exFAT and SafeexFAT for Linux FUSE User Guide

Version 1.20

For use with exFAT for Linux FUSE versions 1.02 and above

Table of Contents

1 System Overview

This chapter contains the fundamental information for this module.

The component sections are as follows:

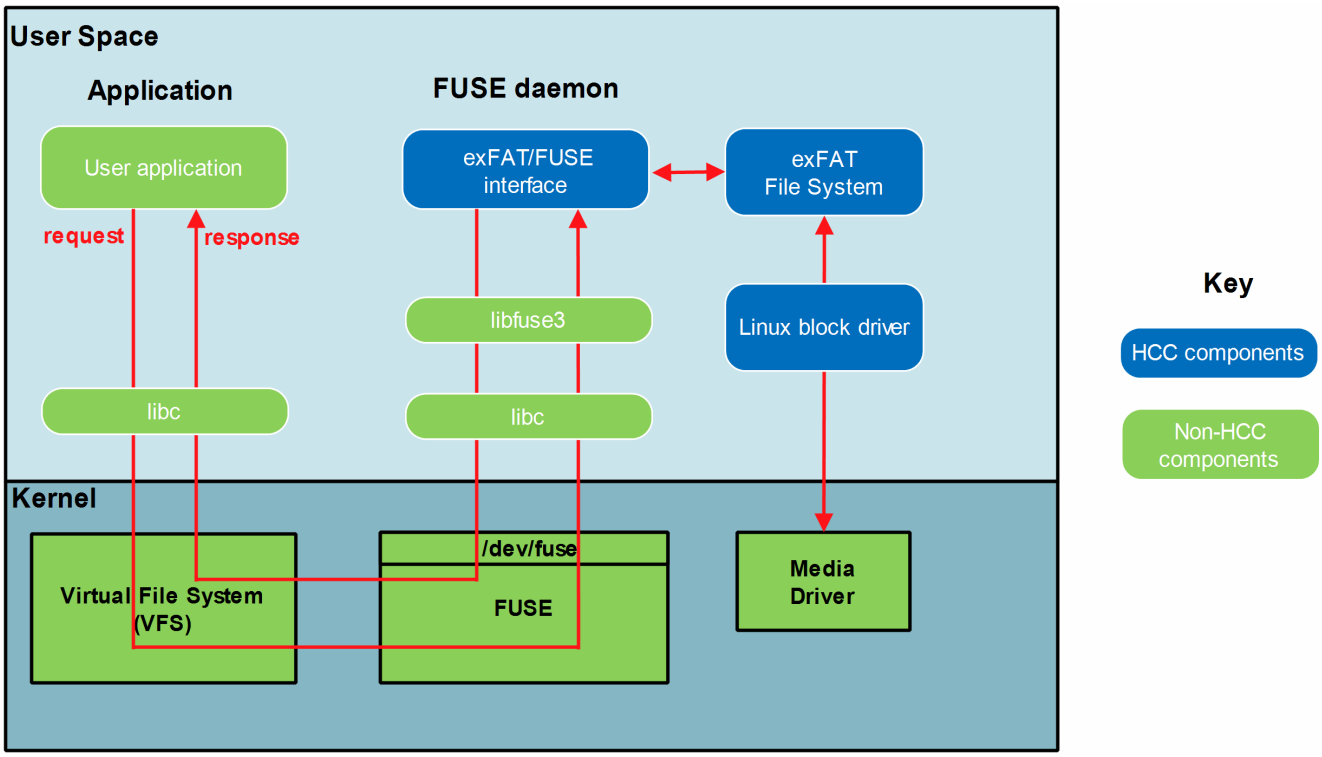
- Introduction – describes the main elements of the module.
- Feature Check – summarizes the main features of the module as bullet points.
- Packages and Documents – the *Packages* section lists the packages that you need in order to use this module. The *Documents* section lists the relevant user guides.
- Change History – lists the earlier versions of this manual, giving the software version that each manual describes.

1.1 Introduction

This guide is for those who wish to implement HCC's exFAT file system (and optionally SafeexFAT as well) on a Linux system. HCC Embedded's exFAT/SafeexFAT for Linux uses the standard FUSE (Filesystem in User Space) library. Use of the SafeexFAT extension makes the file system fail-safe.

This HCC product is a Linux application running in user-space. It is supplied as source code from which a Linux executable can be built. This executable can then be used to mount an exFAT drive whose files can be accessed like any other files in the system.

The diagram below summarizes the FUSE architecture.



