24 DBFS Access Using OFS

You can access Database File System (DBFS) using the Oracle File Server (OFS) process.

This chapter provides details about how OFS uses OFSD, a dedicated background process, to manage DBFS. It also provides details about how you can access and manage DBFS.

To access a newly created DBFS across multiple nodes where there are no Oracle Client installations, use OFS to NFS mount the file system. In the absence of an Oracle Client installation, use OFS to mount the newly created DBFS to NFS and use it across multiple nodes. All file system requests are served by threads in the OFS background process.

- About OFS
  Oracle File Server (OFS) addresses the need to store PDB specific scripts, logs, trace files and other files produced by running an application in the database.
- About Oracle File Server Process
  OFS manages the database file system using a non-fatal and dedicated background process called Oracle File Server Deamon (OFSD).
- OFS Configuration Parameters
- OFS Client Interface
  The OFS interface includes views and procedures that support the OFS operations.
- Managing DBFS Locally Using FUSE
  Understand how you can manage DBFS using Filesystem in User Space (FUSE).
- Accessing DBFS and OFS with an NFS Account

24.1 About OFS

Oracle File Server (OFS) addresses the need to store PDB specific scripts, logs, trace files and other files produced by running an application in the database.

Additionally, you can use OFS for the following tasks:

- As a staging area where you can host the source data before it is loaded into database tables.
- To store import or export files from Oracle Data Pump process.

Ensure that you do not place core database files such as data, redo, archive log files, and database trace file on OFS as this can produce a dependency cycle and cause the system to hang. Similarly, the `diagnostic_dest` initialization parameter that sets the location of the automatic diagnostic repository should not point to a directory inside OFS.

OFS provides methods and procedures to allow you to create a Database file system using storage that is part of the PDB. You can mount the created file system, unmount it like any other Unix file system using PL/SQL procedures, and destroy the file system when it is no longer in use. When the PDB is destroyed, the file system is also destroyed, which frees up the underlying storage space.

