VMS Software

VSI OpenVMS

DECnet-Plus FTAM Programming

Document Number: DO-FTAMPG-01A

Publication Date: April 2024

Operating System and Version: VSI OpenVMS IA-64 Version 8.4-1H1 or higher
VSI OpenVMS Alpha Version 8.4-2L1 or higher

VMS Software, Inc. (VSI)
Boston, Massachusetts, USA

DECnet-Plus FTAM Programming

VMS Software

Legal Notice

Preface

This manual provides information about the FTAM application programming interface (FTAM API) that is part of the DECnet-Plus product set.

1. About VSI

VMS Software, Inc. (VSI) is an independent software company licensed by Hewlett Packard Enterprise to develop and support the OpenVMS operating system.

2. Intended Audience

The audience for this manual is OSI application programmers who require a basic understanding of the upper-layer standards implemented by FTAM (File Transfer, Access, and Management) product.

3. Related Documents

VSI DECnet-Plus FTAM and Virtual Terminal Use and Management provides additional information on the FTAM software.

Read the *Release Notes* before you read any other document in this set.

4. VSI Encourages Your Comments

You may send comments or suggestions regarding this manual or any VSI document by sending electronic mail to the following Internet address: `<docinfo@vmssoftware.com>`. Users who have VSI OpenVMS support contracts through VSI can contact `<support@vmssoftware.com>` for help with this product.

5. OpenVMS Documentation

The full VSI OpenVMS documentation set can be found on the VMS Software Documentation webpage at https://docs.vmssoftware.com.

6. Typographical Conventions

VMScluster systems are now referred to as OpenVMS Cluster systems. Unless otherwise specified, references to OpenVMS Cluster systems or clusters in this document are synonymous with VMScluster systems.

The contents of the display examples for some utility commands described in this manual may differ slightly from the actual output provided by these commands on your system. However, when the behavior of a command differs significantly between OpenVMS Alpha and Integrity servers, that behavior is described in text and rendered, as appropriate, in separate examples.

In this manual, every use of DECwindows and DECwindows Motif refers to DECwindows Motif for OpenVMS software.

The following conventions are also used in this manual:

Convention	Meaning
Ctrl/*x*	A sequence such as **Ctrl/** *x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 *x*	A sequence such as PF1 *x* indicates that you must first press and release the key labeled PF1 and then press and release another key or a pointing device button.
Return	In examples, a key name enclosed in a box indicates that you press a key on the keyboard. (In text, a key name is not enclosed in a box.)
…	A horizontal ellipsis in examples indicates one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
( )	In command format descriptions, parentheses indicate that you must enclose the options in parentheses if you choose more than one.
[ ]	In command format descriptions, brackets indicate optional choices. You can choose one or more items or no items. Do not type the brackets on the command line. However, you must include the brackets in the syntax for OpenVMS directory specifications and for a substring specification in an assignment statement.
[|]	In command format descriptions, vertical bars separate choices within brackets or braces. Within brackets, the choices are options; within braces, at least one choice is required. Do not type the vertical bars on the command line.
{ }	In command format descriptions, braces indicate required choices; you must choose at least one of the items listed. Do not type the braces on the command line.
bold text	This typeface represents the introduction of a new term. It also represents the name of an argument, an attribute, or a reason.
italic text	Italic text indicates important information, complete titles of manuals, or variables. Variables include information that varies in system output (Internal error *number*), in command lines (/PRODUCER= *name*), and in command parameters in text (where *dd* represents the predefined code for the device type).
UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
`Monospace type`	Monospace type indicates code examples and interactive screen displays. In the C programming language, monospace type in text identifies the following elements: keywords, the names of independently compiled external functions and files, syntax summaries, and references to variables or identifiers introduced in an example.
-	A hyphen at the end of a command format description, command line, or code line indicates that the command or statement continues on the following line.
numbers	All numbers in text are assumed to be decimal unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

Other conventions are:

- All numbers are decimal unless otherwise noted.
- All Ethernet addresses are hexadecimal.

Chapter 1. Introduction

Accessing the FTAM protocol through a programmable interface, such as FTAM application programming interface (FTAM API), requires a basic understanding of the portions of the complex FTAM protocol that are supported by FTAM products. This manual explains the syntax and programming codes of the FTAM API.

FTAM products are communications products that support file transfer and basic file management between open systems. An open system is a computer system that contains implementations of the seven layers of the Open Systems Interconnection (OSI) Reference Model for communications.

The FTAM API provides an interface to the FTAM protocol machine. The FTAM API is consistent with all the specified FTAM file service primitives and with the structure and scope of other OSI upper layer programming interfaces.

The FTAM API provides the functions specified in the FTAM standard (ISO 8571-3) by supporting the following:

- Functional Units
    - Read
    - Write
    - File Access
    - Limited File Management
    - Enhanced File Management
    - Grouping
    - FADU Locking
- Service Classes
    - Unconstrained
    - Management
    - Transfer
    - Transfer and Management
    - Access
- Attribute Groups
    - Storage
    - Security
- Document Types
    - FTAM-1

- FTAM-2
- FTAM-3
- NBS-9

Note

This document describes both the FTAM API Version 3.0 and the FTAM API Version 3.2. Sections specific to either Version 3.0 or Version 3.2 are designated in the text. See Section 1.10 for issues around using FTAM API applications written to earlier versions of the FTAM API with Version 3.2.

1.1. Overview of the FTAM API

The FTAM API is a low-level interface providing access to the FTAM protocol machine. To use the FTAM API, you must have a good working knowledge of the FTAM standard. You should obtain a copy of the FTAM standard (ISO 8571) for active reference purposes.

The FTAM API consists of the following function calls. Chapter 5 describes these calls in detail.