This shows the three HCC components in blue:

- The exFAT file system (and optionally SafeexFAT as well).
- The exFAT/FUSE interface - this wrapper that provides the interface from FUSE to HCC exFAT.
- The Linux block driver - converts the HCC media driver interface to a Linux block media driver.

HCC Embedded is a licensed supplier of exFAT implementations and can provide a full technology and patent license solution for incorporation into customers' devices. This means:

- For those who already have a Microsoft license for exFAT, HCC can supply its exFAT software implementation.
- For those who do not have a Microsoft license for exFAT, HCC can provide a Microsoft-approved license for exFAT and supply its exFAT software implementation.

Note: To use this product, the following extra packages are required:

- **fs_exfat** - the HCC exFAT File System. For full details of HCC's exFAT (Extended File Allocation Table) file system, refer to the HCC exFAT File System User Guide.
- **fs_exfat_safe** - the SafeexFAT extension for the file system.
- **media_drv_blk_linux** - interfaces HCC media driver specification to Linux block media driver.

Note:

To understand this document the reader needs a good understanding of the following:

- The Linux kernel.
- Linux distribution (Ubuntu, Debian, and Yocto Project), embedded Linux kernel-space and user-space.
- Mounting a volume and mount points.
- Character and block devices.
- The Kernel module and built-in kernel modules.
- Kernel configuration and compiling.
- The FUSE kernel module and FUSE library.

Note:

- HCC offers hardware and firmware development consultancy to assist developers with the implementation of various types of file system.
- Although every attempt has been made to simplify the system's use, developers must have a good understanding of the requirements of the systems they are designing in order to obtain the maximum practical benefits.

1.2 Feature Check

The main features of Microsoft's exFAT are the following:

- Almost unlimited card storage – exFAT means devices can handle growing requirements for media file storage, raising capacity from 32 GB to 256 TB.
- Handles vast amounts of media in one directory – exFAT can handle more than 100 HD movies, 4000 RAW images, or 60 hours of HD recording in a single directory.
- Interoperability between systems and devices – exFAT supports interoperability between many operating systems, so there's no need to keep reformatting files and media.
- Fast transfer speeds – file saves on SDXC cards can achieve their full speed of 300 MBps.
- Provides an extensible format – this includes parameters that OEMs can define to customize exFAT for specific devices.

The main features of the HCC system are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Fail-safe (if SafeexFAT used), protecting against unexpected reset or power loss.
- Linux FUSE integration available.
- Supports FUSE library version 2.x and FUSE 3.x.
- Multiple instances can be run with multiple volumes, one for each volume used.
- Cache options for optimal performance.
- Code size 35 KB (exFAT) or 47 KB (with SafeexFAT).
- RAM usage >16 KB (exFAT) or >18 KB (with SafeexFAT).
- ANSI ‘C’.
- Unicode 16.
- Multiple open files.
- Multiple users of open files.
- Multiple volumes.
- Multi-sector read/write.
- Variable sector sizes.
- Partition handling.
- Handles media errors.
- Test suite.
- Zero copy.
- Re-entrant.
- Boundary alignment offset for the FAT table.
- Boundary alignment offset for the data region.

1.3 Packages and Documents

Packages

This table lists the packages that need to be used with this module, and also optional modules that may interact with this module, depending on your system's design:

Package	Description
hcc_base_doc	This contains the two guides that will help you get started.
fs_fuse_hcc_exfat	The exFAT for Linux FUSE package described in this document.
fs_exfat	The exFAT File System package that this package interfaces to.
fs_exfat_safe	The SafeexFAT extension for the file system
media_drv_*	A media driver that can access a block or character device in a Linux system. An example is **media_drv_blk_linux**.
media_drv_test	This is optional but is needed if you want to run media failure tests. This can initialized by using the PSP function **psp_fuse_initvolume()**.

Documents

For an overview of HCC file systems and guidance on choosing a file system, see Product Information on the main HCC website.

Readers should note the points in the HCC Documentation Guidelines on the HCC documentation website.

HCC Firmware Quick Start Guide

This document describes how to install packages provided by HCC in the target development environment. Also follow the *Quick Start Guide* when HCC provides package updates.

HCC Source Tree Guide

This document describes the HCC source tree. It gives an overview of the system to make clear the logic behind its organization.

HCC exFAT and SafeexFAT File System User Guide

This document describes the main HCC exFAT and SafeexFAT file system packages.

HCC exFAT and SafeexFAT for Linux FUSE User Guide

This is this document.

