IEC/TR 62291

Edition 2.0 2009-02

TECHNICAL REPORT

Multimedia data storage – Application program interface for UDF based file systems

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub

The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub

Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Electropedia: www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv

If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC/TR 62291

Edition 2.0 2009-02

TECHNICAL REPORT

Multimedia data storage – Application program interface for UDF based file systems

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

PRICE CODE Q

ICS 33.160.40; 35.220

ISBN 2-8318-1031-0

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

MULTIMEDIA DATA STORAGE – APPLICATION PROGRAM INTERFACE FOR UDF BASED FILE SYSTEMS

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

The main task of IEC technical committees is to prepare International Standards. However, a technical committee may propose the publication of a technical report when it has collected data of a different kind from that which is normally published as an International Standard, for example "state of the art".

IEC 62291, which is a technical report has been prepared by IEC technical committee 100: Audio, video and multimedia systems and equipment.

This second edition cancels and replaces the first edition, published in 2002, and constitutes a technical revision.

The significant changes with respect to the first edition are the following:

- reference document ISO/IEC 9945-1:1990 is replaced with ISO/IEC 9945-2:2003.
- reference document UDF 2.00 is replaced with UDF 2.01.

The text of this technical report is based on the following documents:

Enquiry draft	Report on voting
100/1452/CDV	100/1499/RVC

Full information on the voting for the approval of this technical report can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

The committee has decided that the contents of this publication will remain unchanged until the maintenance result date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

A bilingual version of this publication may be issued at a later date.

INTRODUCTION

Interchangeable storage media have been widely employed for information interchange; the following extensions to their media format should therefore be standardized:

a) additional facilities including security (access control, originality management, etc.);

b) volume and file structure supporting the facilities;

c) API (Application Program Interfaces) to the volume and file structure.

For a number of disc media, ISO/IEC JTC1/SC15 developed a generic standard of volume and file structure and it was actually used for rewritable, recordable and read-only DVD and recordable CD file systems with some subsetting by OSTA (Optical Storage Technology Association). The subsetting specification is called a UDF (Universal Disk Format).

Additional facilities and an API for the UDF have been discussed in OITDA (Optoelectronic Industry and Technology Development Association). OITDA drafted their specifications and submitted them to JISC (Japanese Industrial Standard Committee). METI (Ministry of Economy, Trade and Industry, Japan) approved them and JSA (Japanese Standards Association) published them in July 2001 as

a) JIS/TR X 0040:2001 Security Extension to Universal Disk Format (UDF);

b) JIS/TR X 0041:2001 Application Program Interface for UDF based File Systems.

IEC/TC100 National Committee of Japan then submitted the English version of JIS/TR X 0041:2001 to IEC/TC100.

MULTIMEDIA DATA STORAGE – APPLICATION PROGRAM INTERFACE FOR UDF BASED FILE SYSTEMS

1 Scope

This Technical Report (TR) specifies an application program interface (API) for reading and writing the files which conform to the Universal Disk Format (UDF) ®[1] specification revision 2.00 developed by the Optical Storage Technology Association (OSTA) and its security extension.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 9945-2:2003, *Information technology – Portable Operating System Interface (POSIX) – Part 2: System Interfaces*

Universal Disk Format (UDF) Specification Revision 2.01, Optical Storage Technology Association (OSTA), 2000

Secure UDF, Revision 1.00, OSTA, 2002

JIS/TR X 0040:2001, *Security Extension to Universal Disk Format (UDF)*

NOTE JIS/TR X 0040 was translated into English under the title *Secure UDF Specification* (2002), see reference above.

3 Terms and definitions

For the purposes of this technical report, the following definitions apply.

3.1
access logging
security feature to record and refer to operations for each file; an access logging contains information on when the operation is applied, who applies the operation, and what type of operation is applied

3.2
security function
function that ensures that an implementation applies to each file when the implementation accesses to the files

NOTE A security function contains access logging, etc.

---

[1] Universal Disk Format (UDF) is a registered trademark developed by the Optical Storage Technology Association. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

3.3
usage environment
environment in which use of contents is allowed

NOTE An example of the environment is a combination of specific storage medium, specific storage device and specific decoder.