- `osif_assign_port` and `osif_deassign_port` are used to create and tear down the connection to the remote system.
- `osif_give_buffer` is used locally to pass buffers to the FTAM API.
- `osif_send` and `osif_get_event` are used to send and receive FTAM service primitives to and from the remote system.

Similar to the OSAK API, the FTAM API is a parameter block interface. That parameter block is the `osifpb` structure. The `osifpb` is used by the FTAM API user to specify which FTAM service primitive to send and what the parameters should be. It is also used by the FTAM API to tell the API user which FTAM service primitive was received and what the parameters are. The `osifpb` structure contains a field for each parameter in any of the FTAM primitives. In this document, **parameter** refers to a field of the `osifpb` structure.

Chapter 4 describes the `osifpb` structure and the other structures used by the FTAM API.

1.2. Using the FTAM API

This section shows a typical way that the FTAM API can be used to establish an association and perform a protocol exchange sequence with the peer entity.

1.2.1. Setting up an Association

The first call to the FTAM API is to the routine `osif_assign_port`. This routine returns a port identifier which is the local identifier of the association.

Before requesting any additional services, use the `osif_give_buffer` call to provide FTAM with buffers for receiving inbound events. FTAM returns these buffers on subsequent `osif_get_event` calls.

1.2.2. Typical FTAM Protocol Exchange

A typical FTAM protocol exchange can resemble the following:

Operation	**FTAM Primitives**
Send:	f-initialize-request
Receive:	f-initialize-response
Send:	f-begin-group-request f-create-request f-open-request f-end-group-request
Receive:	f-begin-group-response f-create-response f-open-response f-end-group-response
Send:	f-write-request
Send:	f-data-request, f-data-request, ...
Send:	f-data-end-request
Send:	f-transfer-end-request
Receive:	f-transfer-end-response
Send:	f-terminate-request
Receive:	f-terminate-response

1.2.3. Typical FTAM API Call Sequence

To implement this typical protocol exchange, you should expect to see the following calls to the FTAM API:

Call	**Purpose**
`osif_send`	send the f-initialize-request
`osif_get_event`	receive the f-initialize-response
`osif_send`	send the f-begin-group-request
`osif_send`	send the f-create-request
`osif_send`	send the f-open-request
`osif_send`	send the f-end-group-request
`osif_get_event`	receive the f-begin-group-response
`osif_get_event`	receive the f-create-response
`osif_get_event`	receive the f-open-response
`osif_get_event`	receive the f-end-group-response
.	

Call	Purpose
.	
.	

1.3. Mapping Block Types to Primitives

The FTAM service primitives available through the FTAM API map to a set of constants that must be set in the `osif_block_type` parameter of the `osifpb` structure. These constants are described in Section 4.1.

These values determine the type of FTAM service primitive that a particular `osifpb` structure represents. The block type will be set to one of the defined constants upon receipt of an incoming FTAM event. A single value refers to either a request or an indication depending on the context in which the `osifpb` structure is used.

For example, if an F-INITIALIZE-request is to be sent to an FTAM responder,then the `osif_block_type` parameter must be set to OSIF_PBDEF_INIT_REQ. The `osif_send` function call can then be used to send the F-INITIALIZE-request to the remote responder. If the responder receives an `osifpb` as part of an `osif_get_event` function call and the `osif_block_type` parameter is set to OSIF_PBDEF_INIT_REQ, then the `osifpb` received describes an F-INITIALIZE-indication.

1.4. Handling API Calls

Except for the `osif_get_event` call, all API calls are blocking in nature. Blocking means that control does not return to the user program until the requested call has completed successfully or unsuccessfully. The `osif_get_event` call may be used either in blocking or non-blocking mode. If it is used in non-blocking mode, the `osif_get_event` call can poll for events by using the timeout parameter.

1.5. Managing the User Buffer

The API must be provided with buffers after a port is assigned but before any other operations. This is accomplished with calls to `osif_give_buffer`. Once you give the API a buffer and the structure that points at the buffer (`osif_buffer_list` structure) through the `osif_give_buffer` function call, the API owns the buffer. You should not try to use the buffer until the APIreturns the buffer to you. Buffers are returned to the user when you deassign the port using the `buffer_list` argument of the `osif_deassign_port` function call or when an event is received by the `osif_get_event` function call.

If the API has insufficient buffers to complete an `osif_get_event` request, the call returns an OSIF_FAILURE status and the OSIF_NOBUFFS error. If this situation occurs, your application should provide additional buffers using the `osif_give_buffer` call and retry the `osif_get_event` call.

To avoid this situation, use the `osif_give_buffer` call before each call to `osif_get_event`:

```
osif_give_buffer(...)osif_get_event(...)
```

When an event is received successfully, the `osif_get_event`call might return user buffers in the `osifpb` structure using the `osif_returned_buffer` parameter. The `osif_returned_buffer` parameter might also return a null value. The

`osif_returned_buffer` parameter is a pointer to the `osif_buffer_list` structure which points at the user buffer. Once you have finished using all the parameters in an `osifpb` structure, you can reuse the buffers that the `osif_returned_buffer` parameter points at by passing them back to the API using the `osif_give_buffer` call. Do not return buffers to the API before you are finished using them, because information might be lost in the process.

1.6. Handling User Data

The user data, which corresponds to the F-DATA service primitive, is passed to the FTAM API through the `osif_userdata` parameter of the `osifpb`. `osif_userdata` is implemented as an `osif_sdesc` structure, which contains a pointer to the data buffer containing the user's file data, and an integer which is the buffer length. Only one buffer can be used for each transfer. The FTAM API does not support buffer chains. The FTAM API owns the buffer (that is, the user should not change it) until the call returns. If the buffer contents change, the unpredictable results might cause the call to fail.