1.4 Change History

This section describes past changes to this manual.

- To download this manual, see File System PDFs.
- For the history of changes made to the package code itself, see History: fs_fuse_hcc_exfat.

The current version of this manual is 1.20. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.20	2019-08-28	1.02	Added references to SafeexFAT.
1.10	2019-04-02	1.02	Added parameters to **psp_fuse_initvolume()** and **psp_fuse_do_test()**. Added loopback switch at end of *Running exFAT* section.
1.00	2019-02-13	1.01	First version.

2 Source File List

This section lists and describes all the source code files included in the system. These files follow HCC Embedded's standard source tree system, described in the HCC Source Tree Guide. All references to file pathnames refer to locations within this standard source tree, not within the package you initially receive.

Note: Do not modify any of these files.

2.1 FUSE Source Code

These files are in the directory **src/exfat/fuse**. **These files should only be modified by HCC**.

File	Description
fuse2_hcc_exfat.c	Implements FUSE library version 2.x.
fuse3_hcc_exfat.c	Implements FUSE library version 3.x.

2.2 Version File

The file **src/version/ver_fuse_hcc_exfat.h** contains the version number of this module. This version number is checked by all modules that use this module to ensure system consistency over upgrades.

2.3 Platform Support Package (PSP) Files

These files are in **src/psp/target/fuse**. Modify them as required for your hardware.

Note:

- These are PSP implementations for the specific microcontroller and board; you may need to modify these to work with a different microcontroller and/or development board; see PSP Porting for details.
- In the package these files are offset to avoid overwriting an existing implementation. Copy them to the root **hcc** directory for use.

File	Description
psp_fuse.c	FUSE functions code.
psp_fuse.h	FUSE functions header file.

The PSP has its own version file, **version/ver_psp_fuse.h**.

3 About FUSE

FUSE is the standard FUSE (Filesystem in User Space) library. This HCC package has been tested with two versions of FUSE.

3.1 FUSE in Different Linux Distributions

If the target system is a Debian-based Linux distribution like Ubuntu for ARM, you can install FUSE by running a few commands on the embedded device. In Debian-based distributions the stock Linux kernel usually includes FUSE as a loadable module.

If the kernel does not contain the FUSE module, you must configure, re-compile, and install the kernel. If the distribution cross-compiles the target system (Yocto Project does this, for example), configure the build system to add FUSE libraries and compile FUSE's kernel module to a final bootable image. The FUSE kernel module can be compiled into the kernel or as a loadable module as well.

Before using any FUSE application the FUSE kernel module must be loaded (if it was not loaded automatically). The FUSE module is usually automatically loaded on Debian based systems.

3.2 Installing FUSE2 on Debian based Linux distributions

As FUSE2 is officially part of Debian-based distributions, installing development libraries is simple. Just run these commands:

```
# apt-get update
# apt-get install libfuse-dev
```

3.3 Installing FUSE3 on Debian based Linux distributions

FUSE3 is not part of these distributions, so it must be downloaded and compiled manually. Different versions of FUSE have different installation procedures. This example shows how to install FUSE version 3.4.1.

> **Note:** HCC uses the FUSE library to interface to exFAT but this is not HCC software so follow your FUSE instructions.

To compile the FUSE 3.4.1 binary, run the commands shown below:

```
# apt-get update
# apt-get install make gcc binutils meson libfuse-dev pkg-config python3-pip
# pip3 install -U pytest # wget https://github.com/libfuse/libfuse/releases/
download/fuse-3.4.1/fuse-3.4.1.tar.xz
# tar xf fuse-3.4.1.tar.xz
# cd fuse-3.4.1
# # The following directions are documented here: https://github.com/libfuse/
libfuse
# mkdir build
# cd build
# meson ..
# ninja
# python3 -m pytest test/
# ninja install
```

4 Using and Testing exFAT

This section describes how to:

1. Compile exFAT.
2. Run exFAT.
3. Test exFAT.
4. Use standard Linux commands to access the exFAT files.

Note that everything in this section applies equally to SafeexFAT.

4.1 Compiling exFAT

To build exFAT:

1. Take all the source code from the exFAT package.
2. From the FUSE package take either the FUSE2 or FUSE3 source code.
3. Take the block media driver source code.
4. Compile the source code.

The output is a Linux executable named, for example, **myhcc_exfat**.

Makefile

For reference a Makefile is provided, which was used to compile the **fuse_hcc_exfat** application in Ubuntu 18.04.2 LTS running on the armv7l architecture. To use the reference Makefile, ‘gcc’ and ‘make’ must be installed.

