes for more information.

args are arbitrary command-specific tokens and may include special variables (see the notes for substitution rules) that are substituted each time *cmd* is invoked. The following variables may be specified:

$CID   the global cluster identifier issuing this request.

$CGI   all of the opaque information specified after the question mark in the file path.

$DST   the target file name. This may be **$PFN** or **$RFN**, depending on the transfer direction. It may also be a url for url-based copy commands.

$INS   instance name (**–n** option) issuing this request.

$HOST   the hostname of where the daemon is running.

$LFN   logical file name.

$PFN	the local physical file name as modified by **localroot** or the **namelib**.
$RFN	remote file name as modified by remoteroot or the namelib.
$NOTIFY	notification string. See the cms.prepmsg or the **oss.stagemsg** directive for details.
$MDP	the offset into the destination file path where directories likely need to be created for the transfer to succeed. See the notes for more information about this variable.
$OFLAG	a character letter describing the file open processing flags:

                     w – O_WRONLY | O_RDWR   **r** – O_RDONLY

$PRTY	request priority.
$RID	request identifier.
$SRC	the source file name. This may be **$PFN** or **$RFN**, depending on the transfer direction. It may also be a url for url-based copy commands.
$TID	original requesting client's trace identification.
$*eVar*	any variable that has been passed along with the file name as opaque information (i.e., contained in **$CGI**).

frm.xfr.copymax *cmax*

    cmax specifies the maximum number of files that may be transferred at the same time. The default is 2.

frm.xfr.copymax split *inmax outmax*

    inmax specifies the maximum number of files that may be transferred to the server due to get or stage requests at the same time. *outmax* specifies the maximum number of files that may be transferred from the server due to put or migration requests at the same time. The total numbers of simultaneous transfers is *inmax+outmax*.

frm.xfr.qcheck [*qsec*] [*qpath*]

>   specifies the queue check interval and/or the queue path. Specify *qsec* or *qpath*, or both, as follows:

>   *qsec*   specifies how often the transfer queues are to be checked for new additions. Normally, **xfr_xfrd** is notified when an addition is made and immediately checks the appropriate queue. A manual check is periodically made in case a notification is lost. The *qsec* may be suffixed by **h**, **m**, or **s** to indicate **h**ours, **m**inutes, or **s**econds (the default), respectively. The default is 5 minutes.

>   *qpath*   specifies the absolute location where the queue files are to be written. The specified path is not augmented by the instance name. This allows you to share a single instance of **frm_xfrd** with multiple clusters running on the same host. By default, the queue files are written using the **adminpath** location.

frm.xfr.migr.idlehold *isec*

>   *isec* specifies the number of seconds a file must be idle before it is eligible for automatic migration to a Mass Storage System. This is part of automatic migration which is enabled when migratable paths have been defined via the **all.export** directive. The *isec* may be suffixed by **h**, **m**, or **s** to indicate hours, **m**inutes, or **s**econds (the default), respectively. The default is 10 minutes. See the usage notes before changing the default.

frm.xfr.migr.waittime *wsec*

>   *wsec* specifies the number of seconds to wait between scans of the name space to find files eligible for migration to a Mass Storage System. This is part of automatic migration which is enabled when migratable paths have been defined via the **all.export** directive. The *wsec* may be suffixed by **h**, **m**, or **s** to indicate **h**ours, **m**inutes, or **s**econds (the default), respectively. The default is 60 minutes. See the usage notes before changing the default.

Notes

1) You must use the **frm_xfragent** command to communicate with **frm_xfrd**. The **xfr_agent** command is responsible for adding, removing, and listing transfer requests queued to **frm_xfrd** (see the next section).

2) You can have up to four different copy commands:

	non-url copy command	url based copy command
In	1	2
Out	3	4

3) The copy command *must* indicate success or failure by its return code:

RC	Meaning	RC	Meaning	RC	Meaning
0	Copy succeeded	2	file not found	5	I/O error
>0	fatal error				

4) **r.migr.idlehold** value sets the minimum amount of time that a file must remain unmodified before it can be migrated. This minimizes migrations that need to be restarted should the file change. Smaller values increase the chance a file migration will have to be redone.