24.2 About Oracle File Server Process

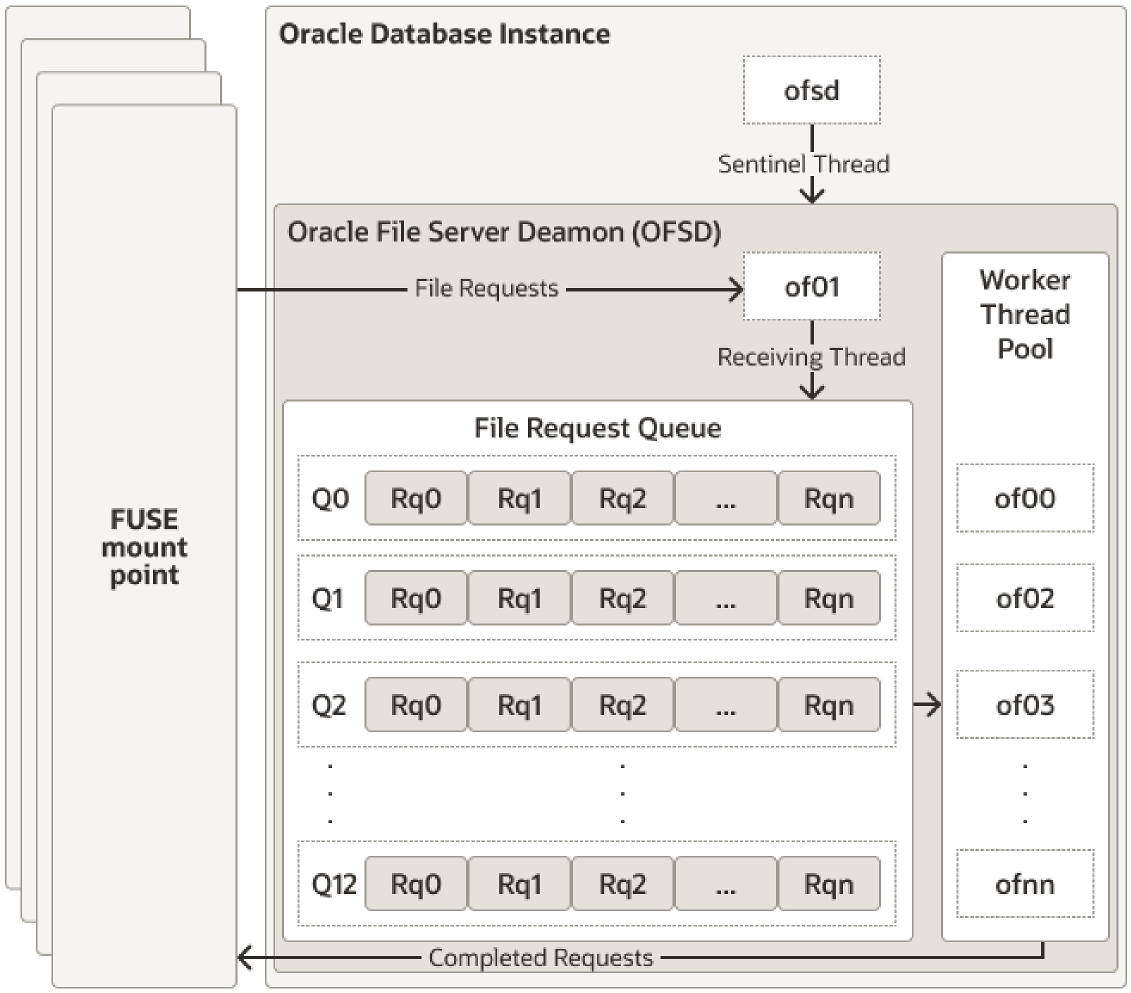
OFS manages the database file system using a non-fatal and dedicated background process called Oracle File Server Deamon (OFSD).

For more information about background process, see Background Processes in the *Database Reference* guide.

When an instance starts, the OFSD process gets spawned on operating system platforms, such as Linux, where OFS is supported. OFSD is multi-threaded and non-fatal. It serves both file system management requests and file requests from each mounted file system.

The centralized server background process model of OFS allows multiple file systems to be mounted and accessed using a limited set of server threads. It allows better resource sharing and a linear scalability with new file server threads created on demand. Both memory and CPU used by these threads are controlled through system wide parameters set in the RDBMS instance.

OFSD process starts two types of threads: receiver thread and worker thread. The receiver thread receives requests from the mounted file system. The name of this thread name is similar to `of01`. The requests received by this thread are placed in a submit queue which is served by different worker threads. The submit queue is hash partitioned to efficiently distribute the incoming requests across all the worker threads. By default, OFSD starts 3 worker threads. You can update the value of the `OFS_THREADS` parameter to increase the number of worker thread.



OFSD supports these 2 types of file systems: DBFS and OFS.

Use the `DBMS_FS` PL/SQL procedures to create, mount, and work with the file systems managed by the OFSD process.

OFSD uses a pool of worker threads to serve requests from multiple file systems that are mounted on the instance. Use `V$OFSMOUNT` to query the mounted file systems. The response that is returned is specific to each PDB. It lists only the file systems that are mounted in the specified PDB.

OFS Configuration Parameters

The following table specifies the parameter that can be tuned to provide file system access for database objects using OFS.