To compile the FUSE application, 64-bit file operations are needed, so add these defines to the compiler:

```
__USE_LARGEFILE64
_LARGEFILE_SOURCE
_LARGEFILE64_SOURCE
_FILE_OFFSET_BITS=64
```

In the following examples

- **my_hcc_exfat** is the application name that can mount an exFAT volume using HCC's exFAT implementation, media driver, FUSE library, and FUSE kernel module.
- **/dev/mmcblk1** is the name of the block device to mount.
- **/myvolume<n>** is the mount point.

Start the application using the following switches, which are processed by the FUSE library: `-o allow_other -s`

The switches ‘—format’ and ‘—test’ are processed and executed by the HCC implementation.

4.2 Running exFAT

As in a standard Linux system you must create a mount point for each exFAT volume. For example, you might call these **myvolume1**, **myvolume2**, and **myvolume3**.

Some sample commands are shown below. Here **my_hcc_exfat** is the name of the executable.

To format an SD card with exFAT, use this command:

```
# ./my_hcc_exfat -o allow_other -s --format -i /dev/mmcblk1 /myvolume1
```

To mount an SD card with exFAT, use this command:

```
# ./my_hcc_exfat -o allow_other -s -i /dev/mmcblk1 /myvolume1
```

To mount an SD card with exFAT with debug prints, use this command:

```
# ./my_hcc_exfat -o allow_other -s -d -i /dev/mmcblk1 /myvolume2
```

To run the test suite without mounting the card, use this command:

```
# ./my_hcc_exfat -o allow_other -s --test -i /dev/mmcblk1 /myvolume3
```

After mounting exFAT, you can access files on the volume, for example using /**myvolume1**. Access the exFAT files in exactly the same way as the files on any other Linux volume.

The commands detailed in Using Standard Commands are examples of command usage.

The `-loopback` switch is the equivalent of the `mount -o loop` command. It allows mounting of regular files as shown below for a mount disk image:

```
# ./my_hcc_exfat -o allow_other -s --loopback -i /home/me/my_disk.img /mnt
```

4.3 Testing exFAT

A test suite is provided to test exFAT.

To run HCC's test suite for exFAT, do the following:

1. Run the command **make** in the directory **hcc/util/tests** to compile the test suite.
2. Mount the exFAT media in the **/myvolume** directory.
3. Start the test by using the following command:

   ```
   ~/hcc/util/tests# ./test_hcc_exfat -dir /myvolume
   ```

4.4 Using Standard Commands

This section gives examples using standard Linux commands with exFAT files.

Formatting a drive

Note that all data on the **/dev/mmcblk1** will be lost. The drive will not be mounted to **/myvolume1**. To format a drive, use this command:

```
root@linux:~# ./my_hcc_exfat -o allow_other -s --format -i /dev/mmcblk1 /myvolume1

Formatting drive...

Drive formatted
```

Mounting a drive

To mount a drive, use this command:

```
root@linux:~# ./my_hcc_exfat -o allow_other -s -i /dev/mmcblk1 /myvolume1
```

Listing mounted drives

To list mounted drives, where the exFAT volume is mounted to /myvolume1:

```
root@linux:~# mount

...

my_hcc_exfat on /myvolume1 type fuse.my_hcc_exfat
(rw,nosuid,nodev,relatime,user_id=0,group_id=0,allow_other)
```

Creating a file

To create a file on the media, use this command:

```
root@linux:~# echo Hello world >/myvolume1/hello.txt
```

Listing files

To list files, use this command:

```
root@linux:~# ls -l /myvolume1

total 1
```

-rw-rw-rw- 1 root root 12 Feb 12 13:14 hello.txt

Displaying a new file

To display the newly created file, use this command:

```
root@linux:~# cat /myvolume1/hello.txt

Hello world

root@linux:~# umount /myvolume1
```