5) The **frm.xfr.migr.waittime** value sets the maximum amount of time between full name space scans. Full name space scans are disk I/O intensive and should be minimized. Since scans more often than the **frm.xfr.migr.idlehold** value can't produce new files to migrate, the minimum is automatically capped at the **idlehold** value.

6) When **copymax** *without* **split** is specified, the queue is alternately search for incoming copies and outgoing copies. When a choice exists in one of those selections (e.g. incoming copies due to get vs. stage requests) the oldest request is chosen. This allows fair starvation free scheduling.

7) When **copymax** *with* **split** is specified, *inmax* plus *outmax* threads are created. The *inmax* threads get requests from incoming while the *outmax* threads get requests from the outgoing queue. When a choice exists in one of those selections (e.g. incoming copies due to get vs. stage requests) the oldest request is chosen. This allows fair starvation free scheduling. Be aware that threads can become idle if they do not have enough work in the queue assigned to them even though the alternate queue is has more than enough work to satisfy all the threads.

8) Using the **$MDP** variable causes **frm_xfrd** to keep track of successful outgoing transfers by destination. When a transfer succeeds, the destination path is recorded under the presumption that the transfer command created the path at the destination. Subsequent transfers using the path (or some part of it) are told which portion of the path should already exist at the destination by having **$MDP** provide the offset into the destination path of where the first unseen directory occurs. If all directories should have been created the **$MDP** offset refers to the slash just before the file name.

9) The **oss** component in **XRootD** provides an automatic staging facility where missing files on local disk can be automatically fetched from a remote location. This is done using the **oss.remoteroot** and **oss.stagecmd** directives along with the stage option on the **all.export** directive. Refer to the **ofs**/**oss** reference for more information.

10) The **frm_xfrd** creates queue files that describe pending requests in the same partition as the **adminpath**. If you need to place these files in a larger partition or, perhaps, in a more reliable partition, use a symbolic link at the "Queues" directory (see the next section) to point to where the queue files are to be placed. Alternatively, specify the path using the **qcheck** directive. Be aware that this directive allows all clusters running on the same machine to use the same queue file and consequently the same **frm_xfrd**.

11) If the **copycmd** causes data to flow back through the initiating **XRootD**, you should specify the **noalloc** option to prevent **frm_xfrd** from interfering with the **XRootD** in terms of file creation.

12) Monitoring records are created when **migr** (outgoing data) or **stage** (incoming data) is specified on the **xrootd.monitor** directive. The **xpd** option of the **copycmd** allows you to include additional information from the copy command, typically a script wrapper. When **xpd** is specified, the first line of output (i.e. the first line ending with a new line character, '\n', or physically the only line) is added after the 'pd=' tag in the record. The data in the line should adhere to standard CGI syntax to allow for uniform processing.

4.2.1   Directives used but documented in the "ofs/oss Reference"

ofs.osslib *libpath* [*parms*]

> *libpath* is the absolute path to the shared library that contains the implementation of the storage system interface that **frm_xfrd** is to use for file system specific storage operations (e.g., create, rename, etc). The default is to use a built-in mechanism that is identical to what **XRootD** uses by default.

oss.localroot *lpath*

> *lpath* is the path that must prefix any logical file name before using it as a local physical file name (i.e. to map *lfn* to local *pfn*). By default, no prefix is used. Also see the **oss.namelib** directive.

oss.namelib *npath* [*parms*]

> *npath* is the absolute path to the shared library that contains the implementation of the name-to-name translation interface that **frm_xfrd** is to use to convert logical file names to local and remote physical file names. The default is to use a built-in mechanism that relies on the **oss.localroot** and **oss.remoteroot** directives.

oss.rmoteroot *rpath*

> *rpath* is the path that must prefix any logical file name before using it as a remote physical file name (i.e. to map *lfn* to Mass Storage System *pfn*). By default, no prefix is used. Also see the **oss.namelib** directive.

oss.xfr deny *dt*

> *dt* specifies how long a ".fail" file may block a transfer. When a transfer fails file *fn*, a "*fn*.fail" file is created. The presence of this file prevents additional transfer to be attempted for the file. After *dt* seconds have elapsed, the fail no longer prevents a transfer attempt. The *dt* may be suffixed by **h**, **m**, or **s** to indicate **h**ours, **m**inutes, or **s**econds (the default), respectively. The default is 3 hours.

