70
DBMS_DBFS_SFS

The DBMS_DBFS_SFS package provides an interface to operate a SecureFile-based store (SFS) for the content interface described in the DBMS_DBFS_CONTENT package.

This chapter contains the following topics:

- Overview
- Security Model
- Constants
- Summary of DBMS_DBFS_SFS Subprograms

> **See Also:**
>
> *Oracle Database SecureFiles and Large Objects Developer's Guide*

DBMS_DBFS_SFS Overview

The DBMS_DBFS_SFS package is a sample implementation of a package that implements and extends the DBMS_DBFS_CONTENT_SPI interface. It provides a POSIX-compliant file system stored in the RDBMS.

DBMS_DBFS_SFS Security Model

The DBMS_DBFS_SFS package runs with AUTHID CURRENT_USER.

DBMS_DBFS_SFS Constants

The DBMS_DBFS_SFS package uses the constants shown in the following tables.

> **Note:**
>
> Oracle has deprecated the older encryptions and hashing algorithms. The deprecated algorithms for DBMS_CRYPTO and native network encryption include MD4, MD5, DES, 3DES, and RC4-related algorithms as well as 3DES for Transparent Data Encryption (TDE). Removing older, less secure cryptography algorithms prevents accidental use of these algorithms. To meet your security requirements, Oracle recommends that you use more modern cryptography algorithms, such as the Advanced Encryption Standard (AES).

- Table 70-1

- Table 70-2
- Table 70-3
- Table 70-4

Table 70-1 DBMS_DBFS_SFS Constants - Compression Levels

Constant	Type	Value	Description
`COMPRESSION_DEFAULT`	`VARCHAR2(32)`	`''`	Use the default SecureFile compression level
`COMPRESSION_LOW`	`VARCHAR2(32)`	`'LOW'`	Use compression level `'LOW'`
`COMPRESSION_MEDIUM`	`VARCHAR2(32)`	`'MEDIUM'`	Use compression level `'MEDIUM'`
`COMPRESSION_HIGH`	`VARCHAR2(32)`	`'HIGH'`	Use compression level `'HIGH'`

Table 70-2 DBMS_DBFS_SFS Constants - Used by the encryption Parameter

Constant	Type	Value	Description
`ENCRYPTION_DEFAULT`	`VARCHAR2(32)`	`''`	Use the default SecureFile encryption algorithm
`ENCRYPTION_AES128`	`VARCHAR2(32)`	`'AES128'`	Use encryption AES 128 bit
`ENCRYPTION_AES192`	`VARCHAR2(32)`	`'AES192'`	Use encryption AES 192 bit
`ENCRYPTION_AES256`	`VARCHAR2(32)`	`'AES256'`	Use encryption AES 256 bit

Table 70-3 DBMS_DBFS_SFS Constants - Used by the npartitions Parameter

Constant	Type	Value	Description
`DEFAULT_PARTITIONS`	`INTEGER`	`16`	Default to 16 partitions

Table 70-4 DBMS_DBFS_SFS Constants - Used by the partition_key Parameter

Constant	Type	Value	Description
`PARTITION_BY_ITEM`	`INTEGER`	`1`	Use a hash of the item name for the partition key
`PARTITION_BY_PATH`	`INTEGER`	`2`	Use a hash of the path name for the partition key
`PARTITION_BY_GUID`	`INTEGER`	`3`	Use a hash of the GUID as the partition key

Summary of DBMS_DBFS_SFS Subprograms

This table lists and describes the `DBMS_DBFS_SFS` Package subprograms.

Table 70-5 DBMS_DBFS_SFS Package Subprograms

Subprogram	Description
CREATEFILESYSTEM Procedure	Creates a file system store
CREATESTORE Procedure	Creates a new DBFS SFS store
DROPFILESYSTEM Procedures	Drops the DBFS SFS store
INITFS Procedure	Initializes a POSIX file system store

CREATEFILESYSTEM Procedure

This procedure creates a file system store.

Syntax

```
DBMS_DBFS_SFS.CREATEFILESYSTEM (
   store_name          IN     VARCHAR2,
   schema_name         IN     VARCHAR2     DEFAULT NULL,
   tbl_name            IN     VARCHAR2     DEFAULT NULL,
   tbl_tbs             IN     VARCHAR2     DEFAULT NULL,
   lob_tbs             IN     VARCHAR2     DEFAULT NULL,
   use_bf              IN     BOOLEAN      DEFAULT FALSE,
   properties          IN     DBMS_DBFS_CONTENT_PROPERTIES_T DEFAULT NULL,
   create_only         IN     BOOLEAN      FALSE,
   use_objects         IN     BOOLEAN      DEFAULT FALSE,
   with_grants         IN     BOOLEAN      DEFAULT FALSE,
   do_dedup            IN     BOOLEAN      DEFAULT FALSE,
   do_compress         IN     BOOLEAN      DEFAULT FALSE
   compression         IN     VARCHAR2     DEFAULT COMPRESSION_DEFAULT,
   do_encrypt          IN     BOOLEAN      DEFAULT FALSE,
   encryption          IN     VARCHAR2     DEFAULT ENCRYPTION_DEFAULT,
   do_partition        IN     BOOLEAN      DEFAULT FALSE,
   npartitions         IN     NUMBER       DEFAULTDEFAULT_PARTITIONS,
   partition_key       IN     NUMBER       DEFAULT PARTITION_BY_ITEM,
   partition_guidi     IN     BOOLEAN      DEFAULT FALSE,
   partition_pathi     IN     BOOLEAN      DEFAULT FALSE,
   partition_prop      IN     BOOLEAN      DEFAULT TRUE);
```