3.4
usage environment identifier
identifier for distinguishing a specific usage environment with other usage environments

NOTE A storage medium identifier can be used as a usage environment identifier.

3.5
license
encrypted decryption key for encrypted contents, i.e., a default stream and user streams

3.6
secure UDF
Universal Disk Format (UDF) that contains the structures specified by JIS/TR X 0040 (or OSTA Secure UDF, Revision 1.00)

4 Notation

The conventions for symbol constants that are returned by functions as error numbers conform to the description on “error numbers” in 2.3 of ISO/IEC 9945-2:2003.

5 File operations conforming to ISO/IEC 9945-2

The file operations in Table 1 conform to the system interfaces specified by ISO/IEC 9945-2:2003.

Table 1 – File operations conforming to ISO/IEC 9945-2

ISO/IEC 9945-2 Clause	Operation	Function name
3	Directory Operations	opendir() readdir() rewinddir() closedir()
3	Change Current Working Directory	chdir()
3	Working Directory Pathname	getcwd()
3	Open a File	open()
3	Create a New file or Rewrite an Existing One	creat()
3	Set File Creation Mask	umask()
3	Link to a File	link()
3	Make a Directory	mkdir()
3	Make a FIFO Special File	mkfifo()
3	Remove Directory entries	unlink()
3	Remove a Directory	rmdir()
3	Rename a File	rename()
3	Get File Status	stat() fstat()
3	File Accessibility	access()
3	Change File Modes	chmod()
3	Change Owner and Group of a File	chown()
3	Sets File Access and Modification Times	utime()
3	Close a File	close()
3	Read from a File	read()
3	Write to a File	write()
3	File Control	fcntl()
3	Reposition Read/Write File Offset	lseek()

6 UDF specific file operations

This clause specifies UDF specific file operations.

6.1 Header and data structure

6.1.1 General

The udf.h header file includes the structure definitions that are dealt with in UDF file systems. For information about each structure definition, see the UDF Specification revision 2.01.

6.1.2 File entry structures in udf.h

The udf.h header file defines the udfent UDF file entry structures that are returned by the udfgetent() function. Tables 2 to 5 list the definitions of related structures.

Table 2 – udfent structure

Member type	Member name	Description
Udftag	desc_tag	Tag ID of descriptors (261 for basic files and 266 for extension files)
Uint32	Uid	User ID of file owner
Uint32	Gid	Group ID of file owner
Uint32	Perm	Access permission
Uint16	file_link_cnt	Number of file links
Uint8	Rec_fmt	Record format
Uint8	rec_disp_attr	Record display attribute
Uint32	Rec_len	Record length (bytes)
Uint64	info_len	File length (bytes)
Uint64	Obj_size	File length including all streams (bytes)
Uint64	lblk_rec	Number of logical blocks recorded
timestamp	Acstime	Time of last file access
timestamp	Modtime	Time of last file modification
timestamp	Crttime	Time of file creation
timestamp	Atrtime	Time of last file attribute modification
Uint32	checkpoint	Checkpoint
regid	impl_id	Implementation ID
Uint64	uniq_id	Unique ID

Table 3 – udftag structure

Member type	Member name	Description
Uint16	Tag_id	Tag ID
Uint16	desc_ver	Descriptor version
Uint8	Tag_chksum	Checksum of the udftag structure
Uint8	Pad1	Reserved (#00)
Uint16	Tag_serno	Descriptor group ID
Uint16	desc_crc	CRC of the descriptor
Uint16	desc_crc_len	Length of CRC
Uint16	Tag_loc	Location of the descriptor (logical block number)

Table 4 – timestamp structure

Member type	Member name	Description
Uint16	type_timezone	Parameters in this member have the following meanings: Type = 0: UTC, Type = 1: local, and Type = 2: needs agreement between sender and receiver
Uint16	Year	Year (1 to 9999)
Uint8	Month	Month (1 to 12)
Uint8	Day	Day (1 to 31)
Uint8	Hour	Hour (0 to 23)
Uint8	Min	Minute (0 to 59)
Uint8	Sec	Second (type 2: 0 to 60, others: 0 to 59)
Uint8	Centisec	Centisecond (1 hundredth of a second) (0 to 99)
Uint8	microsec100	100 microseconds (0 to 99)
Uint8	Microsec	Microsecond (0 to 99)

Table 5 – regid structure

Member type	Member name	Description
Uint8	flags	Attribute
Uint8[23]	id	Implementation identifier (ID)
Uint8[8]	id_suffix	Suffix information

6.1.3 Extended attribute structure in udf.h

The udf.h header file defines the udfxattr extended attribute structure that is returned by the udfgetxattr() function. Table 6 lists the definitions of the udfxattr structure members.

