
Fractal Technologies

Managing FracSIS Databases

Best Practices for Administering FracSIS
Databases



www.fractaltechnologies.com

Contents

Administering FracSIS databases	1
What is a FracSIS database?	2
What is a Database system?	2
So what does all this mean to FracSIS?	3
Data management vs. Database management	4
Roles	6
Best practices	9
Performing regular backups	10
Archiving databases between backups.....	10
Verifying database integrity	11
Verifying backup integrity	11
Ensuring Transaction Log security and integrity	12
Handling the Transaction Log during system maintenance	12
Shutting down the server and checkpointing the Transaction Log	13
Moving, renaming, or copying databases	14
Reporting errors	14
ObjectStore utilities	15
osbackup	16
osrestore	17
osarchiv	18
osrecovr.....	19
ossvrstat	20
ossvrping	20
osserver.....	20
ossvrshd	21
ossvrchkpt	21
osdbcontrol.....	22
oscopy	22
osmv	22
osrm	23
oscompact.....	23



SECTION 1

Administering FracSIS databases

This document outlines the procedures necessary to ensure the integrity and consistency of data stored in FracSIS databases. It covers the principles underlying the workings of database systems in general, as well as the specific practices and operations involved in the regular maintenance of FracSIS databases.

In This Section

What is a FracSIS database?.....	2
What is a Database system?.....	2
So what does all this mean to FracSIS?	3
Data management vs. Database management.....	4
Roles	6

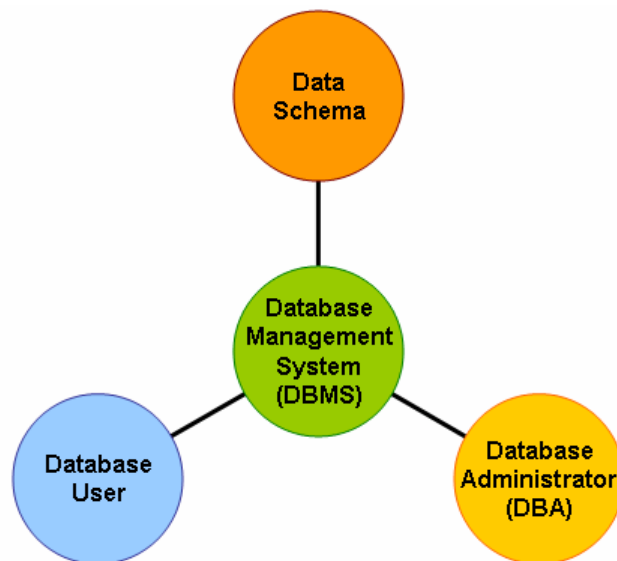


What is a FracSIS database?

Although a FracSIS database file may appear to be nothing more than a file with an **.fsd** extension in a Windows file system, it is actually part of a database system. FracSIS employs ObjectStore, a third party Object-Oriented Database Management System, to manage this database system. FracSIS conforms to a set of specifications and schema defined within the ObjectStore framework – without ObjectStore, a FracSIS database is just an unreadable file.

What is a Database system?

A database system can be thought of as a repository of data administered by a database management system.



The Database Management System usually consists of the physical database, a Database Server, and a Transaction Log.

The Database Server controls all access to the database. All attempted changes to the physical database are relayed to the server, which then makes the changes. The server also ensures that all changes to the physical database are performed as transactions. A transaction is a group of operations that must be performed as a single unit. In other words, these operations should be completed together (committed) or not at all (rolled back). For example, if a modification made by the FracSIS user to a FracSIS object changes several parts of the database, all of these changes are grouped within a single transaction so they are made together



or not at all.

By managing transactions and all access to the physical database, the Database Server ensures that the database is kept in a consistent state in all situations, including hardware failures and system crashes.

For performance reasons, transactions are not written to the physical database immediately. Instead they are written to a Transaction Log, from which they are periodically written – or propagated – to the physical database. The Transaction Log contains a list of all transactions that have modified the data but have not yet been propagated.