5 Reference Makefile

This makefile is provided with the package. It was used to compile the **fuse_hcc_exfat** application in Ubuntu 18.04.2 LTS running on the armv7l architecture.

```
# if 0: use FUSE 2.x
# if 1: use FUSE 3.x
ENABLE_FUSE3 = 1
ENABLE_DEBUG = 1

APP_NAME = fuse_hcc_exfat
ENABLE_CROSS_COMPILE = 0

# Tool chain settings
ifeq ($(ENABLE_CROSS_COMPILE),1)
CROSS = arm-
CPP = $(CROSS)g++
CC  = $(CROSS)gcc
LD  = $(CROSS)g++
OBJCOPY = $(CROSS)objcopy
endif

INCLUDE = -I.

# Options for compiler and linker as well
COMMON_FLAGS =

# Compiler options
CFLAGS = -c $(INCLUDE)
CFLAGS += -Wall -pedantic
CFLAGS += -Wextra -std=c11
# -Wno-format: suppress warnings of printf("%S", ...)
CFLAGS += -Wno-format
ifeq ($(ENABLE_DEBUG),1)
# debug
CFLAGS += -O0 -ggdb3
CFLAGS += -D_DEBUG=1 # PSP_ASSERT enabled
else
# release
CFLAGS += -O2
endif
CFLAGS += -D__USE_LARGEFILE64=1
CFLAGS += -D_LARGEFILE_SOURCE=1
CFLAGS += -D_LARGEFILE64_SOURCE=1
CFLAGS += -D_FILE_OFFSET_BITS=64
ifeq ($(ENABLE_FUSE3),1)
CFLAGS += -DFUSE_USE_VERSION=30
CFLAGS += `pkg-config fuse3 --cflags`
else
CFLAGS += -DFUSE_USE_VERSION=26
endif
CFLAGS += -D_POSIX_C_SOURCE
ifeq ($(ENABLE_DEBUG),1)
CFLAGS += -DFUSE_HCC_EXFAT_TEST_SUITE=1
endif
CFLAGS += $(COMMON_FLAGS)
CPPFLAGS = $(CFLAGS)
CFLAGS_A = $(CFLAGS) -D_ASSEMBLER_ -x assembler-with-cpp

# Linker options
LDFLAGS = $(COMMON_FLAGS)
LDFLAGS += -pthread
```

```
ifeq ($(ENABLE_FUSE3),1)
LDFLAGS += `pkg-config fuse3 --libs`
else
LDFLAGS += -lfuse
endif

# Set source files
SRC_CPP =
SRC_C =
SRC_C += ../hcc/src/config/config_exfat.c
SRC_C += ../hcc/src/exfat/common/exfat.c
SRC_C += ../hcc/src/exfat/common/exfat_bitmap.c
SRC_C += ../hcc/src/exfat/common/exfat_cache.c
SRC_C += ../hcc/src/exfat/common/exfat_convert.c
SRC_C += ../hcc/src/exfat/common/exfat_dir.c
SRC_C += ../hcc/src/exfat/common/exfat_direntry.c
SRC_C += ../hcc/src/exfat/common/exfat_driver.c
SRC_C += ../hcc/src/exfat/common/exfat_driver_low.c
SRC_C += ../hcc/src/exfat/common/exfat_fat.c
SRC_C += ../hcc/src/exfat/common/exfat_file.c
SRC_C += ../hcc/src/exfat/common/exfat_upcase_table.c
SRC_C += ../hcc/src/exfat/test/exfat_test.c
SRC_C += ../hcc/src/exfat/test/exfat_test_dir.c
SRC_C += ../hcc/src/exfat/test/exfat_test_file.c
SRC_C += ../hcc/src/exfat/test/exfat_test_media.c
SRC_C += ../hcc/src/exfat/test/exfat_test_task.c
SRC_C += ../hcc/src/exfat/test/exfat_test_upcase.c
SRC_C += ../hcc/src/media-drv/blk_linux/mdriver_blk_linux.c
SRC_C += ../hcc/src/media-drv/file/drv_file.c
SRC_C += ../hcc/src/media-drv/ram/ramdrv_f.c
SRC_C += ../hcc/src/media-drv/test/mdriver_test.c
SRC_C += ../hcc/src/oal/os/oalp_mutex.c
SRC_C += ../hcc/src/oal/os/oalp_task.c
ifeq ($(ENABLE_FUSE3),1)
SRC_C += ../hcc/src/exfat/fuse/fuse3_hcc_exfat.c
else
SRC_C += ../hcc/src/exfat/fuse/fuse2_hcc_exfat.c
endif
SRC_C += ../hcc/src/psp/board/demo/my_malloc.c
SRC_C += ../hcc/src/psp/common/psp_reg.c
SRC_C += ../hcc/src/psp/common/psp_string.c
SRC_C += ../hcc/src/psp/target/assert/psp_assert.c
SRC_C += ../hcc/src/psp/target/exfat/psp_exfat_test.c
SRC_C += ../hcc/src/psp/target/fuse/psp_fuse.c
SRC_C += ../hcc/src/psp/target/rand/psp_rand.c
SRC_C += ../hcc/src/psp/target/rtc/psp_rtc.c
OBJ_CPP = $(patsubst %.cpp, %.o, $(SRC_CPP))
OBJ_C   = $(patsubst %.c, %.o, $(SRC_C))
OBJ_S   = $(patsubst %.s, %.o, $(SRC_S))
OBJ     = $(OBJ_CPP) $(OBJ_C) $(OBJ_S)
DEP     = $(patsubst %.o, %.d, $(OBJ))

# Compile rules

.PHONY : all

all : $(APP_NAME)
```

```
$(APP_NAME) : $(OBJ)
        $(CC) $(OBJ) -o $@ $(LDFLAGS)

$(APP_NAME).bin: $(APP_NAME)
        $(OBJCOPY) -O binary $< $@

$(OBJ_CPP) : %.o : %.cpp
        $(CPP) $(CPPFLAGS) -o $@ $<
        @$(CPP) -MM $(CPPFLAGS) -MT $@ $*.cpp > $*.d

$(OBJ_C) : %.o : %.c
        $(CC) $(CFLAGS) -o $@ $<
        @$(CC) -MM $(CFLAGS) -MT $@ $*.c > $*.d

$(OBJ_S) : %.o : %.s
        $(CC) $(CFLAGS_A) -o $@ $<
        @$(CC) -MM $(CFLAGS_A) -MT $@ $*.s > $*.d

# Include dependencies
-include $(DEP)

# Clean rules

.PHONY : clean

clean :
        rm -f $(OBJ) $(DEP) $(APP_NAME)

.PHONY: tags
tags:
        ctags -R .
```