Table 6 – udfxattr structure

Member type	Member name	Description
Uint32	attr_type	Extension attribute type
Uint8	attr_subtype	Extension attribute subtype
Uint8	pad1	Reserved (#00)
Uint32	attr_len	Total length of extended attributes
Uint8[]	attr_data	Data part of extended attributes

6.1.4 Date and time structure in udf.h

The udf.h header file defines the udftime date and time structure specified by the udfsettime() function. Table 7 lists the definitions of the udftime structure members.

Table 7 – udftime structure

Member type	Member name	Description
timestamp	acstime	Date and time of last file access
timestamp	modtime	Date and time of last file modification
timestamp	crttime	Date and time of file creation
timestamp	atrtime	Date and time of last file attribute modification

6.1.5 Permission structure in udf.h

The permission values that are specified by the udfsetperm() function are encoded with the following masks and bits.

S_IXWRADO: Mask permitting others to execute files, write files, read files, change file attributes, and delete files
- S_IXOTH: bit permitting others to execute files: bit 0
- S_IWOTH: bit permitting others to write files: bit 1
- S_IROTH: bit permitting others to read files: bit 2
- S_IAOTH: bit to permitting others to change file attributes: bit 3
- S_IDOTH: bit to permitting others to delete files: bit 4

S_IXWRADG: Mask permitting groups to execute files, write files, read files, change file attributes, and delete files
- S_IXGRP: bit permitting groups to execute files: bit 5
- S_IWGRP: bit permitting groups to write files: bit 6
- S_IRGRP: bits permitting groups to read files: bit 7
- S_IAGRP: bits permitting groups to change file attributes: bit 8
- S_IDGRP: bits permitting groups to delete files: bit 9

S_IXWRADU: Mask permitting owners to execute files, write files, read files, change file attributes, and delete files
- S_IXUSR: bit permitting owners to execute files: bit 10
- S_IWUSR: bit permitting owners to write files: bit 11
- S_IRUSR: bit permitting owners to read files: bit 12
- S_IAUSR: bit permitting owners to change file attributes: bit 13
- S_IDUSR: bit permitting owners to delete files: bit 14

6.2 Get UDF file attribute information

6.2.1 Get a UDF file entry

Function: udfgetent()

6.2.1.1 Synopsis

```
#include <udf.h>

int udfgetent(int fildes, struct udfent *buf);
```

6.2.1.2 Description

The udfgetent() function obtains the entry information about the file specified in the file descriptor argument and writes the entry information to the area specified in the buf argument.

The UDF file entry information is classified into file entry type and extended file entry type. When this function is executed, if the value of tag_id in the desc_tag structure is 261, the UDF file entry information is specified as the file entry type. If the value of tag_id is 266, the UDF file entry information is specified as the extended file entry type. The value of obj_size and crttime can be specified only for the extended file entry type.

6.2.1.3 Values returned by the function

If the function is executed successfully, 0 is returned. Otherwise, −1 is returned and an error code is set in errno.

6.2.1.4 Errors

If the following error occurs, the udfgetent() function returns −1 and an error code, listed below, is set in errno:

[EBADF] The file specified in the file descriptor argument is invalid.

6.2.2 Get UDF extended attribute

Function: udfgetxattr()

6.2.2.1 Synopsis

#include <udf.h>

int udfgetxattr(int fildes, Uint32 atype, struct udfxattr *buf);

6.2.2.2 Description

The udfgetxattr() function obtains extended attribute information about the file specified in the file descriptor argument. Depending on the settings in the atype and buf arguments, only the extended attribute information that matches the attribute type specified in the atype argument is obtained. This function writes the obtained information to the area specified in the buf argument.

