ecma INTERNATIONAL

Technical Report

ECMA TR/112-8

1st Edition / December 2023

Universal Disk Format (UDF) specification – Part 8 (Secure UDF)

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

Table of Contents

1 Introduction

The Secure UDF specification defines a set of security enhancements to the Universal Disk Format (UDF) specification. The primary goal of Secure UDF is to provide support for encryption based security features that are transparent to the user and their applications and is portable between different operating system platforms. Secure UDF is designed to:

- Provide an encryption scheme that should work with any application that is storing information on a Secure UDF volume
- Provide a common encryption scheme that all Secure UDF implementations can support.
- Provide a non-proprietary publicly documented method for supporting encryption in Secure UDF.
- Provide a mechanism that allows Secure UDF to take advantage of all the features of Security Enhanced drives.

The following describe the primary reasons that security is needed in UDF:

- *Native operating system security* - Native operating system security is not portable. For example, a UDF volume created under Windows NT with specific NT security rights on specific directories looses all protection if taken to a UNIX platform, which does not support the NT security rights, resulting in everything on the UDF volume being accessible.

- *Removable Media* - Another very important reason is that UDF is used on *removable* media, which can easily be lost or stolen. Removable media greatly increases the need to have some form of portable protection for the information stored on UDF media.

To accomplish this task this document defines a new *Domain*. A domain defines rules and restrictions on the use of ECMA 167. The domain defined in this specification is known as the "OSTA Secure UDF" domain.

The long-term plan for Secure UDF is to integrate it into a future version of the UDF specification as an optional feature. Until that time Secure UDF shall be a separate Domain.

To develop a Secure UDF implementation, a developer will need to reference the following documents as a minimum:

- Secure UDF Specification (this document)
- Universal Disk Format Specification 2.01
- ECMA 167 3rd edition

1.1 History

The initial draft document that was submitted to OSTA as the first draft of Secure UDF originated from the Optoelectronic Industry and Technology Development Association (OITDA) of Japan. OITDA submitted the draft based on the discussions of the Japanese Optical Industry.

1.2 References

ISO/IEC 13346:1999	Volume and file structure of write-once and rewritable media using non-sequential recording for information interchange
ECMA 167 3rd Edition	Volume and file structure of write-once and rewritable media using non-sequential recording for information interchange
UDF	Universal Disk Format Revision 2.01
ISO/IEC 9945-1:1990	Portable Operating System Interface (POSIX) -- Part 1: System Application Program Interface (API) [C Language]
ISO 9160:1988	Information processing - Data encipherment - Physical layer interoperability requirements
ISO 8372:1987 (2nd. confirmation 1997)	Information processing - Modes of operation for a 64-bit block cipher algorithm
ISO/IEC 9797	Information technology - Security techniques - Data integrity mechanism using a cryptographic check function employing a block cipher algorithm
JIS/TR X 0040:2001	Security Extensions to Universal Disk Format (UDF)

1.3 Definitions

Secure UDF	UDF that contains structures specified in this document.
Export	Concatenating a default stream and all named streams, forwarding it to an application one block at a time.
Import	Receiving one block at a time from an application, and expanding it to a default stream and named streams.
Contents	Movie, image, audio data and so on. It is stored in default stream or user

	streams.
License	Encrypted decryption key for encrypted contents (a default stream and user streams).
DES	Data Encryption Standard. See ISO 9160:1988.
CBC	Cipher Block Chaining. See ISO 8372:1987.
MAC	Message Authentication Code. See ISO/IEC 9797 and ISO/IEC 9798-1-4.
X.509	See ISO/IEC 9594-8:1995.

1.4 Document Terminology

May Indicates an action or feature that is optional.

Optional Describes a feature that may or may not be implemented. If implemented, the feature shall be implemented as described.

Shall Indicates an action or feature that is mandatory and must be implemented to claim compliance to this standard.

Should Indicates an action or feature that is optional, but its implementation is strongly recommended.

Reserved A reserved field is reserved for future use and shall be set to zero. A reserved value is reserved for future use and shall not be used.

2 Basic Restrictions and Requirements

The following table summarizes several of the basic restrictions and requirements defined in this specification, which are in addition to the ones described in the UDF specification. These restrictions & requirements as well as additional ones are described in detail in the following sections of this specification.

Item	Restrictions & Requirements
Secure Partition	The maximum number of Secure Partitions shall be one. A Secure Partition shall not be created on media with a virtual or sparable partition.

3 Volume Data Structures

This section describes Secure UDF changes and additions to the UDF Specification at the Volume space level. These changes include:

- An update to the *Entity Identifier* suffix to allow easy detection of the presence of Secure UDF structures on a volume. This feature will be critical once Secure UDF is merged into the main UDF specification at some time in the future.
- Creation of a *Secure Partition Map* to describe the secure area on a piece of media.
- Creation of a *UDF Secure Partition* descriptor that describes the location and security characteristics of a Secure Partition.

3.1 Domain Identifier

Located within the Logical Volume Descriptor (UDF 2.2.4) is the *Domain Identifier* field which is a EntityID structure. Secure UDF shall define the fields of the *Domain Identifier* as follows:

```
struct EntityID {          /*ECMA 167 1/7.4 */
        Uint8              Flags;
        char               Identifier[23];
        char               IdentifierSuffix[8];
}
```

3.1.1 Uint8 Flags

See 2.1.5.1 of UDF.

3.1.2 char Identifier

This field shall indicate that the contents of this logical volume conforms to the Secure UDF domain defined in this document, therefore the *Identifier* field of the *Domain Identifer* shall be set to:

"***OSTA Secure UDF**"

This field shall contain "***OSTA Secure UDF**" which indicates that the volume is in Secure UDF format.

3.1.3 char Identifier Suffix

The *Identifier Suffix* field of the Domain Identifier field contained within the Logical Volume Descriptor shall be defined as follows:

Domain Identifier Suffix field format for Secure UDF

RBP	Length	Name	Contents
0	2	UDF Revision	Uint16 (=#0201)
2	1	Domain Flags	Uint8
3	2	Secure UDF Revision Level	Uint16 (=#0100)
4	3	Reserved	bytes (=#00)

Secure UDF Revision Level shall specify revision of this document for which the Logical Volume is compatible. It shall be set to 1 to indicate this document.

Domain Flags

Bit	Interpretation
0	Hard Write-Protect.
1	Soft Write-Protect
2	Secure UDF
3-7	Reserved

A Secure UDF flag value of ONE shall indicate that a logical volume contains Secure UDF structures.

NOTE: At some future time when Secure UDF is integrated into a new version of UDF, this flag will allow an implementation to detect if Secure UDF descriptors are present on a logical volume.

3.2 Secure Partition Map

Secure UDF creates the concept of a Secure Partition, were information is protected through the methods described in this document. There may be additional protection methods used within the Secure Partition by a Security Enhanced Drive.

The SecurePartitionMap identifies a Secure Partition Descriptor..

Layout of Type 2 partition map for secure partition

RBP	Length	Name	Contents
0	1	Partition Map Type	Uint8 = 2
1	1	Partition Map Length	Uint8 = 64
2	2	Reserved	#00 bytes
4	32	Partition Identifier	EntityID
36	2	Volume Sequence Number	Uint16
38	2	Partition Number	Uint16
40	24	Reserved	#00 bytes

For Secure UDF, the Partition Identifier field shall specify:

"***UDF Secure Partition**".