The point at which all transactions in the Transaction Log are propagated to the physical database is known as the checkpoint. At this point in time the database is in a consistent state. However, in between checkpoints, it is possible for the database to be in an inconsistent state, since some changes still in the Transaction Log have not yet been written to the physical database. Checkpoints normally occur automatically at a fixed interval, the length of which can be pre-set; it is also possible to force a checkpoint. In the case of ObjectStore, this checkpoint (the propagation of transactions from the Transaction Log to the physical database) takes place every 60 seconds.

When we refer to a FracSIS database, we are not referring to just the physical file (*.fsd) in which the data is stored, but to both the file and the Transaction Log.

So what does all this mean to FracSIS?

A FracSIS database is accessed through an ObjectStore Server process running on the computer hosting the database. In a shared database environment, where one or more FracSIS databases are located on a particular host computer, all access to these databases by FracSIS client applications goes through this ObjectStore Server process. When a FracSIS user makes changes to the data within a FracSIS database, the actual changes to the physical database are made by that ObjectStore Server process. To manipulate any of these databases without understanding the relationship between the database and the ObjectStore Server is inviting disaster.

This is why a controlled system of Database administration (DBA) procedures and guidelines is necessary. Such a system would include how database system backups, restores, and archiving are carried out. For example, a system employing an Oracle database at its back-end is administered by an Oracle DBA, who follows the



Oracle procedures and commands – ObjectStore is no different.

In a shared database environment, where several users may be accessing a FracSIS database through the control of the ObjectStore Server, it cannot be assumed that the database is in a consistent state or that it is not in use by other users. A FracSIS database is not just another Windows file that can be moved, renamed, or deleted in the same manner as text documents. Such Windows commands do not recognise the presence of the ObjectStore Server or its links to FracSIS databases in its control; they are therefore unable to guarantee data consistency when used to manipulate FracSIS databases. This caution also extends to other maintenance and housekeeping procedures such as backing up and restoring databases. Instead, there are a number of ObjectStore commands and utilities that provide equivalent functionality while guaranteeing database consistency.

We recommend that FracSIS databases are kept separate from other files used by other applications on the same server. This reduces the possibility of accidentally affecting FracSIS databases while performing other System Administrator duties. If possible, FracSIS databases should be located on their own dedicated server; if this is not practicable, separate areas should be allocated for the exclusive use of FracSIS databases.

In the simplest FracSIS environment – a FracSIS database hosted on a stand-alone PC and being accessed by a single user – the potential for loss of database integrity is relatively unlikely and easily mitigated. In such an environment, providing that all FracSIS applications have been closed down and a period of at least 60 seconds has elapsed before any manipulation of the physical database file is performed, using either file-system commands or ObjectStore utilities, database integrity will not be compromised since the checkpoint operation will have executed completely.

Data management vs. Database management

There is an important difference between managing data within FracSIS (*Data Management*) and managing FracSIS databases (*Database Management*).

Any user of FracSIS Professional can import data into a FracSIS database, add new data to a database, export data from the database, or even create new databases, either empty or containing data from another existing database. These operations are referred to as Data Management, since all of them are performed within the database under the control of FracSIS and the ObjectStore server.



However, operations involving the control and manipulation of each FracSIS database, such as backup, restore, archiving and relocation, fall within the domain of Database Management. FracSIS itself does not offer the functionality necessary for such operations; they take place under the control of the operating system, using utilities provided by ObjectStore.



ObjectStore is installed as part of any FracSIS installation; therefore any computer where FracSIS is installed will have the necessary ObjectStore utilities. And any computer hosting FracSIS databases should have ObjectStore installed on it.