4.2.2   Directives used but documented in the "xrd/xrootd Reference"

all.adminpath *apath*

> *apath* is the fully qualified administrative base path where various directories
> and special files may be created to control execution. The default is "/tmp".
> The **–n** option augments the base path.

all.pidpath *apath*

> *ppath* is the fully qualified base path where the **frm_xfrd.pid** file is to be
> written. The default is "/tmp". The **–n** option augments the base path.

xrootd.monitor dest stage *host:port*

> *host:port* is the logging destination for statistics about incoming transfers. By
> default, incoming transfers are not externally logged.

4.3   Created Files

The following files are created by the **frm_xfrd**:

Path	Type	Modified by	Purpose
/tmp/[*name*/]frm/CIDS	File	**adminpath** or **qcheck** and **–n** option	Cluster ID checkpoint file.
/tmp/[*name*/]frm/frm_xfrd.lock	File	**adminpath** or **qcheck** and **–n** option	Local **frm_xfrd** execution lock.
/tmp/[*name*/]frm/xfrd,udp	UDP Socket	**adminpath** or **qcheck** and **–n** option	Local **frm_xfrd** event notifications
/tmp/[*name*/]frm_xfrd.pid	File	**pidpath** and **–n** option	Holds the process id of the associated frm_xfrd server
/tmp/[*name*/]frm/Queues/	Dir	**adminpath** or **qcheck** and **–n** option	Holds queue files and lock files for maintain queued requests.
/tmp/[*name*/]cmsd.superpid	File	**pidpath** and **–n** option	Holds the process id and the local path prefix (i.e., **localroot**) for a supervisor **cmsd**.
/var/adm/frm/core/[*name*/]core	File	**–n** option	Core file via default in **StartXRD.cf**
/var/adm/frm/logs/[*name*/]xfrlog	File	**-l** option and **–n** option	Log file via default in **StartXRD.cf**

The **adminpath** directive specifies the directory where the remaining files are written. For certain files, the **frm.xfr.qcheck** directive overrides the **adminpath** directive. The -**n** option specifies the **frm_xfrd** instance name. If specified, the instance name is automatically suffixed to the **adminpath** or /tmp, as shown by "[*name*/]." The path is not augmented if it was specified via the **frm.xfr.qcheck** directive. A directory is also create in the current working directory for core files and the log file destination is modified by inserting "[*name*/]." in the destination specified by the **–l** option. If necessary, the directory is created.

The "Queues" directory contains numerous files that track pending requests. The files in this directory are also modified by **frm_xfragent**.

4.4 Logged Transfer Statistics

When the **copycmd** directive specifies **stats**, transfer statistics are written to the log whenever the command successfully transfers a file. The preferred alternative to getting this information is to use the **frm.all.monitor** directive. This mechanism provides a comprehensive way to collect transfer statistics. The format is:

```
yymmdd hh:mm:ss tnum cpy: fsz qt: qsec xt: xsec up: usr lfn
```

Where

yymmdd hh:mm:ss
>   *y*ear, *m*onth, *d*ay, *h*our (24-hour time), *m*inute, *s*econd the transfer completed.

tnum   thread number used in the transfer.

cpy   the token **Got:** for and incoming and **Put:** for an outgoing transfer.

fsz   the size of the file in bytes.

qsec   the number of seconds the transfer waited to start (i.e., queue time).

xsec   the number of seconds the transfer command actually ran (i.e., transfer time).

usr   the identity of the client that requested the transfer.

lfn   the transfer source or target logical file name.