3.3 Partition Descriptor

The Partition Descriptor referenced by the Secure Partition Map shall be defined as follows:

```
struct PartitionDescriptor {          /* See ECMA 167 3/17 */
        struct tag              DescriptorTag;
        Uint32                  VolumeDescriptorSequenceNumber;
        Uint16                  PartitionFlags;
        Uint16                  PartitionNumber;
        struct EntityID         PartitionContents;
        byte                    PartitionContentsUse[128];
        Uint32                  AccessType;
        Uint32                  PartitonStartingLocation;
        Uint32                  PartitionLength;
        struct EntityID         ImplementationIdentifier;
        byte                    ImplementationUse[128];
        byte                    Reserved[156];
}
```

Unless otherwise specified below the fields of the Partition Descriptor shall be defined according to ECMA 167 3/16.10.5.

3.3.1 struct EntityID ImplementationIdentifier

This field shall specify "***UDF Secure Partition**" as ID value and UDF Identifier Suffix as suffix type.

3.3.2 byte ImplementationUse[128]

This field shall contain a struct encspec, which defines the encryption information associated with the Secure Partition being defined.. Rest of this field shall contain all #00.

3.3.2.1 struct encspec Encryption

Type 1 struct encspec Encryption format

RBP	Length	Name	Contents
0	2	**encspec Type**	Uint16
2	2	**enspec Length**	Uint16
4	4	Encryption Algorithm Type	Uint32
8	4	Encryption Algorithm Sub Type	Uint32
12	4	Encryption Key Type	Uint32
16	4	Encryption Key Sub Type	Uint32
20	4	Type of User ID	Uint32
24	*1	Encryption Key	bytes
24+*1	*2	Padding	bytes

3.3.2.1.1 Uint16 encspec Type

enspec Type

Type	Interpretation
0	Shall mean the type of encspec is not defined by this field.
1	Shall mean that the encspec is a Type 1 encspec.
2-63	Reserved
64-	Shall be subject to agreement between the originator and recipient of the medium.

3.3.2.1.2 Uint16 encspec Length

This field shall specify the length in bytes, of this encspec, including encspec Type and encspec Length fields.

3.3.2.1.3 Uint32 Encryption Algorithm Type

Encryption Algorithm Type

Type	Interpretation
0	Shall mean that the algorithm type is not specified by this field.
1	Shall mean that no algorithm is specified.
2	Shall mean that single DES-CBC is specified.
3	Shall mean that triple DES-CBC is specified.
4-	Reserved

3.3.2.1.4 Uint32 Encryption Algorithm Sub Type

This field shall specify sub type of encryption algorithm. It shall be specified for each Encryption Algorithm Type.

3.3.2.1.5 Uint32 Encryption Key Type

Encryption Key Type

Type	Interpretation
0	Shall mean that the encryption key type is not specified by this field.
1	Shall mean that the encryption key type is storage media specific key.
2	Shall mean that the encryption key type is storage drive specific key.
3	Shall mean that the encryption key type is system specific key.
4	Shall mean that the encryption key type is user specific key.
5	Shall mean that the encryption key is stored in Key field.
6-	Reserved

3.3.2.1.6 Uint32 Encryption Key Sub Type

This field shall specify sub type of encryption key. It shall be specified for each Encryption Key Type.

3.3.2.1.7 Uint32 Type of User ID

This field shall specify type of user ID (5.5.10.5) if Encryption Key Type contains type of user specific key.

3.3.2.1.8 bytes Encryption Key

This field shall contain encryption key when the Encryption Key Type field contains value 5.

3.3.2.1.9 bytes Padding

This field shall be ((encspec Length)-(24+*1)) bytes long and shall contain all #00 bytes.

4 Logical Volume Data Structures

This section describes changes and additions to the UDF Specification at the Logical Volume level. These changes include:

- Creation of a single *User Identifier* Stream used to maintain User Identification information in regards to the users who have access to the secure information located on the media. The *User Identifier* stream is a system stream associated with the entire Logical Volume, and shall be located in the system stream directory of the File Set Descriptor.

The intended purpose of the *User Identifier* Stream is to provide a portable method for identifying *Users*. One possible method that this maybe accomplished is by storing a copy of the users X.509 certificate that contains all the necessary information to uniquely identify the *User*.

The very first time a new *User* is referenced by any of the other security descriptors defined is this document, a *User ID Record* shall be created within the *User Identifier* Stream. This *User ID Record* may be referenced by any of the other security descriptors defined in this document to uniquely identify the *User*.

4.1 User Identifier Stream

User Identifier Stream

Stream Name	Stream Location	Metadata Flag
"*UDF_UserID"	File Set Descriptor	1

4.1.1 Type 1 User ID Stream

Type 1 User ID Stream Format

BP	Length	Name	Contents
0	32	Implementation Identifier	EntityID
32	4	User ID Stream Type	Unit32
36	4	Number of User ID Records	Uint32
40	4	Index Number to be used	Uint32
44	84	Reserved	bytes
128	*	User ID Records	bytes

4.1.1.1 EntityID Implementation Identifier

For more information on the proper handling of this field see section 2.1.5 of UDF.

4.1.1.2 Uint32 User ID Stream Type

User ID Stream Type Interpretation

Type	Interpretation
0	Shall mean that the User ID Stream is not specified by this field.
1	Shall mean that the User ID Stream is a Type 1 User ID Stream.
2-63	Reserved
64-	Shall be subject to agreement between the originator and recipient of the medium.

4.1.1.3 Uint32 Number of User ID Records

This field shall specify the number of User ID Records. Deleted User ID Record shall not be counted.

4.1.1.4 Uint32 Index Number to be used

This field shall contain index number to be used for next record. This value shall be initialized to ZERO and incremented after creation of new record.

4.1.1.5 bytes Reserved

All bit of this field shall be set to ZERO.

4.1.1.6 bytes User ID Record

User ID Record

RBP	Length	Name	Contents
0	4	Record Length	Uint32
4	2	Flags	Uint16
6	4	Index Number	Uint32
10	4	Length of User ID (=L_UI)	Uint32
14	L_UI	User ID	bytes
14+ L_UI	*	Padding	bytes

4.1.1.7 Uint32 Record Length

This field shall contain the length of this User ID Record in bytes. Shall be a multiple of four bytes.

4.1.1.8 Uint16 Flags

Flags Interpretation

Bit	Interpretation
0-15	Reserved. Shall be set to ZERO.

4.1.1.9 Uint32 Index Number

This field shall contain index number corresponding to User ID field.

4.1.1.10 Uint32 Length of User ID

This field shall contain length of User ID.

4.1.1.11 bytes User ID

This field shall contain the User ID.

> Note: User ID may be X.509 certificate and so on.

4.1.1.12 bytes Padding

This field shall be ((Record Length) – (14+L_UI)) bytes long and shall contain all #00 bytes.

5 File & Directory Data Structures

This section describes additions to the UDF Specification at the file and directory level. These changes include the addition of a new extended attribute and several new system streams that can be associated with either a file or directory. The changes are as follows:

- Creation of a *Requirement Information* extended attribute to store a files security requirements in regards to access control, data privacy, data integrity and access logging.
- Creation of an *Access Control Stream* used to control access to the default stream as well as any other stream associated with the file entry.
- Creation of a *Data Privacy Stream* used to protect the contents of the default stream as well as any other stream associated with the file entry, through the use of encryption.
- Creation of a *Data Integrity Stream* used to assure data integrity of the default stream as well as any other stream associated with the file entry.
- Creation of an *Access Log Stream* used to provide an access audit trail of the default stream as well as any other stream associated with the file entry.
- Creation of a *License Stream* used to provide license control of the default stream.