Roles

FracSIS users can be divided into four groups:

FracSIS Reader users	<p>While FracSIS Reader users can navigate anywhere within a FracSIS database, they can only view data, not change, import, or create data in the FracSIS database. A FracSIS Reader user will never effect changes to a database.</p>
FracSIS Professional Users	<p>FracSIS Professional users can change the content of a FracSIS database, either by importing new data into it, creating new objects within it, or by reorganising the folders and objects within the database. However, all of these changes are made through the control of FracSIS and the ObjectStore Server running on the host computer on which the database is located. As such, they do not need, nor should they be granted, file system level write-access to the physical database files; changes to the physical database are made by the ObjectStore Database System.</p>
FracSIS Data Manager	<p>This role is not enforced by the FracSIS framework – it is a recommended role to ensure that there is one person who has responsibility for controlling how data is stored and managed within FracSIS databases. While FracSIS grants any user of FracSIS Professional the ability to change data in a FracSIS database, we recommend that such operations be monitored and controlled by one employee who is nominated as the FracSIS Data Manager. This person will specify the guidelines for the import, creation, and organisation of data within a FracSIS database.</p> <p>However, the FracSIS Data Manager will not be the only person who can change a FracSIS database; it is an administrative role that specifies the <i>guidelines</i> for these operations. Recommendations for such guidelines are outside the scope of this document.</p>



**FracSIS
database
Administrator**

The FracSIS database administrator is responsible for the management of all FracSIS databases, as well as the functioning of the ObjectStore Server. The FracSIS database administrator must be familiar with the workings of the ObjectStore Server and with the various ObjectStore utilities that can be used on ObjectStore-based databases.

The FracSIS database administrator is the only person allowed to administer the physical database. All FracSIS database management issues should be directed to this person.

The FracSIS database Administrator must maintain a regime of 'Best Practices' to ensure the integrity and consistency of data in FracSIS databases, including:

- Performing regular backups.
- Archiving databases between backups.
- Verifying database integrity.
- Verifying the integrity of backups.
- Ensuring security and consistency of the Transaction Log.
- Correct, safe handling of the Transaction Log during database maintenance.
- Thorough and consistent error reporting.



SECTION 2

Best practices

This section lists the 'best practice' procedures that should be followed by the FracSIS database administrator to maintain ObjectStore and FracSIS data integrity. However, these are broad recommendations rather than the exact syntax of each command – a summary of these commands is provided in the section ObjectStore Utilities (page 15).

A complete description of these utilities can be obtained from ObjectStore's *Managing ObjectStore*. This document is part of any ObjectStore installation, and is available by default at **C:\ODI\OStore\doc\mo\index1.htm**, or from the ObjectStore website at <http://www.ObjectStore.com>.

In This Section

Performing regular backups	10
Archiving databases between backups	10
Verifying database integrity	11
Verifying backup integrity	11
Ensuring Transaction Log security and integrity	12
Handling the Transaction Log during system maintenance	12
Shutting down the server and checkpointing the Transaction Log	13
Moving, renaming, or copying databases	14
Reporting errors	14



Performing regular backups

FracSIS databases should be backed up regularly. This does not refer to the normal system backup when all files on a computer are backed up. Rather, it refers to the use of the **osbackup** utility provided by ObjectStore to create transactionally consistent backups.

Since **osbackup** uses multi-version concurrency control, databases can be backed up even while they are being used. This means that while querying a database, both users and the backup see a snapshot of data – a database version – as it was some time ago, regardless of the current state of the underlying data. However, where possible, we recommend that backups are done during non-work hours to reduce the possibility of error.

Use **osbackup** to copy FracSIS databases to another online location or to a removable storage device that can be moved to an off-site storage location.

In the event of a hardware failure such as a disk crash, use the **osrestore** utility to restore the above backups to your system.

For more information, see **osbackup** (page 16) and **osrestore** (page 17) in the ObjectStore Utilities section (page 15).

Archiving databases between backups

The **osarchiv** utility enables archive logging - periodic incremental backups to track incremental changes between full backups. All transaction activity for the specified FracSIS databases is recorded by **osarchiv**.