Table 24-1 OFS Configuration Parameters

Parameter Name	Description
`OFS_THREADS`	Set the number of OFS worker threads to handle OFS requests. Enter an integer value in the range of 3–128. The default value of `OFS_THREADS` is 4, which includes 1 receiver thread and 3 worker threads. by default, OFSD starts 3 worker threads. Set the value for this parameter based on the total number of mounted file systems and the rate at which file operations are performed in each file system. You can update the number `OFS_THREADS` to increase the number of `DBMS_FS` requests that are executed in parallel. This increases the number of worker threads executing the make, mount, unmount, and destroy operations on Oracle file systems in the Oracle database. Set this value after careful consideration as you can only increase this value dynamically and you cannot decrease it. Before changing the value, query the `V$OFS_THREADS` view to list all the running OFS threads and to retrieve details about the different OFS threads. For information about the columns and data types of this view, see V$OFS_THREADS in *Oracle Database Reference*. Increasing the value of `OFS_THREADS`, results in a significant reduction of time taken to execute parallel file system requests in environments that contain multiple PDBs. If you set a high value for `OFS_THREADS`, then the specified number of threads are created. If there is no workload, these threads remain in an idle state and wait for new work.

Note:

The `diagnostic_dest` initialization parameter sets the location of the automatic diagnostic repository. When you use `dbfs_client` or Oracle File Server (OFS) as the file system server, ensure that this parameter does not point to a directory inside `dbfs_client` or OFS as this can produce a dependency cycle and cause the system to hang.

24.4 OFS Client Interface

The OFS interface includes views and procedures that support the OFS operations.

Use the PL/SQL procedures provided by OFS to create, mount, unmount, and destroy file systems. You can query the views to identify the state of the mounted file systems and to collect performance data on individual file I/O operations.

- DBMS_FS Package
  Use the `DBMS_FS` package to manage the file systems. Use the procedures in this package to create, mount, unmount and destroy a file system in the Oracle Database.
- Views for OFS
  The views that support OFS operations start with `V$OFS`.

24.4.1 DBMS_FS Package

Use the `DBMS_FS` package to manage the file systems. Use the procedures in this package to create, mount, unmount and destroy a file system in the Oracle Database.

Multiple PDBs can submit these jobs in parallel. The requests are executed in parallel by different worker threads. You can obtain the status of the operations by querying `V$OFS_THREADS`. OFSD manages the threads. It creates new worker threads, when the number of current worker threads are less than the value of the `OFS_THREADS` parameter.

See Also:

Oracle Database PL/SQL Packages and Types Reference for more information about Oracle OFS procedures.

The following example illustrates the use of `DBMS_FS` package.

```
BEGIN
 DBMS_FS.MAKE_ORACLE_FS (
  fstype           => 'dbfs',
  fsname           => 'dbfs_fs1',
  mount_options    => 'TABLESPACE=dbfs_fs1_tbspc');
END;
/
BEGIN
 DBMS_FS.MOUNT_ORACLE_FS (
  fstype           => 'dbfs',
  fsname           => 'dbfs_fs1',
  mount_point      => '/oracle/dbfs/testfs',
  mount_options    => 'default_permissions, allow_other');
END;
/
/************** Now you can access the file system. All the FS operations go
here ***************/

BEGIN
 DBMS_FS.UNMOUNT_ORACLE_FS (
  fsname           => 'dbfs_fs1',
  mount_point      => '/oracle/dbfs/testfs',
  unmount_options    => 'force');
END;
/
BEGIN
 DBMS_FS.DESTROY_ORACLE_FS (
  fstype           => 'dbfs',
  fsname           => 'dbfs_fs1');
END;
/
```

24.4.2 Views for OFS

The views that support OFS operations start with `V$OFS`.

Table 24-2 Fixed Views for OFS

View	Description
`V$OFS_THREADS`	Query this view to list all the running OFS threads and to retrieve details about the different OFS threads, such as the thread type, file system on which the thread is working, time spent by the worker thread on a particular operation. For information about the columns and data types of this view, see V$OFS_THREADS in *Oracle Database Reference*.
`V$OFSMOUNT`	Query this view to retrieve details about the file systems that are mounted by Oracle File System. For information about the columns and data types of this view, see V$OFSMOUNT in *Oracle Database Reference*.
`V$OFS_STATS`	Query this view to list the number of times each file operation has been called for a mount point. For information about the columns and data types of this view, see V$OFS_STATS in *Oracle Database Reference*.