5.1 Requirement Information Extended Attribute

The *Requirement Information* extended attribute shall be used to store security requirement information for the associated file. This extended attribute shall be stored as an Implementation Use Extended Attribute whose Implementation Identifier shall be set to:

"***UDF Secure Requirement**"

Note: A Secure UDF implementation shall apply the required functionality according to this information. If the implementation does not have required functionality, the implementation shall not access or modify the associated file or directory.

5.1.1 Requirement Info Extended Attribute Format

The Implementation Use area for this extended attribute shall be structured as follows:

Requirement Info format

RBP	Length	Name	Contents
0	2	Header Checksum	Uint16
2	2	Length of Required Function (=L_RF)	Uint16
4	L_RF	Required Functions	bytes

5.1.1.1 Uint16 Length of Required Function

This field shall specify length of Required Function field in bytes.

5.1.1.2 bytes Required Functions

Required Functions

Bit	Interpretation
0	Shall mean that Access Control is required.
1	Shall mean that Data Privacy is required.
2	Shall mean that Data Integrity is required.
3	Shall mean that Access Logging is required.
4-	Reserved

5.2 Access Control Stream

The *Access Control Stream* is used to control access to the default stream as well as any other stream associated with the file entry.

Access Control Stream

Stream Name	Stream Location	Metadata Flag
"*UDF_AccessControl"	Any file/directory	1

Type 1 Access Control Stream format

BP	Length	Name	Contents
0	32	Implementation Identifier	EntityID
32	4	Access Control Stream Type	Uint32
36	4	Number of Access Control Records	Unit32
40	88	Reserved	bytes
128	*	Access Control Records	bytes

5.2.1 EntityID Implementation Identifier

For more information on the proper handling of this field see section 2.1.5 of UDF.

5.2.2 Uint32 Access Control Stream Type

Access Control Stream Type Interpretation

Type	Interpretation
0	Shall mean that the Access Control Stream is not specified by this field.
1	Shall mean that the Access Control Stream is a Type 1 Access Control Stream.
2-63	Reserved
64-	Shall be subject to agreement between the originator and recipient of the medium.

5.2.3 Uint32 Number of Access Control Records

This field shall specify the number of Access Control Records. Deleted Access Control Record shall not be counted.

5.2.4 bytes Reserved

All bits of this field shall be set to ZERO.

5.2.5 bytes Access Control Records

Access Control Record Format

RBP	Length	Name	Contents
0	4	Record Length	Uint32
4	2	Flags	Uint16
6	1	Length of Stream Name (=L_SN)	Uint8
7	1	Reserved	Uint8
8	L_SN	Stream Name	dchars
8+L_SN	*1	Padding1	bytes
8+L_SN+ *1	4	Type of ACL	Uint32
12+ L_SN+*1	4	Permission	Uint32
16+ L_SN+*1	4	Type of ID	Uint32
20+ L_SN+*1	4	ID/index of ID	Uint32
24+ L_SN+*1	*2	Padding2	bytes

5.2.5.1 Uint32 Record Length

This field shall contain the length of this Access Control Record in bytes. Shall be a multiple of four bytes.

5.2.5.2 Uint16 Flags

Flags Interpretation

Bit	Interpretation
0	If this bit is set to ONE, this record is deleted. If this bit is set to ZERO, this record is under use.
1-15	Reserved. Shall be set to ZERO.

5.2.5.3 Uint8 Length of Stream Name

This field shall specify length of Stream Name. for all streams except for default stream. For default stream, this field shall be set to ZERO.

5.2.5.4 Uint8 Reserved

All bits of this field shall be set to ZERO.

5.2.5.5 dchars Stream Name

This field shall specify stream name for which MAC is calculated.

5.2.5.6 bytes Padding1

This field shall be 4 * ip((8+L_SN+3)/4)-(8+L_SN) bytes long and shall contain all #00 bytes.

5.2.5.7 Uint32 Type of ACL

Type of ACL Interpretation

Type	Interpretation
1	Shall mean that the type of ACL is “USER_OBJ”.
2	Shall mean that the type of ACL is “USER”.
4	Shall mean that the type of ACL is “GROUP_OBJ”.
8	Shall mean that the type of ACL is “GROUP”.
16	Shall mean that the type of ACL is “CLASS_OBJ”.
32	Shall mean that the type of ACL is “OTHER_OBJ”.
65536	Shall mean that the type of ACL is “ACL_DEFAULT”.
65537	Shall mean that the type of ACL is “DEF_USER_OBJ”.
65538	Shall mean that the type of ACL is “DEF_USER”.
65540	Shall mean that the type of ACL is “DEF_GROUP_OBJ”.
65544	Shall mean that the type of ACL is “DEF_GROUP”.
65552	Shall mean that the type of ACL is “DEF_CLASS_OBJ”.
65568	Shall mean that the type of ACL is “DEF_OTHER_OBJ”.

DEF_* can be set only to directory and corresponding permission shall be applied to all files and directories under the directory. Bit by bit product value of CLASS_OBJ and USER, GROUP_OBJ, GROUP is used for USER, GROPU_OBJ, GROUP as permissions respectively. Bit by bit product value of DEF_CALSS_OBJ and DEF_USER, DEF_GROUP_OBJ, DEF_GROUP is used for DEF_USER, DEF_GROUP_OBJ, DEF_GROUP as permissions respectively. Any other type are reserved.

5.2.5.8 Uint32 Permission

This field shall specify permission.

Permission Interpretation

Bit	Interpretation
0	Shall mean that the file can be read.
1	Shall mean that the file can be written.
2	Shall mean that the file can be executed.
3	Shall mean that the file can be deleted.
4-	Reserved. Shall be set to ZERO.

5.2.5.9 Uint32 Type of ID

This field shall contain type of ID (5.5.10.5).

5.2.5.10 Uint32 ID/index of ID

This field shall contain ID or index of ID according to the Type of ID field..

5.2.5.11 Bytes Padding2

This field shall be ((Record Length) – (24+L_SN+*1)) bytes long and shall contain all #00 bytes.

5.3 Data Privacy Stream

The *Data Privacy Stream* used to protect the contents of the default stream as well as any other stream associated with the file entry, through the use of encryption

Data Privacy Stream

Stream Name	Stream Location	Metadata Flag
"*UDF_DataPrivacy"	Any file/directory	1

Type 1 Data Privacy Stream Format

BP	Length	Name	Contents
0	32	Implementation Identifier	EntityID
32	4	Data Privacy Stream Type	Unit32
36	4	Number of Data Privacy Records	Uint32
40	88	Reserved	bytes
128	*	Data Privacy Records	bytes

5.3.1 EntityID Implementation Identifier

For more information on the proper handling of this field see section 2.1.5 of UDF.

5.3.2 Uint32 Data Privacy Stream Type

Data Privacy Stream Type Interpretation

Type	Interpretation
0	Shall mean that the Data Privacy Stream is not specified by this field.
1	Shall mean that the Data Privacy Stream is a Type 1 Data Privacy Stream.
2-63	Reserved
64-	Shall be subject to agreement between the originator and recipient of the medium.

5.3.3 Uint32 Number of Data Privacy Records

This field shall specify the number of Data Privacy Records. Deleted Data Privacy Record shall not be counted.

5.3.4 bytes Reserved

All bits of this field shall be set to ZERO.

5.3.5 bytes Data Privacy Records

Data Privacy Record

RBP	Length	Name	Contents
0	4	Record Length	Uint32
4	2	Flags	Uint16
6	2	Number of Encryption	Uint16
8	1	Length of Stream Name (=L_SN)	Uint8
9	1	Reserved	Uint8
10	L_SN	Stream Name	dchars
10+ L_SN	*1	Padding	bytes
10+ L_SN+*1	*2	Encryptions	Encspec[]
10+ L_SN+*1 +*2	*3	Padding	bytes

5.3.5.1 Uint32 Record Length

This field shall contain the length of this Data Privacy Record in bytes. Shall be a multiple of four bytes.