Parameters

Table 70-6 CREATEFILESYSTEM Procedure Parameters

Parameter	Description
`store_name`	Name of store
`schema_name`	Schema for the store, defaulting to the current schema
`tbl_name`	Table for store entries. If not specified, an internally generated name is used.
`tbl_tb`	Tablespace for the store, defaulting to the schema's default tablespace
`lob_tbs`	Tablespace in which to create the LOB segment. It defaults to the user's default tablespace.
`use_bf`	If `TRUE`, a BasicFile `LOB` is used; otherwise a SecureFile `LOB` is used.
`properties`	Table of (name, value, typecode) tuples used to configure the store properties. Currently no such properties are defined or used.

Table 70-6 (Cont.) CREATEFILESYSTEM Procedure Parameters

Parameter	Description
`create_only`	If `TRUE`, the file system is created, but not registered with the current user
`use_objects`	If `TRUE`, a single base-table with an object-type column (using a nested table) is created to backup the new file system. Otherwise, a pair of (parent, child) tables is used to backup the file system. In both cases, the object type nested table or the child table is used only for user-defined properties.
`with_grants`	If `TRUE`, DML and query access permissions are granted to the `DBFS_ROLE` as part of creating the file system. Otherwise, explicit grants (or existing permissions) are required to access the file system.
`do_dedup`	If `TRUE`, do deduplication the underlying SecureFile column
`do_compress`	If `TRUE`, do compression the underlying SecureFile column
`compression`	Compression algorithm to use (see Table 70-1)
`do_encrypt`	If `TRUE`, encrypt the underlying SecureFile column
`encryption`	encryption algorithm to use (see Table 70-2)
`do_partition`	If `TRUE`, partition the table used for storage
`npartitions`	Number of partitions to create for the table (see Table 70-3).
`partition_key`	How to partition the table: by item name, by path name, or by GUID (see Table 70-4).
`partition_guidi`	If `TRUE`, build an index on GUID
`partition_pathi`	If `TRUE`, build an index on path name
`partition_prop`	If `TRUE`, partition the properties table

Usage Notes

The procedure executes like a DDL in that it auto-commits before and after its execution.

CREATESTORE Procedure

This procedure creates a new DBFS SFS store owned by the invoking session user.

Syntax

```
DBMS_DBFS_SFS.CREATESTORE  (
   store_name      IN     VARCHAR2,
   tbl_name        IN     VARCHAR2  DEFAULT NULL,
   tbs_name        in     VARCHAR2  DEFAULT NULL,
   use_bf          in     BOOLEAN   DEFAULT FALSE,
   stgopts         in     VARCHAR2 DEFAULT '');
```

Parameters

Table 70-7 CREATESTORE Procedure Parameters

Parameter	Description
`store_name`	Name of store

Table 70-7 (Cont.) CREATESTORE Procedure Parameters

Parameter	Description
`store_type`	`STORETYPE_TAPE` or `STORETYPE_AMAZONS3`
`tbl_name`	Placeholder for the store content cached in database
`tbs_name`	Named tablespace
`use_bf`	If `TRUE`, a BasicFile `LOB` is used; otherwise a SecureFile `LOB` is used.
`stgopts`	Currently non-operational, reserved for future use

DROPFILESYSTEM Procedures

This procedure drops the DBFS SFS store, purging all dictionary information associated with the store, and dropping the underlying file system table.

Syntax

```
DBMS_DBFS_SFS.DROPFILESYSTEM  (
   schema_name     IN      VARCHAR2 DEFAULT NULL,
   tbl_name        IN      INTEGER);

DBMS_DBFS_SFS.DROPFILESYSTEM  (
   store_name      IN      VARCHAR2);
```

Parameters

Table 70-8 DROPFILESYSTEM Procedure Parameters

Parameter	Description
`schema_name`	Name of schema
`tbl_name`	Name of tablespace
`store_name`	Name of store path

Usage Notes

- If the specified store table is registered by the current user, it will be unregistered from the content interface described in the `DBMS_DBFS_CONTENT` package and the POSIX metadata tables.
- Subsequent to unregistration, an attempt will be made to store table(s). This operation may fail if other users are currently using this store table.
- The user attempting a drop of the tables underlying the store must actually have the privileges to complete the drop operation (either as the owner of the tables, or as a sufficiently privileged user for cross-schema operations).
- The procedure executes like a DDL in that it auto-commits before and after its execution.

INITFS Procedure

This procedure initialize a POSIX file system store. The table associated with the POSIX file system store `store_name` is truncated and reinitialized with a single "root" directory entry.

Syntax

```
DBMS_DBFS_SFS.INITFS (
   store_name     IN      VARCHAR2);
```

Parameters

Table 70-9   INITFS Procedure Parameters

Parameter	Description
`store_name`	Name of store

Usage Notes

The procedure executes like a DDL in that it auto-commits before and after its execution.