For outgoing F-DATA requests, the buffer used is supplied by the user. Its allocation is a local matter. The data buffer becomes available to the user when the `osif_send` function returns.

For incoming F-DATA indications, the specified buffer in the `osif_userdata` parameter is taken from the `osif_give_buffer` pool. The buffer may or may not become available to the user upon receipt of the F-DATA-indication (`osif_get_event` call). Buffers will be owned by the API until the API relinquishes control of the buffer through the use of the `osif_returned_buffer` parameter.

1.7. Using Document Types

Document types provide information about a file, including its intended use, structure, and scope. The four supported document types are:

- FTAM-1 — Unstructured text files
- FTAM-2 — Sequential text files
- FTAM-3 — Unstructured binary files
- NBS-9 — NBS file directories

1.7.1. Using the FTAM-1 Document Type

An FTAM-1 document type indicates that data is sent as a stream of characters. The buffer and buffer length must be specified in the `osif_userdata` parameter. Carriage control conversion is not supplied by the API and must be performed by the API user before passing the F-DATA to the API. The FTAM protocol machine handles the encoding of the data. Optional escape sequences specifying ISO character set designation are stripped from incoming data before it is delivered to the user by the FTAM protocol machine.

The following values are supported for the FTAM-1 document type parameters:

Maximum string length	OSIF_MSL_UNLIMITED	Unlimited
	Integer>0	Size given
String significance	OSIF_STRSIG_VAR	Variable
	OSIF_STRSIG_FIX	Fixed
	OSIF_STRSIG_NS	Not significant

Universal class number	OSIF_UC_PRINTABLE	PrintableString
	OSIF_UC_IA5	IA5String
	OSIF_UC_GRAPHIC	GraphicString
	OSIF_UC_VISIBLE	VisibleString
	OSIF_UC_GENERAL	GeneralString

1.7.2. Using the FTAM-2 Document Type

An FTAM-2 document type indicates that data is delivered to the user one file access data unit (FADU) at a time. The user receives one `osifpb` structure per FADU which points to the FADU through the `osif_userdata` parameter.

The following values are supported for the FTAM-2 document type parameters:

Maximum string length	OSIF_MSL_UNLIMITED	Unlimited
	Integer>0	Size Given
String significance	OSIF_STRSIG_VAR	Variable
	OSIF_STRSIG_FIX	Fixed
	OSIF_STRSIG_NS	Not significant
Universal class number	OSIF_UC_PRINTABLE	PrintableString
	OSIF_UC_IA5	IA5String
	OSIF_UC_GRAPHIC	GraphicString
	OSIF_UC_VISIBLE	VisibleString
	OSIF_UC_GENERAL	GeneralString

1.7.3. Using the FTAM-3 Document Type

An FTAM-3 document type indicates that data is sent as a stream of octets. The buffer and buffer length must be specified in the `osif_userdata` parameter. The FTAM protocol machine handles the encoding of the data. Optional escape sequences specifying ISO character set designation are stripped from incoming data before it is delivered to the user by the FTAM protocol machine.

The following values are supported for the FTAM-3 document type parameters:

Maximum string length	OSIF_MSL_UNLIMITED	Unlimited
	Integer>0	Size Given
String significance	OSIF_STRSIG_VAR	Variable
	OSIF_STRSIG_FIX	Fixed
	OSIF_STRSIG_NS	Not significant

1.7.4. Using the NBS-9 Document Type

An NBS-9 document type indicates that FTAM initiators can read the contents of a remote directory file. More information on the NBS-9 document type is in the NIST special publication, *Stable Implementation Agreements for Open Systems Interconnection Protocols Version 2 Edition 1*. The file contents are defined by the following abstract syntax (NBS-AS2):

```
NBS_AS2 DEFINITIONS ::=
BEGIN
FileDirectoryEntry ::=[PRIVATE 2] Read-Attributes
ReadAttributes ::=ISO8571-FTAM.ReadAttributes
End
```

Thus, the file contents consist of records that are made up of the syntax of an F-READ-ATTRIBUTE-response primitive. As a result, API users receiving NBS-9 data receive an `osifpb` structure with all the relevant F-READ-ATTRIBUTE-response parameters filled in for each entry in the remote directory.

The NBS-9 document type parameters are defined as a bit string in the `osif_attribute_names` parameter of the `osifpb`. The bit string consists of the following bits:

read-filename(0)
read-permitted-actions(1)
read-contents-type(2)
read-storage-account(3)
read-date-and-time-of-creation(4)
read-date-and-time-of-last-modification(5)
read-date-and-time-of-last-read-access(6)
read-date-and-time-of-last-attribute-modification(7)
read-identity-of-creator(8)
read-identity-of-last-modifier(9)
read-identity-of-last-reader(10)
read-identity-of-last-attribute-modifier(11)
read-file-availability(12)
read-filesize(13)
read-future-filesize(14)
read-access-control(15)
read-legal-qualifications(16)
read-private-use(17)

1.7.5. Default Document Type Parameter Values

For the ISO document types (FTAM-1, FTAM-2, and FTAM-3), the following default parameter values apply if one of the supported values listed for each document type is not specified.

- If the maximum string length parameter is not specified, then the default value is unlimited.
- There is no default for the string significance parameter. If the parameter is not specified, then the length of the character strings is less than or equal to the maximum string length given.
- If the universal class number parameter is not specified, then the default value is GraphicString.

1.8. Passive Versus Active Responders

With the FTAM API, you can design an FTAM responder that is either an active application or a passive application.

To use a passive application, do the following:

- Specify OSIF_ASSIGN_REDIRECT as the value for the `port_flags`argument of the `osif_assign_port` call.