Restoring a backup in the event of a hardware failure will not recover changes made to the database between creation of the backup and the time of the hardware failure. This is where the archive logging comes to the rescue. The transactions contained in the Archive Logs can be used to restore the database to a consistent point just before the failure occurs. Whether or not this level of backup is deemed necessary will depend on the operational policies of your organisation.

For more information, see **osarchiv** (page 18).



Verifying database integrity

The **osverifydb** utility verifies the integrity of the database, specifically the pointers, collections, and ObjectStore metadata stored within the database. For example, if your database contains items for which there are no pointers, these items are effectively garbage items in the database; **osverifydb** will report the occurrence of such items in your database.

Since the occurrence of such garbage items could indicate deficiencies in the FracSIS application, the FracSIS database administrator should report any irregular findings of **osverifydb** to Fractal Technologies at support@fractaltechnologies.com.

For more information, see Chapter 4: Utilities of the *Managing ObjectStore* documentation available from the ObjectStore website <http://www.objectstore.com>.

Verifying backup integrity

There is no point making regular backups of your FracSIS databases if you are not confident or have not proved that these backups contain consistent data that can be used to perform a successful restore.

Many organisations have been lulled into a false sense of security by the knowledge that regular backups are being performed, only to find out at the first major system failure that their 'backups' were not being performed properly and were useless because they could not be restored.

You should regularly restore backups to a test area using **osrestore** and then run **osverifydb** on the restored databases to ensure that they are consistent and usable. If archive logs are being implemented, they can be restored using **osrecover**.

For more information, see **osrestore** (page 17) and **osrecover** (page 19).



Ensuring Transaction Log security and integrity

A FracSIS database is not just the physical database file (*.fsd) - it consists of both the file and the Transaction Log.



All databases controlled by an ObjectStore server share the same Transaction Log.

We recommend that you put the Transaction Log on its own disk partition to prevent other applications from filling up the hard disk. If the hard disk on which the Transaction Log is located is full, ObjectStore will be unable to continue.

The Transaction Log should be accessible only to the FracSIS database administrator. It should be located in an area where it is least likely to be accidentally deleted or destroyed. If the Transaction Log is deleted while it contains unpropagated data, this will corrupt one or more of its associated databases. It is very likely that these databases will then be unrecoverable.

Handling the Transaction Log during system maintenance

When performing regular system maintenance, the Database Administrator must ensure that:

- the ObjectStore server has been properly shut down, and
- the Transaction Log has been checkpointed.

These procedures must also be performed when moving, renaming or copying FracSIS databases.



Shutting down the server and checkpointing the Transaction Log

Databases can be taken offline using the **osdbcontrol** command. This command terminates all FracSIS client processes accessing a database, after which any unpropagated data in the Transaction Log is propagated to the physical database file. Note that **osdbcontrol** is only available in ObjectStore Version 6.0 SP 8 or later. Use **osversion** to check what version of ObjectStore you are running.

An alternative to using **osdbcontrol** is to do the following:

- 1 Run **ossvrstat** to see which client processes are using the ObjectStore server. Ensure that all FracSIS client processes have been terminated.
- 2 Shut down the ObjectStore server process using **ossvrshtd**.

Note that since FracSIS installations recommend the installation of the ObjectStore server as a Windows Service, it is possible to both shut down and restart the ObjectStore Server process using the Windows Services interface (**Start > Settings > Control Panel > Administrative Tools > Services**) or by typing the following at a DOS prompt:

```
net stop "ObjectStore Server R6.0"
```

This will automatically invoke **ossvrchkpt**, propagating everything in the transaction log to the physical database.

- 3 Run **ossvrping** to confirm that the server is no longer running.

Another alternative is to use **ossvrchkpt** directly but it should be noted that while this will safely checkpoint the transaction log and preserve database integrity, it will not terminate any ObjectStore clients that are accessing the database. If any FracSIS clients are accessing the database during or after **ossvrchkpt**, the database may not contain the clients' most recently committed transactions until the next scheduled propagation by the ObjectStore server or by the next forced checkpoint.