If atype is 0, all extended attribute information is obtained. If buf is 0, the function returns the size (in bytes) of the extended attribute information that matches the attribute type specified in the atype argument. Therefore, this function can be used if the size of the necessary areas for storing the extended attribute information is not known in advance. If atype and buf are both 0, the total size of all extended attribute information is returned.

6.2.2.3 Values returned by the function

If buf is not 0, 0 is returned when the function is executed successfully. Otherwise, −1 is returned and an error code is set in errno.

If buf is 0, the size (in bytes) of the extended attribute area that matches the specified attribute type is returned. If the extended attribute information having the specified attribute type is not found, −1 is returned.

6.2.2.4 Errors

If any of the following errors occur, the udfgetxattr() function returns −1 and an error code, listed below, is set in errno:

[EBADF] The file specified in the file descriptor argument is invalid.

[EINVAL] The attribute type specified in atype is invalid.

6.3 Set UDF file attribute information

6.3.1 Set a access permission

Function: udfsetperm()

6.3.1.1 Synopsis

#include <udf.h>

int udfsetperm(int fildes, Uint32 perm);

6.3.1.2 Description

The udfsetperm() function replaces the permission that is specified in a file entry or extended file entry for the file specified in the file descriptor argument with the permission specified in the perm argument (see 6.1.5.)

6.3.1.3 Values returned by the function

If the function is executed successfully, 0 is returned. Otherwise, −1 is returned and an error code is set in errno.

6.3.1.4 Errors

If any of the following errors occur, the udfsetperm() function returns −1 and an error code, listed below, is set in errno:

[EBADF] The file specified in the file descriptor argument is invalid.

[EPERM] The process that called the function does not have valid authority.

6.3.2 Set a date and time

Function: udfsettime()

6.3.2.1 Synopsis

#include <udf.h>

int udfsettime(int fildes, const struct udftime *timebuf);

6.3.2.2 Description

The udfsettime() function replaces the date and time that is specified in the file entry or extended file entry for the file specified in the file descriptor argument with the date and time specified in the timebuf argument. Only file owners and the processes having valid permission are entitled to use this function.

6.3.2.3 Values returned by the function

If the function is executed successfully, 0 is returned. Otherwise, −1 is returned and an error code is set in errno.

6.3.2.4 Errors

If any of the following errors occur, the udfsettime() function returns −1 and an error code, listed below, is set in errno:

[EBADF] The file specified in the file descriptor argument is invalid.

[EPERM] The process that called the function does not have valid authority.

7 Security extension

7.1 General

The functions in this clause execute the operations specified by JIS/TR X 0040 or OSTA Secure UDF, Revision 1.00.

7.2 Header and data structure

7.2.1 Header file

The udfse.h header file includes the structure definitions that are handled by the security extension function.

7.2.2 Access log descriptor structure in the udfse.h header file

The udfse.h header file defines the aldesc access log descriptor structures that are returned by the segetlog() function. Tables 8 and 9 list the definitions of the related structures.

Table 8 – aldesc structure

Member type	Member name	Description
timestamp	optime	Date and time of operation
envspec	envspec	Identifier stating where the operation was taken
Uint32	op	Operation
Uint32	uidtype	Operation
Uint32	uidlen	Length of user identifier
Uint8[]	uid	User identifier or index of user identifier

Table 9 – envspec structure

Member type	Member name	Description
Uint8[64]	systemid	System identifier
Uint8[64]	deviceid	Storage device identifier
Uint8[192]	mediaid	Storage media identifier
Uint8[16]	lsn	Logical sector number of storage media
Uint8[48]	padding	Reserved for future standardization

7.3 Log operation

7.3.1 General

The function described in this subclause handles the operation of type 1 access log streams. (See JIS/TR X 0040 or OSTA Secure UDF, Revision 1.00.)

7.3.2 Get a log

Function: segetlog()

7.3.2.1 Synopsis

```
#include <udfse.h>

int segetlog(int fildes, int logno, int loglen, struct aldesc *logbuf);
```

7.3.2.2 Description

The segetlog() function reads log data, stores it in the aldesc structure (see 7.2.2), and returns a value. The logno argument is used to specify the number of the log to be read, the logbuf argument to specify the buffer for reading the log data, and the loglen argument to specify the length of the aldesc structure.