5.3.5.2 Uint16 Flags

Flags Interpretation

Bit	Interpretation
0	If this bit is set to ONE, this record is deleted. If this bit is set to ZERO, this record is under use.
1-15	Reserved. Shall be set to ZERO.

5.3.5.3 Uint16 Number of Encryption

This field shall specify number of Encryption.

5.3.5.4 Uint8 Length of Stream Name

This field shall specify length of Stream Name. for all streams except for default stream. For default stream, this field shall be set to ZERO.

5.3.5.5 Uint8 Reserved

All bit of this field shall be set to ZERO.

5.3.5.6 dchars Stream Name

This field shall specify stream name for which MAC is calculated.

5.3.5.7 bytes Padding

This field shall be 4 * ip((10+L_SN+3)/4)-(10+L_SN) bytes long and shall contain all #00 bytes.

5.3.5.8 Encspec Encryption
This field shall be specified in 3.3.2.1.

5.3.5.9 Padding
This field shall be ((Record Length) – (10+L_SN+*1+*2)) bytes long and shall contain all #00 bytes.

5.4 Data Integrity Stream

The *Data Integrity Stream* used to assure data integrity of the default stream as well as any other stream associated with the file entry, through the calculation of a Message Authentication Code (MAC).

Data Integrity Stream

Stream Name	Stream Location	Metadata Flag
"*UDF_DataIntegrity"	Any file/directory	1

Type 1 Data Integrity stream format

BP	Length	Name	Contents
0	32	Implementation Identifier	EntityID
32	4	Data Integrity Stream Type	Uint32
36	4	Number of MAC Records	Uint32
40	88	Reserved	bytes
128	*	MAC Records	bytes

5.4.1 EntityID Implementation Identifier

For more information on the proper handling of this field see section 2.1.5 of UDF.

5.4.2 Uint32 Data Integrity Stream Type

Data Integrity Stream Type Interpretation

Type	Interpretation
0	Shall mean that the Data Integrity Stream is not specified by this field.
1	Shall mean that the Data Integrity Stream is a Type 1 Data Integrity Stream.
2-63	Reserved
64-	Shall be subject to agreement between the originator and recipient of the medium.

5.4.3 Uint32 Number of MAC Records

This field shall specify the number of MAC Records. Deleted MAC Record shall not be counted.

5.4.4 bytes Reserved

All bits of this field shall be set to ZERO.

5.4.5 bytes MAC Record

MAC Record format

RBP	Length	Name	Contents
0	4	Record Length	Uint32
4	2	Flags	Uint16
6	1	Length of Stream Name (=L_SN)	Uint8
7	1	Reserved	Uint8
8	L_SN	Stream Name	dchars
8+ L_SN	*1	Padding	bytes
8+ L_SN+*1	2	MAC Calculation Type	Uint16
10+ L_SN+*1	16	Algorithm ID	encspec
26+ L_SN+*1	2	Length of MAC (=L_MA)	Uint16
28+ L_SN+*1	L_MA	MAC	bytes
28+ L_SN+*1 +L_MA	*2	Padding	bytes

5.4.5.1 Uint32 Record Length

This field shall contain the length of this MAC Record in bytes. Shall be a multiple of four bytes.

5.4.5.2 Uint16 Flags

Flags Interpretation

Bit	Interpretation
0	If this bit is set to ONE, this record is deleted. If this bit is set to ZERO, this record is under use.
1-15	Reserved. Shall be set to ZERO.

5.4.5.3 Uint8 Length of Stream Name

This field shall specify the length of Stream Name. for all streams except for the default stream. For the default stream, this field shall be set to ZERO.

5.4.5.4 Uint8 Reserved

All bits of this field shall be set to ZERO.

5.4.5.5 dchars Stream Name

This field shall specify the stream name for which the MAC is calculated.

5.4.5.6 bytes Padding

This field shall be 4 * ip((8+L_SN+3)/4)-(8+L_SN) bytes long and shall contain all #00 bytes.

5.4.5.7 Uint16 MAC Calculation Type

This field shall specify mode for MAC calculation.

MAC Calculation Type Interpretation

Type	Interpretation
0	Shall mean that the Calculation Type is not specified by this field.
1	Shall mean that the MAC is calculated from concatenated data of time stamp and stream body.
2	Shall mean that the MAC is calculated from concatenated data of time stamp, secret information and stream body.
3-63	Reserved
64-	Shall be subject to agreement between the originator and recipient of the medium.

Note: Secret information contains system identifier, drive identifier, media identifier and logical sector number.

5.4.5.8 Encspec Algorithm ID

This field shall specify an encspec as defined in 3.3.2.1.

5.4.5.9 Uint16 Length of MAC

This field shall specify the length of the MAC.

5.4.5.10 bytes MAC

This field shall specify MAC calculated from file or each stream or directory.

5.4.5.11 bytes Padding

This field shall be ((Record Length) – (28+L_SN+*1+L_MA)) bytes long and shall contain all #00 bytes.

5.5 Access Log Stream

The *Access Log Stream* is used to provide an access audit trail of the default stream as well as any other stream associated with the file entry.

Access Log Stream

Stream Name	Stream Location	Metadata Flag
"*UDF_AccessLog"	Any file/directory	1

Type 1 Access Log Stream Format

BP	Length	Name	Contents
0	32	Implementation Identifier	EntityID
32	4	Access Log Stream Type	Unit32
36	4	Number of Access Log Records	Uint32
40	4	Strategy of File Access Logging	Uint32
44	4	Strategy of Directory Access Logging	Uint32
48	8	Max Access Log Size	Uint64
56	8	Head Pointer	Uint64
64	8	Tail Pointer	Uint64
72	56	Reserved	bytes
128	*	Access Log Records	bytes

5.5.1 EntityID Implementation Identifier

For more information on the proper handling of this field see section 2.1.5 of UDF.

5.5.2 Uint32 Access Log Stream Type

Access Log Stream Type Interpretation

Type	Interpretation
0	Shall mean that the Access Log Stream is not specified by this field.
1	Shall mean that the Access Log Stream is a Type 1 Access Log Stream.
2-63	Reserved
64-	Shall be subject to agreement between the originator and recipient of the medium.

5.5.3 Uint32 Number of Access Log Records

This field shall specify the number of Access Log Records. Deleted Access Log Record shall not be counted.

5.5.4 Uint32 Strategy of File Access Logging

This field shall specify the strategy of access logging by bit mask of Action for file. If some bit is set to ONE, corresponding Action shall be recorded.

5.5.5 Uint32 Strategy of Directory Access Logging

This field shall specify the strategy of access logging by bit mask of Action for

directory. If some bit is set to ONE, corresponding Action shall be recorded.

5.5.6 Uint64 Max Access Log Size

This field shall specify allowed max size of access log. If size of access log reached this value, the oldest Access Log Record(s) shall be erased. Value ZERO means that the size of access log is not limited.

5.5.7 Uint64 Head Pointer

This field shall specify byte position of first byte of oldest Access Log Record in ring buffer.

5.5.8 Uint64 Tail Pointer

This field shall specify byte position of next byte of newest Access Log Record in ring buffer.

5.5.9 Uint64 Reserved

Reserved for future use. All bit of this field shall be set to ZERO.