If FracSIS database files are to be handled at a file-system level (for example, to be copied or renamed), the process outlined above of first taking the database offline, then shutting down and checkpointing the ObjectStore Server, is the only guaranteed safe way of carrying out such an operation. Although it is, in theory, safe to manipulate the database at file-system level once all users have released a database and a safe period has been allowed – at least 60 seconds – to allow the transaction log to be checkpointed by the server, it must be stressed that this requires the Database Administrator to ensure, by means of human intervention, that



nobody reopens the database during this period. This method is nowhere as reliable as shutting down the ObjectStore server to disable interference.

For more information, see `osdbcontrol` (page 22), `ossvrstat` (page 20), `ossvrshd` (page 21), `ossvrchkpt` (page 21), and `ossvrping` (page 20).

Moving, renaming, or copying databases

These operations should be avoided unless absolutely necessary.

If they must be performed, take the database offline, shut down the server (page 13), and checkpoint the Transaction Log. Failure to do so can result in loss or corruption of data.

For example, if an ObjectStore Server is terminated and a FracSIS database is moved without checkpointing the Transaction Log, then when the ObjectStore Server is restarted it will no longer be able to associate the database with its old Transaction Log since it now has a new name or address. Consequently, any unpropagated transactions for that database in the Transaction Log will be lost and the database's integrity will be compromised.

Reporting errors

If the FracSIS database gets corrupted or the ObjectStore Server fails to restart, copy the following files then contact support@fractaltechnologies.com:

- The transaction log (**C:\OSSERVER.LOG** by default).
- All the databases under the ObjectStore Server.
- The **%OS_TMPDIR%\osserver.txt** file.

If **osverifydb** reports inconsistencies in a database, record and report them to support@fractaltechnologies.com.



SECTION 3

ObjectStore utilities

This section does not provide details of the command-line syntax for all available ObjectStore utilities that can be used to administer FracSIS databases, only the most common ones. Only brief descriptions of the purpose of each command-line utility, its recommended usage and, where possible, brief examples are given; refer to the ObjectStore documentation if you require detailed explanations of command line arguments and switches.

Full details of all ObjectStore utilities are available from Chapter 4: Utilities of the *Managing ObjectStore* document (this document is part of any ObjectStore installation and is available by default at **C:\ODI\OSStore\doc\mo\index1.htm**, or from the ObjectStore website at <http://www.objectstore.com>).

In This Section

osbackup	16
osrestore.....	17
osarchiv	18
osrecovr.....	19
ossvrstat	20
ossvrping	20
osserver.....	20
ossvrshd	21
ossvrchkpt	21
osdbcontrol.....	22
oscopy	22
osmv	22
osrm.....	23
oscompact	23



osbackup

The **osbackup** utility copies specified databases to another online location or to tape; it creates an image file (*.img), which can contain the image of one or more databases. A suggested strategy for naming this image file is that it should contain the date of the backup – in this manner, several backup images can be kept rather than overwriting the previous image every night. A suggested format would be:

```
Drive:\chosen_path\project_date.img
```

where **project** and **date** are the project for which the image is being made and the date of the backup, respectively.

A simple format for a command to perform a backup is:

```
osbackup -f image_file_name dbase1 dbase2 dbaseN
```

For example,

```
osbackup -f E:\Backups\Kambalda_20030516.img  
db1.fsd db2.fsd db3.fsd
```

This creates an image called **Kambalda_20030516.img** containing backed up data for the FracSIS databases **db1.fsd**, **db2.fsd**, and **db3.fsd** on the backup device **E:** in the **Backups** folder.



osrestore

The **osrestore** utility restores databases from an image in a backup storage location to your disk. The **osrestore** utility needs to be told the name of the image file containing the databases to be restored.

One or more databases contained in the image can be restored. Make sure that **osrestore** does not overwrite the existing database by first moving or renaming the existing database using **osmv** (page 22).