- On OpenVMS, declare your FTAM responder as an OSAK application. Refer to the *VSI DECnet-Plus FTAM and Virtual Terminal Use and Management* for instructions on how to perform this operation. Once your responder is declared, the OSAK Server starts up your responder when a connection request arrives for its address.
- On UNIX, start the FTAM listener and specify your FTAM responder as the responder to use. Refer to the *VSI DECnet-Plus FTAM and Virtual Terminal Use and Management* for information about starting up a listener.

  Once started, the FTAM listener starts up your responder when a connection request arrives for the specified address.

To use an active application, do the following:

- Specify OSIF_ASSIGN_RESPONDER as the value for the `port_flags` argument of the `osif_assign_port` call.
- On both OpenVMS and UNIX, start the responder by running the responder executable image directly in a process.

1.9. Using Presentation Addresses

A presentation address (p-address) specifies service access points (SAPs) for the service providers of all the upper layers to be accessed. For FTAM product, a p-address always contains presentation, session, and transport selectors. It also must have an NSAP. This information is contained in the `osif_local_p_addrs` and `osif_peer_p_addrs` parameters of the `osifpb`. The `osif_ae_entry` structure is used for these parameters.

The field `p_address` of the `osif_ae_entry` structure is used for the upper layer SAPs. The SAPs should be specified in the following format:

```
psap.ssap.tsap.
```

Field descriptions:

psap	is the presentation service access point. Its value can be any string.
ssap	is the session service access point. Its value can be any string.
tsap	is the transport service access point.

The p-address value can include character strings or octet strings. Octet strings must be preceded by %x (or %X). Each selector is terminated on its right by a delimiter (.). If a particular selector is not required, the delimiter (.) must still be included.

For example, if the SSAP is not required, then the format of the p-address might resemble the following:

```
PSAP..TSAP.
```

For the FTAM API Version 3.0, the `nsap_queue` field of the `osif_ae_entry` structure contains the NSAP, provider, and template information, where:

nsap	is the network service access point. For a remote or local system, you must ask the system manager of the network manager for this information.
provider	is the transport type in use.
template	defines the transport template in use.

For the FTAM API Version 3.2, the NSAP and provider information is contained in a linked list pointed to by the `nsap_queue_ptr` field of the `osif_ae_entry` structure. The template information is contained in a linked list pointed to by the `template_queue_ptr` field of the `osif_ae_entry` structure. Chapter 3 provides more details about these data structures.

1.9.1. Matching NSAPs and Templates

FTAM API Version 3.2 provides the ability to disassociate specific NSAPs from specific templates. That is, there is no one-to-one correlation between NSAP and template as there is in Version 3.0. The user provides a list of potential NSAPs, along with the type of network service that each NSAP is expected to use, and a list of potential transport templates.

The OSAK constants OSAK_C_CLNS, OSAK_C_CONS or OSAK_C_RFC1006 are used to indicate whether the NSAP is appropriate for a CLNS, CONS or RFC 1006 network service. Note that OSAK_C_RFC1006 is only valid for use on DECnet-Plus for UNIX or DECnet-Plus for OpenVMS Version 6.0 or later.

When the FTAM API passes the NSAP and template lists to OSAK, OSAK attempts to establish an association with each appropriate NSAP/template pair.

For example, suppose two NSAPs and two templates are passed:

NSAP List	**Template List**
%x21 (CLNS)	OSIT$LOOP_CONS
%x22 (CONS)	OSIT$LOOP_CLNS

OSAK matches the first template in the list with an appropriate NSAP (in this case, the second NSAP in the list), and constructs a final address to attempt an association. Using our example, the address looks something like:

```
OSIT$LOOP_CONS%x22
```

If the association attempt fails with this particular template/NSAP pair, OSAK continues searching the NSAP list looking for another NSAP appropriate for a CONS connection. Once OSAK attempts all possible combinations within the NSAP list for the first template, OSAK attempts an association with the next template in the template list, repeating the template/NSAP pairing operation until an association is established, or until all valid combinations of template/NSAPs have been attempted.

1.10. FTAM API Version 3.0 Applications With Version 3.2

As discussed in Section 1.9, with FTAM API Version 3.2, the `osif_ae_entry`structure is enhanced to provide additional addressing capabilities. In Version 3.2, the `osif_ae_entry` structure contains two additional fields (see Section 4.9).

This change requires that FTAM API applications written to earlier versions of the API be recompiled and relinked. However, no code changes are required unless you want to use the new addressing capabilities.

The FTAM API checks the `nsap_queue[0].nsap.length`field of the `osif_ae_entry` structure to determine which API format is in use. If the value of length is non-zero, the API determines that the FTAM Version 3.0 format is being used and ignores the new fields in the `osif_ae_entry`

structure. If the value of length is zero, the API determines that the Version 3.2 format is being used. In this case, the API ignores the `nsap_queue` array and instead looks for information in the new fields.

1.11. FTAM API Restrictions

The following list describes known restrictions.

- This manual describes FTAM parameters that are part of attribute groups not supported in the FTAM API code. Parameters for unsupported attribute groups should not be used when programming with the FTAM API. For example, the use of abstract-syntax names and constraint set names causes unknown results and should not be used.
- The `osif_protocol_error` vector and the `osif_prot_error_count` variable are not filled in if OSIF_PROTOCOL_ERROR is returned by any function call. OSIF_PROTOCOL_ERROR is used to signal that an error has occurred at a lower layer. The `osif_protocol_error` vector is used to list all the errors that have occurred in the lower layers.
- The FTAM API only supports a buffer list with one buffer. In other words, one P_DATA must be contained in one user buffer. The size of user buffers passed to the FTAM API must be at least 8K bytes. If the buffer is less than 8K, then the user receives the OSIF_NOBUFFS error for the `osif_get_event` function call.
- If a contents type list is not specified in the F-INITIALIZE-request primitive, the FTAM API sends all the supported document types.
- The checkpoint window parameter defaults to one even though the recovery functional unit is not supported.
- An error should be returned by the service provider when a universal class number is specified with FTAM-3 files on F-OPEN and F-CREATE requests. Currently, the universal class number information is ignored and no error is returned.
- The `osif_fadu_locking` parameter of the F-OPEN-request primitive is specified in the documentation and the `osif.h` file, but it is not used by the FTAM API.