5.5.10 bytes Access Log Record

Access Log Record Format

RBP	Length	Name	Contents
0	4	Record Length	Uint32
4	8	Sequence Number	Uint64
12	16	Action Time Stamp	timestamp
28	4	Action	Uint32
32	4	Type of User ID	Uint32
36	4	User ID/index of User ID	bytes
40	4	Length of Action Dependent Area (=L_AD)	Uint32
44	L_AD	Action Dependent Area	bytes
44+ L_AD	*	Padding	bytes

5.5.10.1 Uint32 Record Length

This field shall contain the length of this Access Log Record in bytes. Shall be a multiple of four bytes.

5.5.10.2 Uint64 Sequence Number

This field shall specify sequence number of Access Log Record. The sequence number is assigned in ascending order from value ZERO.

5.5.10.3 timestamp Action Time Stamp

This field shall contain date and time when action to a file has been taken. It shall indicate time that file is closed.

5.5.10.4 Uint32 Action

This filed shall contain action identifier.

If target is file, action interpretation shall be subject to following.

Action Interpretation (File Operation)

Bit	Interpretation
0	Make a target file secure
1	Make a target file un-secure
2	Read target file
3	Write target file
4	Reserved. Shall be set to ZERO.
5	Reserved. Shall be set to ZERO.
6	Truncate target file
7	Read attributes of target file
8	Write attributes of target file
9	Read a user stream under target file
10	Write a user stream under target file
11	Truncate a user stream under target file
12	Create a user stream under target file
13	Remove a user stream under target file
14	Rename a user stream under target file
15	Export target file and corresponding streams
16	Import target file and corresponding streams
17-31	Reserved. Shall be set to ZERO.

If target is directory, action interpretation shall be subject to following:

Action Interpretation (Directory Operation)

Bit	Interpretation
0	Make a target directory secure
1	Make a target directory un-secure
2	Read target directory
3	Create file/directory or hard link under target directory
4	Remove file/directory or hard link under target directory
5	Rename file/directory or hard link under target directory
6	Reserved. Shall be set to ZERO.
7	Read attributes of target directory
8	Write attributes of target directory
9	Read a user stream under target directory
10	Write a user stream under target directory
11	Truncate a user stream under target directory
12	Create a user stream under target directory
13	Remove a user stream under target directory
14	Rename a user stream under target directory
15	Export target directory and corresponding streams
16	Import target directory and corresponding streams
17-31	Reserved. Shall be set to ZERO.

5.5.10.5 Uint32 Type of User ID
This field shall contain type of User ID.

Type of User ID Interpretation

Type	Interpretation
0	Shall mean that the type of ID is not specified by this field.
1	Shall mean that the type of ID is POSIX user ID.
2	Shall mean that the type of ID is certificate of X.509
3-	Reserved

5.5.10.6 Bytes User ID or index of User ID
This field shall contain user ID or index of User ID by whom action to a file has been taken.

5.5.10.7 Uint32 Length of Action Dependent Area
This field contains length of action dependent area in bytes.

5.5.10.8 bytes Action Dependent Area
This field contains action dependent information.

If one bit from bit number 9 to bit number 14 is set to ONE, user stream name shall be set as following structure.

Action dependent area format

RBP	Length	Name	Contents
0	1	Length of Stream Name (=L_SN)	Uint8
1	1	Reserved	Uint8
2	L_SN	Stream Name	dchars
2+L_SN	*	Padding	bytes

5.5.10.8.1 Uint8 Length of Stream Name
This field shall specify length of stream name.

5.5.10.8.2 Uint8 Reserved
All bit of this field shall be set to ZERO.

5.5.10.8.3 dchars Stream Name
This field shall specify stream name.

5.5.10.8.4 bytes Padding
This field shall be ((L_AD-(2+L_SN)) bytes long and shall contain all #00 bytes.

If bit number 15 or bit number 16 is set to ONE, environment information shall be set as following structure.

envspec format

RBP	Length	Name	Contents
0	128	Logical Volume Identifier	dstring
128	6	Logical Block Address	lb_addr
134	*	Padding	bytes

5.5.10.8.5 dstring Logical Volume Identifier

This field shall contain Logical Volume Identifier where action has been taken.

5.5.10.8.6 lb_addr Logical Block Address

This field shall contain Logical Block Address where action has been taken.

5.5.10.8.7 bytes Padding

This field shall be (L_AD – 134) bytes long and shall contain all #00 bytes.

5.5.10.9 bytes Padding

This field shall be ((Record Length) – (44+L_AD)) bytes long and shall contain all #00 bytes.

5.6 License Stream

The *License Stream* used to provide license control of the default stream.

License Stream

Stream Name	Stream Location	Metadata Flag
"*UDF_License"	Any file	1

Type 1 License Stream format

BP	Length	Name	Contents
0	32	Implementation Identifier	EntityID
32	4	License Stream Type	Uint32
36	4	Number of License Records	Unit32
40	88	Reserved	bytes
128	*	License Records	bytes

5.6.1 EntityID Implementation Identifier

For more information on the proper handling of this field see section 2.1.5 of UDF.

5.6.2 Uint32 License Stream Type

License Stream Types

Type	Interpretation
0	Shall mean that the License Stream is not specified by this field.
1	Shall mean that the License Stream is a Type 1 License Stream.
2-63	Reserved
64-	Shall be subject to agreement between the originator and recipient of the medium.

5.6.3 Uint32 Number of License Records

This field shall specify the number of License Records. Deleted License Record shall not be counted.

5.6.4 bytes Reserved

All bit of this field shall be set to ZERO.

5.6.5 bytes License Record

License Record Format

RBP	Length	Name	Contents
0	4	Record Length	Uint32
4	*	License	bytes

5.6.5.1 Unit32 Record Length

This field shall contain the length of this License Record in bytes. Shall be a multiple of four bytes.

5.6.5.2 bytes License

This field shall contain license. Detail format of this field shall be subject to agreement between the originator and recipient of the medium.

6 Appendix A Application notes (Export/Import)

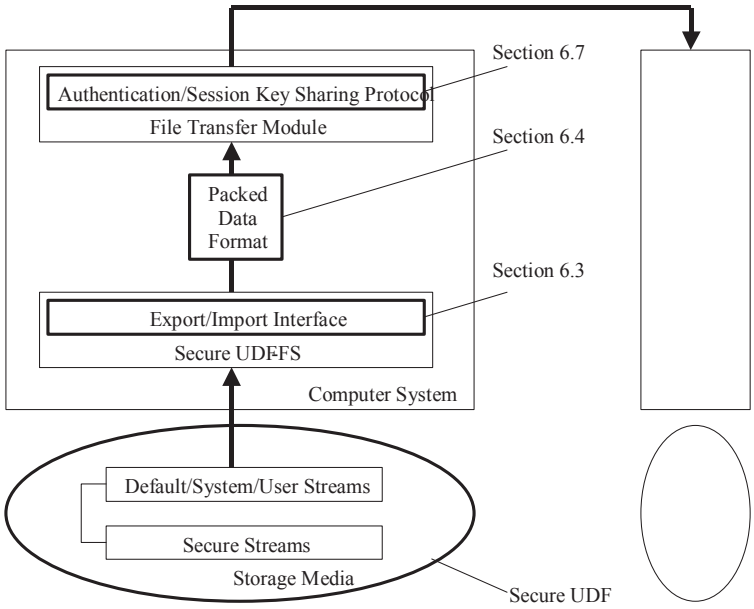
6.1 Introduction

Security information, such as MAC (Message Authentication Code) for data integrity, is stored as system named stream on a Secure UDF volume. When a default stream is transferred to another Secure UDF volume, the security information should be transferred with the default stream. Export/Import functionality provides a means to read/write all streams including system named streams, and create a structure called *Packed Data*. This *Packed Data* structure would contain all the necessary information (default stream, associated system streams, user streams, extended attributes and UserID information associated with the file) to transfer to another SecureUDF system using some form of communication. If needed, a MAC is added to the *Packed Data* in order to guarantee integrity of the packed data.