A simple format for a command to perform a full restore of an image is:

```
osrestore -f image_file_name
```

For example,

```
osrestore -f E:\Backups\Kambalda_20030516.img
```

To restore only two files from the above image, the command would be:

```
osrestore -f E:\Backups\Kambalda_20030516.img  
orig_db_locn1 restored_db_locn1 orig_db_locn2  
restored_db_locn2
```

where **orig_db_locn*** indicates the original location of the databases when backed up and **restored_db_locn*** indicates the location where it should be restored.

If you are unsure of the original locations of the databases stored in the backed up image, use the **-t** option to list them. For example,

```
osrestore -t -f E:\Backups\Kambalda_20030516.img
```



osarchiv

The **osarchiv** utility records all transaction activity for specified databases. You can run this utility interactively or in the background.

Once a nightly backup has been performed, use **osarchiv** to log the changes that take place from that point until the next backup is performed. In the event of data loss, you can restore the previous night's backup, and then revert to the state just prior to the failure by using the incremental changes logged in the archive file.

The easiest way to implement **osarchiv** is to create an incremental record file when the backup is performed. Passed as an argument to **osarchiv**, this provides it with a starting point from which to start incremental logging.

For example,

```
osbackup -i bck_rec.txt -f
E:\Backups\Kambalda_20030516.img db1.fsd db2.fsd
db3.fsd
```

```
osarchiv -a bck_rec.txt -d arch_img -I list
db1.fsd db2.fsd
```

In the above example, the databases **db1.fsd** and **db2.fsd** are first backed up during the nightly **osbackup** to an image called **E:\Backups\Kambalda_20030516.img** while creating an incremental record log called **bck_rec.txt**.

Then, the **osarchiv** is started and commences logging all incremental changes made to **db1.fsd** and **db2.fsd** into the archive image **arch_img**, using **list** as its list of archived database paths.



Perform **osarchiv** immediately after **osbackup**, making sure that there is no client access to the database. This will minimize the number of extra clusters that **osarchiv** has to archive in order to cater for any changes made in the period between completion of **osbackup** and commencement of **osarchiv**.

The example shown above describes creating incremental archive logs in their simplest form; the process can, however, be a bit more involved. If further details are required, please refer to the *Managing ObjectStore* (page 15) documentation or contact support@fractaltechnologies.com for further advice.



osrecover

The **osrecover** utility copies, or rolls forward to the database, modifications from archive log files created by osarchiv.

Just as osarchiv (page 18) complements osbackup (page 16) to provide a lower-level, more detailed backup of incremental changes made since the nightly backup, osrecover is used in conjunction with osrestore (page 17) to restore incremental changes after restoring the backed up image.

Continuing with the previous example, we would issue the following commands:

```
osrestore -t -f E:\Backups\Kambalda_20030516.img
```

```
osrestore -f E:\Backups\Kambalda_20030516.img
orig_db_locn1 restored_db_locn1 orig_db_locn2
restored_db_locn2
```

```
osrecover -c -F archive_list orig_db_locn1
restored_db_locn1 orig_db_locn2 restored_db_locn2
```

where **orig_db_locn*** indicates the original location of the databases and **restored_db_locn*** indicates the location where it should be restored.

In the above example, the first command uses osrestore to produce a list of the original locations of the databases stored in the backed up image. The next command again uses osrestore to restore the nightly backup image

E:\Backups\Kambalda_20030516.img, specifying, in pairs, the original database locations along with the locations to which the databases must be restored. Finally, the osrecover command is used to restore the incremental change log created by osarchiv. To do this, it uses a text list called **archive_list**, prepared earlier. This is simply a list of the archive file(s) used to store the archived incremental changes.

Once again, what is shown in the above example is the use of osrecover in its simplest form. If more complex operations are to be performed, refer to the *Managing ObjectStore* (page 15) documentation or contact support@fractaltechnologies.com for more detailed examples.



ossvrstat

The **ossvrstat** utility can be used to display settings of server parameters, server use meters, and information about each client connected to the server running on the specified host. However, in its most common usage, it is used to check whether an ObjectStore server is running on a particular host (although `ossvrping` (page 20) could probably be more convenient for this) and if there are any clients connected to it.