4.5   Temporarily Stopping frm_xfrd

Each time **frm_xfrd** is about to schedule a transfer operation, it check whether a *stopfile* exists indicating that either an incoming or outgoing transfers are not allowed. These *stopfiles* are:

> **/***apath*[**/***name*]**/frm/STOPCOPYIN**
>
> **/***apath*[**/***name*]**/frm/STOPCOPYOUT**
>
> **/***apath*[**/***name*]**/frm/STOPMIGR**
>
> **/***apath*[**/***name*]**/frm/STOPSTAGE**

Where

apath   comes from the **all.adminpath** directive or its default.

name   comes from the –n command line option and is empty if not specified.

The presence of each file stops a particular type of transfer stream, as follows:
- **STOPCOPYIN** file suspends new incoming url-based transfers,
- **STOPCOPYOUT** suspends new outgoing url-based transfers,
- **STOPMIGR** file suspends new outgoing non-url transfers, and
- **STOPSTAGE** suspends new incoming non-url transfers.

In-progress transfers are allowed to complete.

4.6 The frm_xfrd Notification Messages

When **frm_xfrd** is asked to send a notification it uses one or two message formats as shown in each diagram below.

Successful:	`optype` **OK** `lfn`
Unsuccessful	`optype` {**ENOENT** | **BAD**} `lfn` [`emsg`]
`optype`:	**get** | **migr** | **put** | **stage**

Notification: file:////*path*

Where

optype is the name of the completed operation.

lfn   logical filename of the file that was the source or target of the operation.

ENOENT
        staging failed because the file could not be found.

BAD staging failed for other reasons (see *emsg*).

emsg an optional error message describing the nature of the failure.

Successful:	**ready** `requestid msg lfn`
Unsuccessful:	**unprep** `requestid msg lfn`

Notification: **udp://***rhost***:***port***/***msg*

Where

requestid
        the requestid associate with the staging request.

msg   the message, if any, present in the notification string.

lfn   the logical filename of the file that successfully staged in or whose staging failed.

5   The frm_xfragent Command Interface To frm_xfrd

```
frm_xfragent [ options ]

options: [-c cfn] [-d] [-k {num | sz{k|m|g} | sig}]

         [-l [=]logfn] [-n name] [-z]

sig:      fifo|hup|rtmin|rtmin+1|rtmin+2|ttou|winch|xfsz
```

Function

   Add, delete, and list entries in the **frm_xfrd** request queue.

Options

-c *configfn*

   The configuration file name. The default name for the configuration file
   comes from the environmental variable **XRDCONFIGFN**, if set[1], otherwise it
   defaults to "/opt/xrootd/etc/xrootd.cf".

-d   Turns on debugging mode. Additional information is printed to describe
   various actions.

-k *num | sz{k|m|g} | sig*

   Keep no more than *num* old log files. If *sz* is specified, the number of log files
   kept (excluding the current log file) is trimmed to not exceed *sz* bytes. The *sz*
   must be suffixed by **k**, **m**, or **g** to indicate **k**ilobytes, **m**egabyte, or **g**igbytes,
   respectively. If a *sig* value is specified (i.e. **hup** etc), then an external program
   is expected to handle log file rotation (e.g. logrotate). Except for **fifo**, the
   argument specifies signal that causes the daemon to close and re-open the log
   file to allow rotation to occur. When fifo is specified, the daemon waits for
   data to appear on a fifo whose path is identical to the log file path but whose
   name is prefixed by a dot. Refer to the notes for manual rotation caveats.

---

[1] The cmsd and xrootd set this environmental variable if passed the **–c** command line option.

-l [=]*logfn*

> Routes error messages and any trace output to *logfn*. By default, messages are directed to standard error unless the environmental viable **XRDLOGDIR** is set[2], in which case the *logfn* defaults to "**$XRDLOGDIR/frm/xfralog**". Refer to the notes on how *logfn* is modified by the **–n** option. When *logfn* is prefixed by an equals sign, the *logfn* is not qualified by the instance name, if any. This allows log files to be handled in an arbitrary manual way.