This appendix is a technical guide for implementers who would like to implement Export/Import functionality in a Secure UDF file system implementation.

6.2 Appendix Structure

Following figure shows the structure of this Appendix.



Section 6.3 describes file system API of Export/Import functionality.

Section 6.4 describes format of packed data that file transfer module can get through the Export/Import API.

Section 6.7 describes protocol for authentication, session key sharing between source and destination systems, and protocol for transfer of encrypted "packed data" from source system to destination system.

Note: Structure of the figure described above is an example of implementation. The other implementation can be allowed for each developer. For example, File transfer module can be integrated into Secure UDF-FS.

6.3 Export/Import Interface

This section describes what a file system API may look like for an Export/Import interface to a Secure UDF implementation. This interface could be provided through some type of operating system IOCTL interface.

6.3.1 UDF_OPENEXPORT

Name

UDF_OPENEXPORT request code – Prepare for exporting all streams specified by default stream, and the default stream

Synopsis

```
#include <udf_ioctl.h>

int ioctl(int fd, unsigned int cmd, struct UDF_OPENEXPORT_CMD_T
*arg)
```

Description

UDF_OPENEXPORT prepares for exporting all streams specified by defaults stream, and the default stream. Argument keyid shell be given to calculate MAC for packed data. If the file is not secured file or secured file that the data integrity feature is not applied, the key id is ignored. And also, if value 0xffffffffUL is given as keyid, default key is used for the calculation.

fd: File descriptor of default stream that user would like to export.

cmd: UDF_OPENEXPORT (request code)

arg: Address of argument structure

```
struct UDF_OPENEXPORT_CMD_T {
        unsigned int    keyid;  /* Keyid (Input)                */
}
```

Return Value

ioctl return ZERO if invocation succeed, or non-ZERO if error occurred.

Errors

EBADF fd is not a valid file descriptor.

ENOMEM There are enough memory for kernel.

EPERM	Implementation detects unauthorized modification for Secure file specified by fd.
ENODATA	Value 0xffffffffUL is given as keyid for target file on the system that the default key is not defined.

6.3.2 UDF_EXPORT

Name

UDF_EXPORT request code - Export Packed Data by specified block size

Synopsis

#include <udf_ioctl.h>

int ioctl(int fd, unsigned int cmd, struct UDF_EXPORT_CMD_T *arg)

Description

Read a packed data by specified size and store it into read buffer. If read pointer reaches end of packed data, ZERO value is set to count. size shall be bigger than 512 bytes and smaller than 4096, and the size shall be multiple integral of 512 bytes.

fd: File descriptor of default stream that user would like to export.

cmd: UDF_EXPORT (request code)

arg: Address of argument structure

```
struct UDF_EXPORT_CMD_T {
        char    *buf;   /* Buffer address (Input)              */
        int     size;   /* Buffer size (Input)          */
        int     count;  /* Bytes count exported (Output)      */
}
```

Return Value

ioctl return ZERO and bytes count exported is set to count if invocation succeed, or non-ZERO if error occurred.

ENOMEM	There are enough memory for kernel.
EBADF	fd is not a valid file descriptor.
EINVAL	size is not valid.
EFAULT	Address specified by buf is not valid.
ENETDOWN	Implementation detect any error on encryption, authentication or trusted time module.

6.3.3 UDF_CLOSEEXPORT

Name

UDF_CLOSEEXPORT request code – Release all resources used for export functionality

Synopsis

#include <udf_ioctl.h>

int ioctl(int fd, unsigned int cmd, struct UDF_CLOSEEXPORT_CMD_T *arg)

fd: File descriptor of default stream that user would like to export.

cmd: UDF_CLOSEEXPORT (request code)

arg: Address of argument structure

```
struct UDF_CLOSEEXPORT_CMD_T {
        /* Nothing */
}
```

Description

Release all resources used for export functionality.

Return Value

Always return ZERO value.

6.3.4 UDF_OPENIMPORT

Name

UDF_OPENIMPORT request code - Prepare for importing all streams specified by default stream, and the default stream

Synopsis

#include <udf_ioctl.h>

int ioctl(int fd, unsigned int cmd, struct UDF_OPENIMPORT_CMD_T *arg)

fd: File descriptor of file with size ZERO. The empty file shall be created in prior to invoke UDF_OPENIMPORT.

cmd: UDF_OPENIMPORT (request code)

arg: Address of argument structure

```
struct UDF_OPENIMPORT_CMD_T {
        unsigned int    keyid;          /* Key ID for packed data       */
        unsigned int    i_keyid;        /* Key ID for data integrity    */
}
```

Description

Prepare for importing all streams specified by default stream, and the default stream. If file descriptor that is not empty is specified, the file is overwritten. If file descriptor that is already secured, EBADF is returned. Argument keyid shall be given for checking integrity of packed data. Argument i_keyid shall be given for calculating MAC for data integrity.

Return Value

ioctl return ZERO if invocation succeed, or non-ZERO if error occurred.

EBADF	fd is not a valid file descriptor.
ENOTEMPTY	File specified by argument fd is not empty.
ENOMEM	There are enough memory for kernel.

6.3.5 UDF_IMPORT

Name

UDF_IMPORT request code - Import Packed Data by specified block size

Synopsis

#include <udf_ioctl.h>

int ioctl(int fd, unsigned int cmd, struct UDF_IMPORT_CMD_T *arg)

fd: File descriptor used for UDF_OPENIMPORT.

cmd: UDF_IMPORT (request code)

arg: Address of argument structure

```
struct UDF_IMPORT_CMD_T {
        char    *buf;   /* Buffer address       (Input) */
        int     size;   /* Buffer size (Input)  */
        int     count;  /* Bytes count imported(Output)      */
}
```

Description

Write a packed data stored in write buffer by specified size. If write pointer reaches end of packed data, count returns ZERO value. In this case implementation shall invoke UDF_CLOSEIMPORT. “size” shall be as same value as “size” specified in export.

Return Value

ioctl return ZERO and bytes count imported is set to count if invocation succeed, or non-ZERO if error occurred.

ENOMEM	There is not enough memory for kernel.
EBADF	fd is not a valid file descriptor.
EINVAL	size is not a valid or packed data is not valid.
EFAULT	Address specified by buf is not valid.
EBADE	Packed data is secure file and implementation detect

unauthorized modification of the packed data.

ENETDOWN Implementation detect any error on encryption, authentication or trusted time module.

EUMATCH Import function of packed data that requires data Integrity is directed on system that does not have MAC verification mechanism.

EMEDIUMTYPE File type of packed data is different from that of file specified by fd.