24.5 Managing DBFS Locally Using FUSE

Understand how you can manage DBFS using Filesystem in User Space (FUSE).

The FUSE interface in the Linux kernel makes the file systems available to the operating system processes. After mounting the file system, you can export it, and then NFS mount it on other nodes where client applications can access this file system.

- Configuring FUSE
  OFSD exposes the database file system through FUSE. Before using OFSD to mount the database file systems, you must install and configure the FUSE module.
- Accessing OFS in Cloud
  To access files from an OFS mounted on any Cloud environment, you must perform additional steps to configure the environment.

24.5.1 Configuring FUSE

OFSD exposes the database file system through FUSE. Before using OFSD to mount the database file systems, you must install and configure the FUSE module.

If you are running your database instance in a Compute node, configure the FUSE module in that node. The file system gets mounted and is visible through a mounted path on the compute node. In a RAC configuration, configure FUSE in each node, so that the OFS file systems can be mounted independently in each node.

To configure the FUSE module in Cloud or on-premises environment, where the database instance is running:

1. Set read and execute permissions for an Oracle user to use the FUSE executable file, `fusermount`.

   ```
   sudo chmod o+rx /usr/bin/fusermount
   ```

   Use the `fusermount` file to mount and unmount the FUSE user mode file systems.

2. Set the `setuid` bit on the `fusermount` file to permit an Oracle user to mount file systems.

   ```
   sudo chmod u+s /usr/bin/fusermount
   ```

3. Permit other users to access the mounted file system.

   ```
   sudo sh -c ''echo user_allow_other >> /etc/fuse.conf''
   ```

4. Optional. By default, the maximum number of file systems that you can mount using FUSE is 1000. If you are running a large number of PDBs and need to configure a separate file system for each PDB, then run the following command to increase the number of file systems that can be mounted using FUSE. The following command increases the number of file systems that can be mounted using FUSE to 4000.

   ```
   sudo sh -c ''echo mount_max=4000 >> /etc/fuse.conf''
   ```

5. Allow all users to read the `fuse.conf` file, so that the Oracle process can read this file at run time.

   ```
   sudo chmod a+r /etc/fuse.conf
   ```

24.5.2 Accessing OFS in Cloud

To access files from an OFS mounted on any Cloud environment, you must perform additional steps to configure the environment.

To access files in an OFS mount in the Cloud environment, you may need to perform additional configuration. It may not be possible to export the OFS mount point from database node to client node due to security reasons. This may hinder client applications from accessing the OFS files through operating system commands and utilities and the OFS mount path may not be available to access using system calls. In such situations, Oracle recommends that you use the `utl_file` package to access files in the OFS mount. For information about UTL file package, see Summary of UTL_FILE Subprograms in *PL/SQL Packages and Types Reference*.

You can also use the `impdp` and `expdp` command-line clients to access files in the OFS mount. See Oracle Data Pump Import and Oracle Data Pump Export in the *Utilities guide*.

To configure the environment so that client applications in Cloud can access files in an OFS:

1. Create a directory object using the OFS mount path.

   The following sample code displays how you can create a directory object called `pdb1_ofsdir` when `/u03/dbfs/<pdbid>/data` is the OFS mount directory on the db node.

   ```
   CREATE DIRECTORY pdb1_ofsdir AS '/u03/dbfs/<pdbid>/data/';
   ```

2. Grant access to the user to access the directory object.

   For more information on creating a directory object and setting access permissions on it, see CREATE DIRECTORY in *PL/SQL Packages and Types Reference*.

Do not access the OFS files by directly querying or modifying the DBFS tables. Do not use `dbfs_client` when the DBFS file system is mounted through OFS or else it could lead to metadata and data inconsistency. To access the OFS files, use the `UTL_FILE` package in addition to the procedures listed in the `DBMS_FS` package. See FS_EXISTS and LIST_FILES in *PL/SQL Packages and Types Reference*.