-n *name*

> Assigns *name* to the **frm_xfragent** instance. The default *name* comes from the environmental variable **XRDNAME**, if set[3], otherwise **frm_xfragent** is unnamed. When **frm_xfragent** is used as a **cmsd** or **xrootd** service, you need not specify **–n** as it defaults to the underlying system's instance name. See the notes for additional information.

-z   provides microsecond resolution for log file message timestamps.

Notes

1) The **frm_xfragent** also accepts additional directives from the configuration file, *configfn*. These directives are described in the following section.
2) The **–n** option allows you to run multiple instances of **frm_xfragent** with a common configuration file. This is possible because the specified name is used to modify various file system paths **frm_xfragent** uses for output files. By automatically differentiating such paths by instance name prevents two **frm_xfragent** processes from interfering with each other.
3) Generally, you will only need to specify –n when you invoke **frm_xfragent** outside the context of a **cmsd** or **xrootd** server. For instance, you may wish to write scripts the use **frm_xfragent** to extend services outside the **cmsd** or **XRootD** context.
4) The **–n** option modifies the *logfn* specified on the command line. Assuming *logfn* is composed of "*/path/fn*" and **–n** *name* is specified, the log file name becomes "*/path/name/fn*".

---

[2] The cmsd and xrootd set this environmental variable if passed the **–l** command line option.
[3] The cmsd and xrootd set this environmental variable if passed the **–n** command line option.

5) The **frm_xfragent** command reads requests from standard input and provides responses via standard out. This is compatible with the requirements of the **xrootd**'s **oss.stagecmd** and **oss.stagemsg** directives and **cmsd**'s **cms.prep** and **cms.prepmsg** directives. These directives allow you to easily use the built-in version of **frm_xfragent**. Refer to each respective manual for more information.

6) The **frm_xfragent** command can also be used with any other program to manage the transfer queue. Refer to the description o**f frm_xfrd** command for more information.

7) The **frm_xfragent** command has no specific directive. All the information that it needs is to execute properly is tied to **frm_xfrd** directives.

8) The **cmsd** and **xrootd** have a built-in **frm_xfragent**. These daemons do not require that you configure **frm_xfragent** to use **frm_xfrd**.

9) If a log file is specified without a signal **-k** option, the file is closed at midnight, renamed to have a date suffix (i.e., *fn.yyyymmdd*) and possible sequence number (i.e. *fn.yyyymmdd.n*), and a new log file is opened.

10) When a signal value is specified, log files are not automatically renamed at midnight. Instead an external program must be used to properly rotate log files. Make sure to choose a signal that is *not* in use by *any* plug-in. If unsure, choose one of the obscure signal names and monitor for any odd behavior. Otherwise, use the **fifo** option. Be aware that on some non-Linux platforms the fifo file descriptor may leak.

11) When **fifo** is specified the fifo file name must not exists or exist as a fifo file. A simple "**echo x >>** */path/.lfn*" causes the log file to close and reopen.

12) The *sig* names, except for **fifo**, be fully capitalized as well prefixed by "**sig**" or "**SIG**" when capitalized.

5.1   The **frm_xfragent** Requests

The following requests can be directed to **frm_xfragent** via standard-in:

```
Add to queue:       aop[traceid] reqid notify prty mode fn

Remove from queue:  dop requestid

List the queue:     ?[qtype] [varnames]

aop:                < | = | > | + | ^ | &

dop:                - | ~

fn:                 [prot://dest/]lfn[?cgi]

qtype:              op[qtype]

varnames:           lfn | lfncgi | mode | note | prty | qwt |

                    rid | tid | tod | [varnames]
```

5.1.1   Add to queue

traceid
>   a 1- to- 256 character identifier describing the entity that caused the entry to be added. See the notes for an explanation of the standard *traceid* format. A generic default is used if *traceid* is not specified.

op   requests an operation to be performed, as follows :
  <   - copy file *lfn* from a remote location to local disk.
  =   - copy file *lfn* to a remote location and then remove it from local disk.
  >   - copy file *lfn* to a remote location.
  +   - stage file *lfn* from a remote location to local disk.
  ^   - migrate file lfn to a remote location and then remove it from local disk.
  &   - migrate file lfn to a remote location.