This operation is used as follows:

```
ossvrstat hostname
```

ossvrping

The **ossvrping** utility reports whether a server is running on the specified host. If you are having problems with a server, first run `ossvrping` to see if the server is running.

This operation is used as follows:

```
ossvrping [-v] hostname
```

where `hostname` specifies the name of the host on which you want to know if a server is running.

The **-v** (verbose) option can be used to indicate that you want more information when a server is not running on the specified host. If you do not specify a host, the default is the local host.

ossserver

The **ossserver** utility starts the ObjectStore server. However, since FracSIS is currently available on Windows operating systems, Fractal Technologies recommends that ObjectStore be run as a Window Service, as it is more convenient to start the ObjectStore server process using the Windows Services (**Start > Settings > Control Panel > Administrative Tools > Services**) interface or by issuing the following command from a DOS command line:

```
net start "ObjectStore Server R6.0"
```



More details on the use of osserver is available from the *Managing ObjectStore* (page 15) documentation.

ossvrshd

The **ossvrshd** utility immediately shuts down the server running on the specified host. This happens regardless of whether clients are connected to the server. Before shutting down the server, run the `ossvrstat` (page 20) utility to determine whether there are clients using the server. If there are, notify them to exit.

It is used with the following syntax:

```
ossvrshd hostname
```

Shutting down the server automatically propagates everything in the Transaction Log.

ossvrchkpt

The **ossvrchkpt** utility performs a checkpoint for a specified server host. It ensures that all data is copied from the server host's Transaction Log to the database or databases that were changed.

It is used with the following syntax:

```
ossvrchkpt hostname
```

where `hostname` is the name of the server whose log you want to propagate.

Note that it is important to check the return-code of this command to ensure that propagation has taken place successfully. The command can return the following values:

Value	Meaning
0	Success
1	There was an error while passing the command to the ObjectStore server
2	The server could not complete the checkpoint



osdbcontrol

Databases can be taken offline using the **osdbcontrol** command. This command terminates all FracSIS client processes accessing a database, after which any unpropagated data in the Transaction Log is propagated to the physical database file.

Note that **osdbcontrol** is only available in ObjectStore Version 6.0 SP 8 or later. Use **osversion** to check what version of ObjectStore you are running.

oscopy

The **oscopy** utility makes a copy of an ObjectStore database. The major reason for always using **oscopy** instead of file system-based copy commands, such as **copy** on Windows, is that **oscopy** will perform transaction-consistent database copying without incurring locking conflicts. In other words, it makes allowances for the possibility that the database is in use and will not compromise the integrity of the data in the database. However, the onus is still on the user of **oscopy** to first check that the database is not in use in order to ensure that users' changes are wholly retained.

The syntax is:

```
oscopy source target
```

osmv



Fractal Technologies recommends that this utility is not used unless absolutely necessary since the risks of such operations, however minimal, usually outweigh the potential benefits.

The **osmv** utility moves a database, directory, or link. Analogous to the **mv** command available in Unix, it can be used to move or rename an ObjectStore database or its link. However, as with the case with **oscopy** (page 22), it will do so in a transaction-consistent way, so as to guarantee database integrity.

Its syntax is as follows:



```
osmv [-i] oldname/location newname/location
```

The **-i** option makes the command interactive, prompting the user for confirmation and is therefore recommended.

osrm

The **osrm** utility removes databases from the server.

Its syntax is:

```
osrm [-f] [-i] [-r] pathname
```

The **-f** option forces execution and does not prompt the user.

The **-i** option is interactive and is highly recommended.

The **-r** option recursively removes all databases in the specified directory.

oscompact



At the time of writing, there are known deficiencies in the version of this utility being supplied with ObjectStore. Fractal Technologies recommends that you do not use this utility without first contacting support@fractaltechnologies.com for advice.