Chapter 2. Building and Running API Programs

The programs that you have written for the FTAM application programming interface (FTAM API) can be built on different operating systems. The following sections detail the differences you need to consider for the supported operating systems. For examples of the items discussed, see the FTAM API example files described in Section 2.5.

2.1. Compiling Programs with DEC C on OpenVMS VAX

The FTAM API object library on OpenVMS VAX is built with VAX C. If you are compiling your FTAM API application with DEC C, you must specify certain qualifiers on the CC command, as follows:

```
$ CC/EXTERN_MODEL=COMMON_BLOCK/SHARE_GLOBALS example.c
```

example.c is the name of your program that uses the FTAM API.

2.2. Linking Programs on OpenVMS

To link programs using the OpenVMS operating system, use the following command:

```
$ LINK example.obj, API.OPT/OPTION
```

example.obj is the name of your program object file. API.OPT is a file containing the following lines for OpenVMS VAX:

```
sys$library:osif$fmsg_ptr.obj, -
sys$library:osif$api.olb/lib, -
sys$library:osif$asn1code.olb/lib, -
sys$share:osak$osakshr.exe/share, -
sys$share:osak$prv.exe/share, -
sys$share:vaxcrtl.exe/share
```

Note that even if you are using DEC C instead of VAX C on your VAX system, you must link with VAXCRTL.EXE. For OpenVMS Alpha, API.OPT is the same except do not link with VAXCRTL.EXE (DECC$SHR.EXE is pulled in automatically for you).

2.3. Running Programs on OpenVMS

The following items must be set up properly in order for FTAM API programs to run on OpenVMS.

- FTAM must be able to find the contents type database. This database is contained in the file SYS$LIBRARY:OSIF$OIDS.TXT. You may either create the logical name FTAMOIDS that points to the database, or copy the database to the file FTAMOIDS in the directory where the API application is run.
- OSAK requires the following privileges to be turned on in the process that is running the FTAM API program:

  ```
  NETMBX,TMPMBX,SYSNAM,SYSLCK,PRMMBX
  ```

2.4. Compiling and Linking Programs on UNIX

FTAM on UNIX Version 3.0 and later ships with a sharable library (the FTAM API on previous versions shipped a static library for the API). The following example shows how to build with the sharable library. See also the makefile for the FTAM API example files for an example of how to build.

To compile and link C programs using the UNIX operating system Version 3.0 and later, use the following command:

```
cc example.c -lftam -o example
```

where

`cc`	is the command used to compile and link your program.
`example.c`	is the name of your program that uses the FTAM API.
`-lftam`	specifies the FTAM sharable library.
`-o example`	specifies the name of the executable file that is created.

2.5. FTAM API Example Files

Sample programs are provided written in C to demonstrate how to use the FTAM API calls. Build files are provided to demonstrate how to compile and link programs that use the FTAM API. These example programs create, rename, and delete a file on the system running the example responder. The example programs are located in the following files:

On OpenVMS:

sys$examples:osif_api_exam.c	initiator side
sys$examples:osif_api_resp.c	responder side
sys$examples:osif_api_bld.com	build command procedure
sys$examples:osif_api_bld.opt	linker options file

On UNIX:

/usr/examples/ftamapi/ftam_api_example.c	initiator side
/usr/examples/ftamapi/ftam_resp.c	responder side
/usr/examples/ftamapi/Makefile	makefile

On OpenVMS, the address information supplied in the example programs work without modification if the example initiator and example responder are run on the same OpenVMS system.

On UNIX, most of the address information supplied in the example programs work without modification when running on the same system. However, you must set the following variable in the initiator to be the NSAP of the system:

```
REMOTE_NSAP     NSAP of system running responder
```

If you wish to run the initiator and responder on different systems, see comments in the example programs themselves for instructions.

To compile and link the example program, use the provided build files as follows.

On OpenVMS:

```
$ set default sys$common:[syshlp.examples]$ @osif_api_bld.com
```

On UNIX:

```
# cd /usr/examples/ftamapi# make
```

To run the example programs, first run the example responder in one process, then run the example initiator in a second process. Note that the example responder does not use the OSAK server on OpenVMS or the `ftam_listener` on UNIX. The example responder executable is an active application (see Section 1.8 for a discussion of passive versus active responders).

Chapter 3. FTAM File Services

This chapter describes FTAM services and the sequences in which FTAM services can or must occur. It also describes the parameters used by the services. For additional details, refer to the FTAM standard (ISO 8571-3 and ISO 8571-4).

3.1. Service Sequences

In using the FTAM API, you must be aware of service sequences. These sequences are closely tied to the FTAM regimes. The following list summarizes the sequences and functions of FTAM phases. It also shows the correlations between different phases. For example, Phase 1 is associated with Phase 8 because establishing and ending an association are two activities that are closely related.

- Phase 1: Establishing an association
    - Phase 2: Selecting or creating a file
        - Phase 3: Opening a file
            - Phase 4: Locating a FADU
            - Phase 5: Transferring FADUs and erasing a FADU
        - Phase 6: Closing a file
    - Phase 7: Releasing a file
- Phase 8: Ending an association

Each of these phases provides specific services as follows.