Accessing DBFS and OFS with an NFS Account

NFS is a widely used protocol to access any local file system across network. OFS makes use of this protocol and enables access to any DBFS file system that is mounted on the compute node.

NFS enables the compute node to be accessible across all nodes that are authorized to access the file system.

24.6.1 Accessing OFS with an NFS Account

You can export an OFS mount to a specified list of nodes and NFS mount it on them. This allows users to access the contents of an OFS mount point from a node where the database is not running. The NFS exports may not work in cloud environments due to security reasons, but you can use it in on-premise environments.

NFS v3 is a stateless protocol because of which it encapsulates each `readdir` request between `opendir` and `releasedir` calls. This may lead to poor performance when you want to list directories that have a large number of files. Therefore, OFS maintains a directory cache which persists across the `opendir` and `releasedir` calls. Do not use the `no_rbt_cache` mount option to avoid inconsistent directory cache listing and to utilize the benefits of directory cache.

24.6.2 Prerequisites to Access Storage Through NFS Server

Learn about the prerequisites to access storage through NFS server.

Following are the prerequisites:

- DBFS file system must be created before using OFS.
- You should be able to mount the file systems exported by the database.
- NFS server must be configured with `KERNEL` module.

Note:

The `KERNEL` module is supported through `FUSE` driver for Linux.

24.6.3 NFS Security

OFS uses the OS authentication model to authorize NFS client users. If the user accesses a local node (where the Oracle instance is running), the access to each file in the file system is controlled through **Unix Access Control List** set for each object. For NFS clients, you can configure Kerberos to control access.

On Linux, OFS uses FUSE to receive file system requests from the OS kernel or NFS client. This requires `user_allow_other` parameter to be set in `/etc/fuse.conf` configuration file if an OS user other than the `root` user and oracle user need to access the file system.

Note:

Users can also be configured with an Oracle password to log into Oracle client tools like `SQL* Plus` to execute SQL statements.

If the network is not secure, the customer is advised to setup Kerberos to authenticate the user using OS NFS.

Note:

- The Kerberos authentication is available from NFS version 4 onwards. If the OFS is exported via NFS version 3, then the authentication is performed using `AUTH_SYS`.
- For local node, the authentication is performed using `AUTH_SYS` irrespective of how the OFS is exported (NFS version 3 or NFS version 4).

- About Kerberos
  Kerberos uses encryption technology, Key Distribution Center (KDC), and an arbitrator to perform secure authentication on open networks.
- Configuring Kerberos Server
  To configure a Kerberos Server in a Linux system:

24.6.3.1 About Kerberos

Kerberos uses encryption technology, Key Distribution Center (KDC), and an arbitrator to perform secure authentication on open networks.

Kerberos is the widely used security mechanism that provides all three flavors of security:

- Authentication
- Integrity check
- Privacy

Kerberos Infrastructure consists of Kerberos software, secured authentication servers, centralized account and password store, and systems configured to authenticate through the Kerberos protocol. The OS NFS server handles the complete authentication and integrity checks by using kerberos principal name as the user name. Once the authentication is performed, the requests passed to the Oracle kernel are handled based on the user name passed through the **VFS I/O** request.

24.6.3.2 Configuring Kerberos Server

To configure a Kerberos Server in a Linux system:

1. Install Kerberos software in the Linux system.
2. Check if the daemons are running using the following commands.

```
# /sbin/chkconfig krb5kdc on
# /sbin/chkconfig kadmin on
```

3. If the daemons are not running use the following commands to start the daemons manually:

```
# /etc/rc.d/init.d/krb5kdc start
# /etc/rc.d/init.d/kadmin start
```

4. Add user principal using the `kadmin.local` command.

   Example:

```
kadmin.local: addprinc <scott>
```