requestid
>   a 1- to- 64 character identifier to be used to group this request into a unique set of requests. This means that *requestid* should be globally unique.

notify a 1- to- 512 character string describing who should get notified upon completion of the request. Multiple strings, totaling 512 characters, can be specified by separating each but the last one with a carriage return ('\r') character. See the section on **frm_xfrd** messages for the actual message that is sent. The *notify* format[4] is:

-	no notification is to be sent.
file://*path*	send a special message via FIFO named *path*
udp://*rhost*:*port*/*msg*	send msg via udp to *rhost*:*port*
udp://*path*/*msg*	send msg via udp via local Unix path *path*

prty the character '0', '1', or '2', designating the request's priority (0 being the lowest). If an invalid priority is specified, '0' is used. See **frm_xfrd** on how request priorities are used.

mode The processing mode as a sequence of characters:
  r[*x*] - copy-in a file in read-only mode. See below for the *x* values.
  w[*x*] - copy-in a file in read-write mode. See below for the *x* values.
  The **r** and **w** functions can be suffixed with additional letters to indicate how errors and successes are to be handled. As shown above, *x* can be a combination of the following:
  f   send notification if the file cannot be copied. This option is
          not affected by the q letter.
  s   send notification upon successfully copying the file[5].
  q   do not send any notifications associated with this entry. By default,
          only failure notices are sent.

fn is either a simple logical file name of the file to be copied or a URL describing the copy protocol to be used, the source or destination location, as well as the simple logical file name.

cgi opaque information to be forwarded to the staging transfer agent, if configured to accept this information (see **frm_xfrd**'s **frm.xfr.copycmd** configuration directives). The total amount of information, including the *lfn* and '**?**' separator, cannot exceed 2,175 characters.

---

[4] The old mps_prep and mps_PreStage "mailto://" and "tcp://" notify options are not supported.
[5] The letter 'n' is a synonym for 's'.

Notes

1) Currently, there is no function difference between a copy-in and a stage operation. However, future differences may occur so the correct operation code should be used when adding a file to the queue.
Copy-out operations are done without additional checks other than whether the files changed during the copy operation. Migration operations checks require the file not be modified for a certain period of time and not have

5.1.2 Remove from queue

dop   applies the removal either to the a particular queue, as follows:
     -   - removes requestid from the stage-in and copy-in queues.
     ~   - removes requestid from the migrate and copy-out queues.

requestid
     a 1- to- 64 character identifier to be used to identify the requests that are to be removed. All pending requests associated with this identifier are removed. Requests already in progress remain are allowed to complete even though they are no longer in the queue.

5.1.3 List the queue

qtype by default, the stage-in and copy-in queues are listed. Otherwise, *qtype* lists the particular queues corresponding to the description associated with *op*.

varnames
     is an optional list of variable names, as described below. Information is listed in the order in which the variables are listed; each separated by a space. The default variable is **lfn**. Invalid *varnames* are ignored. Valid variables are:

Var	Information	Var	Information	Var	Information
lfn	logical filename	**obj**	**lfn** or *url*	**qwt**	seconds in queue
lfncgi	lfn?[*cgi* string]	**objcgi**	**lfncgi** or *url*?[*cgi*]	**rid**	requestid
mode	processing opts	**op**	Operation (e.g., '<')	**tid**	traceid
note	notification string	**prty**	priority	**tod**	time of day

6 The hpsscp Transfer Command

```
hpsscp [ options ] [user@host:]sfn [user@host:]tfn

options: [-C] [-d] [-f] [-k keytab] [-m] [-N npath] [-n]

         [-o owner] [-p {r|w|mode}] [-P port] [-Q qpath]

         [-t tries] [-W wcmd] [-x pftpcmd] [-z]

owner:   user[:group] | [user]:group

user:    username  | uid

group:   groupname | gid
```

Function

Copy a file to local disk; typically, from a Mass Storage System.