Phase 1:	F-INITIALIZE
Phase 2:	F-CREATE, F-SELECT, F-READ-ATTRIBUTE, F-CHANGE-ATTRIBUTE
Phase 3:	F-OPEN
Phase 4:	F-LOCATE
Phase 5:	F-READ, F-WRITE, F-DATA, F-DATA-END, F-TRANSFER-END, F-CANCEL, F-ERASE
Phase 6:	F-CLOSE
Phase 7:	F-DELETE, F-DESELECT
Phase 8:	F-TERMINATE, F-U-ABORT

Grouping is a convenience that allows you to combine several services into a single communications unit. Grouping functions can occur at many points within an association. Note that the service specifications used here represent the full series of service primitives associated with each service.

When using grouping, you must understand that service classes introduce restrictions. Service classes are defined as combinations of functional units. As a result, grouping can occur only in certain sequences as outlined in the FTAM standard (ISO 8571-3, Annex E). Matching F-BEGIN-GROUP and F-END-GROUP services must occur within the same regime.

The following sections show some sample service sequences that you might use when performing various operations.

3.1.1. Creating a New File and Writing Data

F-INITIALIZE			
	F-CREATE		
		F-OPEN	
			F-LOCATE
			F-WRITE
			F-DATA
			F-DATA-END
			F-TRANSFER-END
		F-CLOSE	
	F-DESELECT		
F-TERMINATE			

3.1.2. Deleting a File Using Grouping

F-INITIALIZE	
	F-BEGIN-GROUP
	F-SELECT
	F-DELETE
	F-END-GROUP
F-TERMINATE	

3.1.3. Reading and Changing Attributes

F-INITIALIZE	
	F-BEGIN-GROUP
	F-SELECT
	F-READ-ATTRIBUTE
	F-CHANGE-ATTRIBUTE
	F-DESELECT
	F-END-GROUP
F-TERMINATE	

3.1.4. Performing a Series of Writes and Reads

F-INITIALIZE				
	F-BEGIN-GROUP			
		F-SELECT		
			F-OPEN	
	F-END-GROUP			
				F-LOCATE
				F-WRITE

				F-DATA
				F-DATA-END
				F-TRANSFER-END
				F-READ
				F-DATA
				F-DATA-END
				F-TRANSFER-END
	F-BEGIN-GROUP			
			F-CLOSE	
		F-DESELECT		
	F-END-GROUP			
F-TERMINATE				

3.1.5. Transferring a File to a Peer System

F-INITIALIZE				
	F-BEGIN-GROUP			
		F-CREATE		
			F-OPEN	
	F-END-GROUP			
				F-LOCATE
				F-WRITE
				F-DATA
				F-DATA-END
				F-TRANSFER-END
	F-BEGIN-GROUP			
			F-CLOSE	
		F-DESELECT		
	F-END-GROUP			
F-TERMINATE				

3.1.6. Canceling a Data Transfer

In this service sequence, the F-CANCEL can be issued during the data transfer phase, in place of F-DATA or F-DATA-END.

F-INITIALIZE					
	F-BEGIN-GROUP				
		F-SELECT			
			F-OPEN		
	F-END-GROUP				
				F-LOCATE	

				F-READ or F-WRITE	
				F-DATA	| F-CANCEL
				F-DATA-END	| F-CANCEL
				F-TRANSFER-END	
	F-BEGIN-GROUP				
			F-CLOSE		
		F-DESELECT			
	F-END-GROUP				
F-U-ABORT					

3.2. FTAM File Services and Parameters

When programming with the FTAM application programming interface (FTAM API), you must be aware of how the FTAM standard is implemented. The FTAM primitives are used in various service sequences to provide the FTAM file services. Each primitive has associated parameters that map to specific descriptors in the `osifpb` parameter block. These relationships are shown in Table 3.1.

Table 3.1. FTAM Primitives and Corresponding Parameters

FTAM Primitive	Parameters	Descriptors
F-BEGIN-GROUP	Threshold	osif_threshold [1]
F-CANCEL	Action Result	osif_action_result
	Diagnostic	osif_diagnostic
F-CHANGE-ATTRIBUTE	Action Result	osif_action_result [2]
	Attributes	osif_filename osif_storage_account osif_file_availability osif_future_filesize osif_access_control osif_legal_qualification
	Diagnostic	osif_diagnostic [2]
F-CLOSE	Action Result	osif_action_result
	Diagnostic	osif_diagnostic
F-CREATE	State Result	osif_state_result [2]
	Action Result	osif_action_result [2]
	Override	osif_override [1]
	Initial Attributes	osif_filename osif_permitted_actions osif_contents_type

FTAM Primitive	Parameters	Descriptors
		osif_storage_account osif_file_availability osif_future_filesize osif_access_control osif_legal_qualification
	Create Password	osif_create_password [1]
	Requested Access	osif_requested_access [1]
	Access Passwords	osif_access_passwords [1]
	Concurrency Control	osif_concurrency_control [1]
	Account	osif_account [1]
	Diagnostic	osif_diagnostic [2]
F-DATA		osif_userdata
F-DATA-END	Action Result	osif_action_result [1]
	Diagnostic	osif_diagnostic [1]
F-DELETE	Action Result	osif_action_result [2]
	Charging	osif_charging [2]
	Diagnostic	osif_diagnostic [2]
F-DESELECT	Action Result	osif_action_result [2]
	Charging	osif_charging [2]
	Diagnostic	osif_diagnostic [2]
F-END-GROUP	—	—
F-ERASE	Action Result	osif_action_result [2]
	FADU Identity	osif_fadu [1]
	Diagnostic	osif_diagnostic [2]
F-INITIALIZE	State Result	osif_state_result [2]
	Action Result	osif_action_result [2]
	Protocol Version	osif_protocol_id
	Implementation Information	osif_implementation_information
	Presentation Context Management	osif_pres_ctx_mgmt
	Service Class	osif_service_class
	Functional Units	osif_functional_units
	Attribute Groups	osif_attribute_groups
	FTAM Quality of Service	osif_ftam_qual_service
	Contents Type List	osif_contents_type_list
	Initiator Identity	osif_initiator_identity [1]
	Account	osif_account [1]