6.3.6 UDF_CLOSEIMPORT

Name

UDF_CLOSEIMPORT request code - Release all resources used for import functionality

Synopsis

#include <udf_ioctl.h>

int ioctl(int fd, unsigned int cmd, struct UDF_CLOSEIMPORT_CMD_T *arg)

fd: File descriptor used for UDF_OPENIMPORT.

cmd: UDF_CLOSEIMPORT (request code)

arg: Address of argument structure

```
struct UDF_CLOSEIMPORT_CMD_T {
        /* Nothing */
}
```

Description

Release all resources used for import functionality

Return Value

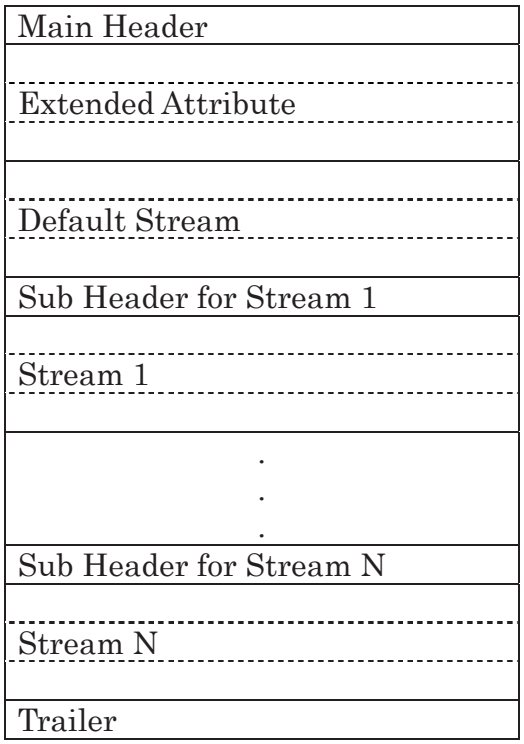
Always return ZERO value.

6.4 Packed Data

Export/Import functionality of Secure UDF file system handles “Packed Data”. Packed Data consist of default stream and all user/system streams corresponding to the default stream. Using this functionality, file system implementation can transfer all streams to another file system through network. “Packed Data Format” specifies structure of Packed Data.

6.4.1 Packed Data Format

Structure of Packed Data is shown below.



Length of each box is specified by block size as argument of EXPORT/IMPORT function. If the box cannot be filled with data, padding shall be inserted. The padding contains all #00 bytes.

Packed Data shall contain UserID stream specified by file set descriptor if some stream contains data that relate to UserID stream. For example, User ID field of access log stream may contain index of X.509 certificate, and mapping between the index and the X.509 certificate is defined in the UserID stream. UserID stream shall be stored as first sub-stream in the packed data.

6.4.2 tag : SUDF_PACKDATA_TAG_T

BP	Length	Name	Contents
0	2	tagcrc	Uint16
2	2	tagcrclen	Uint16
4	2	tagid	Uint16
6	2	tagversion	Uint16
8	8	reserve	bytes

6.4.2.1 Uint16 tagcrc
This field shall specify CRC value calculated from specified area. The area shall be specified in next field.

6.4.2.2 Uint16 tagcrclen
This field shall specify length of area calculated by CRC.

6.4.2.3 Uint16 tagid
This field shall specify tag identifier used for Packed Data.

Type	Interpretation
1	Main Header
2	Sub Header
3	Trailer

6.4.2.4 Uint16 tagversion
This field shall specify version number of Packed Data. Upper one byte shall contain major number and lower one byte shall contain minor number. (=1.1)

6.4.3 Main Header : SUDF_PACKDATA_MAIN_HEADER_T

BP	Length	Name	Contents
0	16	tag	SUDF_PACKDATA_TAG_T
16	2	blksiz	Uint16 (*1)
18	2	Padding	bytes (all #00)
20	4	entmax	Uint32 (*2)
24	4	keyid	Uint32 (*2)
28	8	mac (padding1)	UDFSVC_MAC_T (*3) (bytes)
36	20	icbtag	UDF_icbtag (*4)
56	4	uid	Uint32 (*2)
60	4	gid	Uint32 (*2)
64	4	permissions	Uint32 (*2)
68	8	informlen	Uint64 (*5)
76	12	actime	UDF_timestamp (*6)
88	12	motime	UDF_timestamp (*6)
100	12	crtime	UDF_timestamp (*6)
112	12	attime	UDF_timestamp (*6)
124	4	inoflgs	Uint32 (*2)
128	4	i_keyid	Uint32 (*2)
132	8	i_mac (padding2)	UDFSC_MAC_T (*3) (bytes)
140	4	log_strategy	Uint32 (*2)
144	4	ealen (=L_EA)	Uint32
148	364	reserve	bytes

Note: In case that kernel is configured as “Secure UDF does not supported”, definitions in braces are applied.

Note: This specification allows only a Triple DES-MAC algorithm to generate the MAC.

6.4.3.1 tag
See 3.1. tagid = 1.

6.4.3.2 blksiz
This field shall specify size of block when export functionality is invoked. This value shall be as same as size field of UDF_EXPORT_CMD_T.

6.4.3.3 entmax
This field shall specify number of exported streams including default stream, all system streams and all user streams.

6.4.3.4 keyid
This field shall specify key identifier used for generating/verifying MAC (6.4.3.5).

6.4.3.5 mac
This field shall specify MAC calculated from SUDF_PACKDATA_MAIN_HEADER_T using key specified by keyid(6.4.3.4). In this calculation, keyid, i_keyid, mac and i_mac shall be set to ZERO.

6.4.3.6 icbtag
This field shall specify icbtag(ECMA 167 4/14.9, 4/14.17) for default stream.

6.4.3.7 uid
This field shall specify user identifier(ECMA 167 4/14.9, 4/14.17) for default stream.

6.4.3.8 gid
This field shall specify group identifier(ECMA 167 4/14.9, 4/14.17) for default stream.

6.4.3.9 permissions
This field shall specify permissions(ECMA 167 4/14.9, 4/14.17) for default stream.

6.4.3.10 informlen
This field shall specify size(ECMA 167 4/14.9, 4/14.17) of default stream.

6.4.3.11 actime
This field shall specify last access time(ECMA 167 4/14.9, 4/14.17) for default stream.

6.4.3.12 motime
This field shall specify last modification time(ECMA 167 4/14.9, 4/14.17) for default stream.

6.4.3.13 crtime
This field shall specify creation time(ECMA 167 4/14.9, 4/14.17) for default stream.

6.4.3.14 attime
This field shall specify last modification time(ECMA 167 4/14.9, 4/14.17) of attribute for default stream.

6.4.3.15 inoflgs

This field shall specify expanded attribute for default stream.

Bit	Interpretation
0	i_meta flag of extended inode structure (means stream is system stream)
1	i_secure flag of extended inode structure (means file is secured)
2	Reserved
3	i_integrity flag of extended inode structure (means that integrity check shall be applied)
4	i_logging flag of extended inode structure (means that access logging shall be applied)
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	validity of i_stream_len field of extended inode structure (shows existence of stream directory)
10-31	reserved

6.4.3.16 i_keyid

This field shall specify key identifier used for integrity check functionality for default stream.

6.4.3.17 i_mac

This field shall specify MAC calculated from SUDF_PACKDATA_MAIN_HEADER_T using i_keyid. In this calculation, keyid, i_keyid, mac and i_mac shall be set to ZERO.

6.4.3.18 log_strategy

This field shall specify strategy of access logging.

6.4.3.19 ealen

This field shall specify length of extended attribute.

6.4.4 Sub-Header : SUDF_PACKDATA_SUB_HEADER_T

BP	Length	Name	Contents
0	16	tag	SUDF_PACKDATA_TAG_T
16	20	icbtag	UDF_icbtag (*4)
36	4	uid	Uint32 (*2)
40	4	gid	Uint32 (*2)
44	4	permissions	Uint32 (*2)
48	8	informlen	Uint64 (*5)
56	12	actime	UDF_timestamp (*6)

68	12	motime	UDF_timestamp (*6)
80	12	crtime	UDF_timestamp (*6)
92	12	attime	UDF_timestamp (*6)
104	4	inoflgs	Uint32 (*2)
108	2	namelen (=L_NM)	Uint16 (*1)
110	L_NM + 1	name	bytes
111+L_NM	401-L_NM	reserved	bytes

6.4.4.1 tag
See 3.1.tagid = 2.