6 PSP Porting

The Platform Support Package (PSP) is designed to hold all platform-specific functionality, either because it relies on specific features of a target system, or because this provides the most efficient or flexible solution for the developer.

The module makes use of the following PSP functions, described in the following sections:

Function	Description
psp_fuse_initvolume()	Initializes a FUSE volume. Use this to initialize the media driver and also the **media_drv_test** suite, if this is used.
psp_fuse_delvolume()	Deletes a FUSE volume.
psp_fuse_do_test()	Runs the test suite on a volume.

6.1 psp_fuse_initvolume

The PSP provides this function to initialize a FUSE volume.

Use this to initialize the media driver and also the **media_drv_test** suite, if you want to use this.

Format

```
t_exfat_ret psp_fuse_initvolume (
    t_exfat_drive   drivenum,
    const char *    p_device_name,
    uint8_t         b_is_loopback,
    uint32_t        sector_size )
```

Arguments

Argument	Description	Type
drivenum	The index of the volume to initialize.	t_exfat_drive
p_device_name	A pointer to the Linux device to open, for example: **/dev/mmcblk1**.	char *
b_is_loopback	TRUE: allowed to open regular (disk image) files. FALSE: allowed to open block device files.	uint8_t
sector_size	Sector size in bytes (only applies if *b_is_loopback* is TRUE).	uint32_t

Return values

Return value	Description
EXFAT_SUCCESS	Successful execution.
EXFAT_ERR_MEDIA_DRIVER	Operation failed.

6.2 psp_fuse_delvolume

The PSP provides this function to delete a FUSE volume.

Format

```
t_exfat_ret psp_fuse_delvolume ( t_exfat_drive drivenum )
```

Arguments

Argument	Description	Type
drivenum	The index of the volume to delete.	t_exfat_drive

Return values

Return value	Description
EXFAT_SUCCESS	Successful execution.
EXFAT_ERR_MEDIA_DRIVER	Operation failed.

6.3 psp_fuse_do_test

The PSP provides this function to run the test suite on a FUSE volume.

Format

```
t_exfat_ret psp_fuse_do_test (
    t_exfat_drive   drivenum,
    const char *    p_device_name,
    uint8_t         b_is_loopback,
    uint32_t        sector_size )
```

Arguments

Argument	Description	Type
drivenum	The index of the volume to test.	t_exfat_drive
p_device_name	A pointer to the Linux device to open. Example: **/dev/mmcblk1**.	char *
b_is_loopback	TRUE: allowed to open regular (disk image) files. FALSE: allowed to open block device files.	uint8_t
sector_size	Sector size in bytes (only applies if *b_is_loopback* is TRUE).	uint32_t

Return values

Return value	Description
EXFAT_SUCCESS	Successful execution.
EXFAT_ERR_MEDIA_DRIVER	Operation failed.