FTAM Primitive	Parameters	Descriptors
	Filestore Password	osif_filestore_password [1]
	Diagnostic	osif_diagnostic [2]
	Checkpoint Window	osif_checkpoint_window
	Calling Presentation Address and Application Title	osif_local_p_addrs
	Called Presentation Address and Application Title	osif_peer_p_addrs
F-LOCATE	Action Result	osif_action_result [2]
	FADU Identity	osif_fadu
	FADU Lock	osif_fadu_lock [1]
	Diagnostic	osif_diagnostic [2]
F-OPEN	State Result	osif_state_result [2]
	Action Result	osif_action_result [2]
	Processing Mode	osif_processing_mode [1]
	Contents Type	osif_contents_type
	Concurrency Control	osif_concurrency_control
	Enable FADU Locking	osif_fadu_locking [1]
	Diagnostic	osif_diagnostic [2]
	Activity Identifier	osif_activity_ident [1]
	Recovery Mode	osif_recovery_mode
F-P-ABORT	Action Result	osif_action_result [1]
	Diagnostic	osif_diagnostic [1]
F-READ	FADU Identity	osif_fadu [1]
	Access Context	osif_access_context [1]
	FADU Lock	osif_fadu_lock [1]
F-READ-ATTRIBUTE	Action Result	osif_action_result [2]
	Attribute Names	osif_attribute_names [1]
	Attributes [2]	osif_filename osif_permitted_actions osif_contents_type osif_storage_account osif_date_time_creation osif_date_time_last_attmod osif_date_time_last_read osif_date_time_last_modif osif_identity_creator

FTAM Primitive	Parameters	Descriptors
		osif_identity_last_modify osif_identity_last_reader osif_identity_last_attmod osif_file_availability osif_filesize osif_future_filesize osif_access_control osif_legal_qualification
	Diagnostic	osif_diagnostic [2]
F-SELECT	State Result	osif_state_result [2]
	Action Result	osif_action_result [2]
	Attributes	osif_filename
	Requested Access	osif_requested_access [1]
	Access Passwords	osif_access_passwords [1]
	Concurrency Control	osif_concurrency_control [1]
	Account	osif_account [1]
	Diagnostic	osif_diagnostic [2]
F-TERMINATE	Charging	osif_charging [2]
F-TRANSFER-END	Action Result	osif_action_result [2]
	Diagnostic	osif_diagnostic [2]
F-U-ABORT	Action Result	osif_action_result [1]
	Diagnostic	osif_diagnostic [1]
F-WRITE	FADU Operation	osif_fadu_operation [1]
	FADU Identity	osif_fadu [1]
	FADU Lock	osif_fadu_lock [1]

[1]Used for request primitives only.

[2]Used for response primitives only.

3.3. Parameter Block Description

The API provides all the service primitives offered by ISO 8571 through the use of function calls and data structures. The function calls allow services to be performed, and the data structures provide a way for entering and receiving FTAM parameters from the FTAM protocol machine. The information provided by the parameter block is referenced by the `osif_send` and `osif_get_event` calls described in Chapter 5.

The FTAM parameter block (`osifpb`) describes the FTAM protocol data unit transmitted between peer entities. All of the parameters used by the FTAM primitives, the application-wide types, and the file attribute types are associated with a particular descriptor that has a particular format. Section 3.4 provides more details about the parameters and descriptors.

Parameters are assigned to the `osifpb` parameter block through simple assignment statements. The address or value of the parameters must be stored in the address or value field of the descriptor in `osifpb`, depending on whether it is a string or an integer. If the value is a bit string, the value field of the descriptor is filled in directly. The length of the parameters must also be assigned. If the length is not assigned, the parameter is ignored.

If your parameter requires a choice of values, there is a type field that must be used to specify the type of data. For example, if a parameter can be encoded as an octet string or a graphic string, the type field must reflect the type. The type field is also used to specify a Null type parameter. The possible values for the type field and their ASN.1 universal codes are:

OSIF_UC_BOOLEAN	BOOLEAN
OSIF_UC_INTEGER	INTEGER
OSIF_UC_BIT	BIT STRING
OSIF_UC_OCTET	OCTET STRING
OSIF_UC_NULL	NULL
OSIF_UC_OBJECT_ID	OBJECT IDENTIFIER
OSIF_UC_OBJECT_DSC	ObjectDescriptor
OSIF_UC_EXTERNAL	EXTERNAL
OSIF_UC_REAL	REAL
OSIF_UC_ENUMERATED	ENUMERATED
OSIF_UC_NUMERIC	NumericString
OSIF_UC_PRINTABLE	PrintableString
OSIF_UC_T61	T61String
OSIF_UC_VIDEOTEXT	VideotexString
OSIF_UC_IA5	IA5String
OSIF_UC_UTCTIME	UTCTime
OSIF_UC_GENERALTIME	GeneralizedTime
OSIF_UC_GRAPHIC	GraphicString
OSIF_UC_VISIBLE	VisibleString
OSIF_UC_GENERAL	GeneralString

If the parameter has a default and the length is zero, the default is applied by the API. If the length is a non-zero value, then the default is not applied.

Chapter 4 provides more details on the FTAM parameter block data structure (`osifpb`) and the other data structures used by the API to pass parameter information.