Parameters

[*user@host:*]*sfn*

name of the source file to read. If it is prefixed by "*user@host:*", then the file must exist in a **pftp** accessible Mass Storage system located at host and reachable via login as user. Otherwise, the file whose name is **sfn** must be present on local disk.

[*user@host:*]*tfn*

name of the source file to write. If it is prefixed by "*user@host:*", then the file is to be written into a **pftp** accessible remote Mass Storage system located at host and reachable via login as user. Otherwise, the file will be written to local disk using the name *tfn*.

targetfn

The name of the file as it should exist on local disk.

Options

-C   continues execution even when a transfer slot (see –Q) cannot be obtained.

-d   Turns on debugging.

-f   over-writes any local file, even if it has zero length. The –f option is the
default for zero-length local files. It has no effect on remote files.

-k *keytab*

specifies the **pftp** password's location. The *keytab* must contain a single
password that will be used when logging in as *user@host*. The default location
is "/var/adm/xrootd/pftp/keyfile".

-m   when *tfn* is local, verifies that *tfn* does not exist. Should *tfn* exist, the command fails
with an error message.

-N *npath*

specifies the location of the local name space directory when **hpsscp** is to
record the directory path used to create a file in **hpss**. This allows **hpsscp** to
avoid issuing duplicate mkdir requests for any paths that exist in *npath*. If **–N**
is not specified, this optimization is suppressed.

-n   does not redirect standard error to standard out when invoking the *xfrcmd*.
This is meant purely for debugging purposes.

-o *owner*

sets the ownership of the file. For non-root processes, the default ownership is
uid/gid of the process. For root process, the default ownership is the uid/gid
assigned to the file.

-p {**r** | **w** | *mode*}

sets the permission bits for the target file. '**r**' implies a mode of 0440, '**w**' a
mode of 0640, and mode is 1- to 4-character octal mode. The default mode is
determined by the *xfrcmd* that is used to transfer the file.

-P *port*

is the port number to contact when logging in using *user@host*. The default
port number is 2021.

-Q *qpath*

> specifies the location of the execution pacing queue directory. The *qpath*
> directory must contain one or more writable zero-length files. Each file
> represents a transfer queue slots implying that the maximum number of
> transfers is equal to the number of files in *qpath*. Simultaneous requests over
> this limit wait until one of the queue slots frees up (also see **–C**). The default
> directory is **/var/hpss/xfrq** and is used if it exists. Transfer pacing does not
> apply if the default directory does not exist.

-t *tries*

> The number of times to try a failing copy when the error indicates a transient
> condition. The default is 2.

-W *wcmd*

> is the program that implements the transfer queue pacing mechanism. The
> default is "/opt/xrootd/utils/wait41".

-x *xfrcmd*

> is the **pftp** command to be used to copy *sfn* to *tfn*. The default is to use
> "/opt/xrootd/utils/pftp_client".

-z   does not establish a new process group prior to executing the *xfrcmd*.

6.1   The hpsscp pftp Command Streams

---

user *user pswd*
delete *mssfn*
mkdir *dirpath*
●
●
●
binary
setpblocksize 2097152
site setcos *cos*
pput *localfn mssfn*
site chuid *uid mssfn*
site chgid *gid mssfn*
quit

Transfer Command Stream When Sending a File to HPSS

Where:

cos	The value specified with the **–s** command line option.
dirpath	The path segment that needs to exist in order to store the file. There are as many **mkdir** commands as there are path segments to ensure that the complete target path exists in **HPSS**.
gid	The gid identified by **–o** command line option. The '**site chgid**' command is only present when *user* if **root**.
localfn	The name of the file to be placed in **HPSS** as it is known on local disk.
mssfn	The name of the file as it must be stored in **HPSS**.
pswd	The password in the file identified by **–k** command line option.
uid	The uid identified by **–o** command line option. The '**site chuid**' command is only present when *user* if **root**.
user	The user contained in "*user@host:tfn*" for the target file.

```
user xfruser pswd
binary
setpblocksize 2097152
pget  mssfn localfn
quit
```

Transfer Command Stream When Retrieving a File from HPSS

Where:

user   The user contained in *"user@host:tfn"* for the target file.

keyfn   The password in the file identified by **–k** command line option.

mssfn   The name of the file as it exists in **HPSS**.

localfn   The name of the file as it must be created on local disk.