7.3.2.3 Returns

If the function is executed successfully, 0 is returned. Otherwise, −1 is returned and an error code is set in errno. If the function is not executed successfully, the content of logbuf might be incorrect.

7.3.2.4 Errors

If the following error occurs, the segetlog() function returns −1 and an error code, listed below, is set in errno:

[EINVAL] The size specified in the loglen argument is smaller than the size of the aldesc structure.

7.4 Licensing operation

The functions described in this clause handle the operation of type 1 license streams. (See JIS/TR X 0040 or OSTA Secure UDF, Revision 1.00.)

7.4.1 Get a license list

Function: segetlicenselist**()**

7.4.1.1 Synopsis

int segetlicenselist(int fildes, int bufnum, int buf[], int *num);

7.4.1.2 Description

The segetlicenselist() function reads a list of licensed operations. The file descriptor argument is used to specify the file path name for each content item, the bufnum argument to specify the length of the buffer for reading a list of licensed operations, the buf argument to specify the buffer for reading a list of licensed operations, and the num argument to specify the number of operations that have already been registered.

7.4.1.3 Returns

If the function is executed successfully, 0 is returned. Otherwise, −1 is returned and an error code is set in errno. If the function is not executed successfully, the content of buf might be incorrect.

7.4.1.4 Errors

If the following error occurs, the segetlicenselist() function returns −1 and an error code, listed below, is set in errno:

[EBADF] The file specified in the file descriptor argument is invalid.

7.4.2 Add a license

Function: seaddlicense()

7.4.2.1 Synopsis

int seaddlicense(int fildes, int operation, int buflen, const char *bufp);

7.4.2.2 Description

The seaddlicense() function adds licenses. The file descriptor argument is used to specify the file path name for each content item, the operation argument to specify operation numbers (1 for play and 2 reserved for future use), the buflen argument to specify the length of the buffer for adding licenses, and the bufp argument to specify the buffer for adding licenses.

7.4.2.3 Returns

If the function is executed successfully, 0 is returned. Otherwise, −1 is returned and an error code is set in errno. If the function is not executed successfully, the content of buf might be incorrect.

7.4.2.4 Errors

If any of the following errors occur, the seaddlicense() function returns −1 and an error code, listed below, is set in errno:

[EBADF] The file specified in the file descriptor argument is invalid.

[EINVAL] The operation number specified in the operation argument is invalid.

7.4.3 Delete a license

Function: sedellicense()

7.4.3.1 Synopsis

int sedellicense(int fildes, int operation);

7.4.3.2 Description

The sedellicense() function deletes licenses. The file descriptor argument is used to specify the file path name for each content item, and the operation argument to specify operation numbers (1 for play and 2 reserved for future use).

7.4.3.3 Returns

If the function is executed successfully, 0 is returned. Otherwise, −1 is returned and an error code is set in errno. If the function is not executed successfully, the content of buf might be incorrect.

7.4.3.4 Errors

If any of the following errors occur, the sedellicense() function returns −1 and an error code, listed below, is set in errno:

[EBADF] The file specified in the file descriptor argument is invalid.

[EINVAL] The operation number specified in the operation argument is invalid.

7.4.4 Set a license

Function: sesetlicense()

7.4.4.1 Synopsis

int sesetlicense(int fildes, int operation);

7.4.4.2 Description

The sesetlicense() function checks the requirements of all relevant devices for licensing. If the devices pass all checks, an encryption key for decoding encrypted content is loaded into the device that most recently passed the check. This function is used for each operation for each content. The file descriptor argument is used to specify the file path name for each content item, and the operation argument to specify operation numbers (1 for play and 2 reserved for future use).

7.4.4.3 Returns

If the function is executed successfully, 0 is returned. Otherwise, −1 is returned and an error code is set in errno. If the function is not executed successfully, the content of buf might be incorrect.

7.4.4.4 Errors

If any of the following errors occur, the sesetlicense() function returns −1 and an error code, listed below, is set in errno:

[EBADF] The file specified in the file descriptor argument is invalid.

[EINVAL] The operation number specified in the operation argument is invalid.

———————

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
PO Box 131
CH-1211 Geneva 20
Switzerland

Tel: + 41 22 919 02 11
Fax: + 41 22 919 03 00
info@iec.ch
www.iec.ch