6.4.4.2 icbtag
This field shall specify icbtag(ECMA 167 4/14.9,4/14.17) for corresponding stream.

6.4.4.3 uid
This field shall specify user identifier(ECMA 167 4/14.9, 4/14.17) for corresponding stream.

6.4.4.4 gid
This field shall specify group identifier(ECMA 167 4/14.9, 4/14.17) for corresponding stream.

6.4.4.5 permissions
This field shall specify permissions(ECMA 167 4/14.9, 4/14.17) for corresponding stream.

6.4.4.6 informlen
This field shall specify size(ECMA 167 4/14.9, 4/14.17) of corresponding stream.

6.4.4.7 actime
This field shall specify last access time(ECMA 167 4/14.9, 4/14.17) for corresponding stream.

6.4.4.8 motime
This field shall specify last modification time(ECMA 167 4/14.9, 4/14.17) for corresponding stream.

6.4.4.9 crtime
This field shall specify creation time(ECMA 167 4/14.17) for corresponding stream.

6.4.4.10 attime
This field shall specify last modification time(ECMA 167 4/14.9, 4/14.17) of attribute for corresponding stream.

6.4.4.11 inoflgs
This field shall specify expanded attribute for default stream. See 3.2.15.

6.4.4.12 namelen
This field shall specify length of stream name.

6.4.4.13 name
This field shall specify stream name.

6.4.5 Trailer : SUDF_PACKDATA_TRAILER_T

Note: In case that kernel is configured as “Secure UDF does not supported”, definitions in braces are applied.

BP	Length	Name	Contents
0	16	tag	SUDF_PACKDATA_TAG_T
16	2	crc	Uint16 (*1)
18	2	Padding	bytes (all #00)
20	8	crclen	Uint64 (*5)
28	4	userid	Uint32 (*2)
32	12	timestamp	UDF_timestamp (*6)
44	8	mac (padding)	UDFSVC_MAC_T (*3) (bytes)
52	460	reserved	bytes

6.4.5.1 tag

See 3.1. tagid = 3.

6.4.5.2 crc

This field shall specify CRC value calculated from all packed data except for trailer.

6.4.5.3 crclen

This field shall specify length of all packed data except for trailer in bytes.

6.4.5.4 userid

This field shall specify user identifier who invoked export functionality. This value is used for verifying MAC.

6.4.5.5 timestamp

This field shall specify date and time when export functionality is invoked. This value is used for verifying MAC.

6.4.5.6 mac

This field shall specify MAC value calculated from all packed data except for trailer, userid(6.4.5.4) and timestamp(6.4.5.5) using key specified by keyid(6.4.3.4).

6.4.5.7 Data (Default stream and each stream)

This field shall contains default stream, each user/system streams.

6.5 Processing Flow of Authentication

When import functionality is invoked, implementation shall perform authentication of packed data described below.

1) Perform CRC Check of main header, each sub-header and trailer using each tagcrclen and tagcrc(6.4.2.1).

2) If packed data is secure file, that data integrity functionality was applied implementation shall perform check whether keys specified by keyid and i_keyid are as same as keys on export side host (using mac, i_mac of SUDF_PACKDATA_MAIN_HEADER_T. See 6.4.3.5, 6.4.3.17).

3) If packed data is secure file, that data integrity functionality was applied, implementation shall check authenticity of all packed data except for trailer using keyid and mac of main header.

4) If packed data is not secure file, that data integrity functionality was applied, implementation shall check CRC of all packed data except for trailer.

6.6 A Supplementary Explanation

Following are relationship between arguments of Export/Import API and Packed Data.

- UDF_OPENEXPORT_CMD_T

  keyid
  (If 0xffffffffUL is specified)
  Default key identifier specified by udfformat(8) is used and the key identifier is stored in keyid field of SUDF_PACKDATA_MAIN_HEADER_T.
  (If 0xffffffffULis not specfied)
  Given argument “keyid” is stored in keyid field of SUDF_PACKDATA_MAIN_HEADER_T.

- UDF_EXPORT_CMD_T

  Given argument “size” is stored in blksiz field of SUDF_PACKDATA_MAIN_HEADER_T.

- UDF_OPENIMPORT_CMD_T

  keyid
  (If 0xffffffffUL is specified)
  A value in keyid field of SUDF_PACKDATA_MAIN_HEADER_T is used for Import.
  (If 0xffffffffULis not specified)
  Given argument “keyid” is used for Import.

  i_keyid
  (If 0xffffffffUL is specfied)
  A value in i_keyid field of SUDF_PACKDATA_MAIN_HEADER_T is used for integrity check functionality of default stream.

  (If 0xffffffffUL is not specified)
  Given argument “i_keyed” is used for integrity check functionality of default stream.

- UDF_IMPORT_CMD_T

  Given argument “size” is stored in blksiz field of SUDF_PACKDATA_MAIN_HEADER_T.

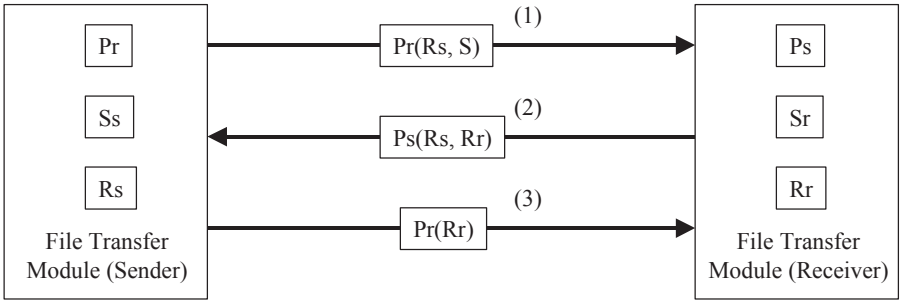
---

*1: Uint16	: ECMA 167 1/7.1.3
*2: Uint32	: ECMA 167 1/7.1.5
*3: UDFSVC_MAC_T	: sudf_security.h
*4: UDF_icbtag	: ECMA 167 4/14.6
*5: Uint64	: ECMA 167 1/7.1.7
*6: UDF_timestamp	: ECMA 167 1/7.3

6.7 Authentication/Session Key Sharing Protocol

There are many protocols(*1) to authenticate each other and share session key used for encrypting data. File transfer module can use preferable protocol under agreement between sender and receiver. Following figure shows an example of the protocol briefly.

(*1) ISO/IEC 11770-3:1999, Information technology – Security techniques – Key management – Part3: Mechanisms using asymmetric techniques



Ps : Public key of sender
Pr : Public key of receiver
Ss : Secret key of sender
Sr : Secret key of receiver
Rs : Random number generated on sender
Rr : Random number generated on receiver
S : Information indicates that sender is S

(1) Sender generates a random number Rs, concatenates it with S, encrypts it by public key of receiver and sends it to receiver.
(2) Receiver decrypts encrypted Rs using Sr. Receiver generates a random number Rr, concatenates it with Rs, encrypts it by Ps and sends it to sender.
(3) Sender decrypts encrypted Rs and Rr and checks whether Rs is as same as that sender generated phase (1). If same, sender send back receiver Rr encrypted by Pr. Receiver checks whether Rr is as same as that receiver generated phase (2). If same, both sender and receiver create session key from Rs and Rr using function f(Rs, Rr). In general, function may be exclusive or of Rs and Rr.

The session key is used to encrypt data to be transferred from sender to receiver, and passed to file system API to calculate MAC in file system.

Note: Transferring exported data to medium that have no physical secure area from medium that have physical secure area may cause a decline of security level. It depends on security policy on each systems whether the system can allow the transfer or not.