7   Document Change History

14 Jan 2009
- New Document.

23 Apr 2009
- Document **frm_admin**.

19 Nov 2009
- Document the **–o** option of **frm_xfr.hpss**.

15 Dec 2009
- Document **frm_purged**.

5 Apr 2010
- Document **frm_xfrd** and **frm_xfragent**.
- This replaces documentation for **frm_pstga** and **frm_pstgd**

12 Apr 2010
- Document the correct *stopfile* names.
- Correct documentation of actual messages sent by **frm_xfrd**.

14 Apr 2010
- Better document actual **frm_xfragent** queue list options.
- Indicate that **frm_xfragent** is built into **cmsd** and **xrootd**.

21 Apr 2010
- Remove the **frm.xfr.qpath** directive and document how to accomplish queue relocation via a symbolic link.
- Document created files.
- Note that **cmsd** and **xrootd** have a built-in **frm_xfragent**.
- Clean-up and reorganization.
- Replace the **failhold** directive with the **oss.xfr** directive as they provide the same information.

6 Jul 2010

- Add **$CID** (cluster ID) and **$INS** (instance name) substitution variables to **frm_xfrd**.
- Document that the **frm_xfrd** transfer queue can be shared by multiple clusters on the same host.
- Extend the **frm.xfr.qcheck** directive to allow the specification of an absolute queue path to be able to implement queue sharing.
- Rename **frm_xfr.hpss** to be **hpsscp** to ease deployment in different contexts.
- Document the **–C**, –N, –Q, and **–W** options enabled for **hpsscp**.

1 Sep 2010

- Document the new **noalloc** option for the **frm.xfr.copycmd** directive.
- Document the new **query xfrq** arguments for **frm_admin**.

7 Oct 2010

- Document how the **STOPURGE** file pauses **frm_purged**.

10 Nov 2010

- Change documentation to reflect the use of extended attributes instead of met-files to control file residency. Most of these changes apply to the **frm_admin** command.

1 Feb 2011

- Minor editorial changes.

8 Mar 2011

- Document the **–b**, **–p**, and **–s** command line options.

30 May 2011

- Document the **chksum** command.

23 Jun 2011

- Document the **frm.all.cnsd** directive.

11 Jul 2011

- Add more descriptive information about purging and staging, especially with respect to federated clusters.

-------------- **Release 3.1.0**

12 Oct 2011
- Document the new **xpd** option on the **copycmd** directive.

3 Dec 2011
- Document the new **frm.all.monitor** directive for **frm_purged** and **frm_xfrd**.

--------------   **Release 3.1.1**

23 Apr 2012
- Correct description of the **–force** option on the **mark** subcommand.

--------------   **Release 3.2.0**
--------------   **Release 3.2.1**
--------------   **Release 3.2.2**
--------------   **Release 3.2.3**
--------------   **Release 3.2.4**
--------------   **Release 3.2.5**
--------------   **Release 3.2.5**

6 Feb 2013
- Remove wrong reference to **$USER** variable.

--------------   **Releases 3.3.0 to 3.3.3**

13 August 2013
- Document the extended **–k, –l** and **–z** command line options.
- Document exported environment variables.
- Document the environment information file contents.
- General clean-up and better explanations.

--------------   **Releases 3.3.4 to 4.11.3**

17 April 2020
- Correct path names of **frm** stop files.
- Document the **split** parameter of the **frm.copymax** directive.
- Remove all mention of the **cnsd** (cluster namespace daemon) as it is no longer supported.

--------------   **Releases 4.12.0 to 4.12.3**

23 June 2020

- Remove references to no longer supported features and directives (e.g. **oss.cache**, **oss.cachescan**, and various meta-data files).
- Remove the now useless **makelf** subcommand and **lockfile** option on the **find** subcommand.