3.4. Parameter Description

This section describes the parameters of `osifpb` and their possible values. The parameters are listed alphabetically. For more details about their usage, refer to the FTAM standard (ISO 8571-3 and ISO 8571-4).

osif_access_context (Access Context)

Specifies the file access structure for read operations. See Section 4.18 for more information.

osif_access_control (Access Control attribute)

Defines conditions under which file access is valid. This value is set at file creation, but it can be altered by the change attribute action. A condition consists of one or two terms stating the type of access allowed (an action list term or a concurrency access term), together with a set of zero to three terms testing for matching attribute values (initiator identity, access passwords, or calling AE-title). See Section 4.15 for more information.

osif_access_passwords (Access Passwords)

Provides passwords for the actions specified in the requested access parameter. This parameter is available only if the security attribute group has been negotiated. See Section 4.16 for more information.

osif_account (Account)

Identifies the account to charge for the cost of a regime establishment. It is used to set the current account activity attribute. If this parameter is not specified, the activity attribute is unset or retains its previous value. The current account activity attribute reverts to its previous value at the end of a regime.

osif_action_result (Action Result)

Passes on summarized information that is available in the diagnostic parameter. It has the following possible values.

OSIF_SR_SUCCESS	success, the default value
OSIF_TRANSIENT_ERROR	transient-error
OSIF_PERMANENT_ERROR	permanent-error

osif_activity_ident (Activity Identifier)

Used only when the recovery functional unit has been negotiated on F-INITIALIZE. Its value (an integer) is used in reestablishing the data-transfer regime after a failure.

osif_application_context (Application Context Name)

Represents the properties of the association. The initiator proposes a name that the responder may accept and return or the responder may return a different name. The application context name returned by the responder is used for the established association.

osif_attribute_groups (Attribute Groups)

Negotiates the set of optional file attribute groups available for the association. The default value is null (empty). The following values are also possible.

OSIF_ATG_STORAGE	storage
OSIF_ATG_SECURITY	security
OSIF_ATG_PRIVATE	private

osif_attribute_names (Attribute Names)

Indicates which file attributes from the kernel or negotiated attribute groups are read. The possible groups are the kernel group, the storage group, the security group, and the private group.

The kernel group has the following possible values for file attributes.

OSIF_ATT_FILENAME	file name
OSIF_ATT_PERMITTED_ACTIONS	permitted actions
OSIF_ATT_CONTENTS_TYPE	contents type

The storage group has the following possible values for file attributes.

OSIF_ATT_STORAGE_ACCOUNT	storage account
OSIF_ATT_CREATION_TIME	date and time of creation
OSIF_ATT_MODIFICATION_TIME	date and time of last modification
OSIF_ATT_READ_TIME	date and time of last read access
OSIF_ATT_ATTRIBUTE_MODIFICATION_TIME	date and time of last attribute modification
OSIF_ATT_CREATOR_ID	identity of creator
OSIF_ATT_MODIFIER_ID	identity of last modifier
OSIF_ATT_READER_ID	identity of last reader
OSIF_ATT_ATTRIBUTE_MODIFIER_ID	identity of last attribute modifier
OSIF_ATT_FILE_AVAILABILITY	file availability
OSIF_ATT_FILESIZE	file size
OSIF_ATT_FUTURE_FILESIZE	future file size

The security group has the following possible values for file attributes.

OSIF_ATT_ACCESS_CONTROL	access control
OSIF_ATT_LEGAL_QUALIFICATIONS	legal qualifications

The private group has the following possible value for file attributes.

OSIF_ATT_PRIVATE_USE	private use

osif_block_size

Passes the size of the `osifpb` parameter block.

osif_block_type

Passes the function code for the FTAM primitive. See Section 4.1 for more information.

osif_character_sets

Describes the character sets used in the file if they are different from the default, ISO 646.

osif_charging (Charging)

Passes cost information attributed to the account during the regime being released. This parameter can be used only if the account parameter was specified at the beginning of the regime. See Section 4.19 for more information.

osif_checkpoint_window (Checkpoint Window)

Used only when the recovery functional unit has been negotiated, this parameter indicates the maximum number of checkpoints that may remain unacknowledged. This integer value is inserted only by the sender and is used for F-INITIALIZE-request and F-INITIALIZE-response.

osif_concurrency_control (Concurrency Control)

Defines the possible actions on a file and their respective access locks during a file-select or file-open regime. See Section 4.14 for more information.

osif_contents_type (Contents Type attribute)

Identifies the abstract data type of the file contents. Its value is either a document type with optional parameters or an abstract syntax and a constraint set name. See Section 4.7 for more information.

osif_contents_type_list (Contents Type List)

Lists the document types and abstract syntaxes and allows the negotiation of presentation context when establishing the FTAM regime. This parameter is mandatory in certain classes if the presentation context management functional unit is not being negotiated. See Section 4.6 for more information.

osif_create_password (Create Password)

Describes the access control parameter `create-password` used by F-CREATE-request as a character or octet string.

osif_date_time_creation (Date and Time of Creation attribute)

Indicates when the file was created in GeneralizedTime. It is set by the responder when the file is created and refers to the local date and time of the responder. If this parameter is not supported, set the type field of the descriptor to OSIF_UC_NULL. It cannot be altered by the change attribute action.

osif_date_time_last_attmod (Date and Time of Last Attribute Modification attribute)

Indicates when a file attribute value was last modified in GeneralizedTime. If this parameter is not supported, set the type field of the descriptor to OSIF_UC_NULL. It is altered by the responder whenever the change attribute action is successfully performed on one or more attributes. This attribute is not modified by an implicit change to an attribute and it cannot be altered by the change attribute action.

osif_date_time_last_modif (Date and Time of Last Modification attribute)

Indicates when the file contents were last modified in GeneralizedTime. If this parameter is not supported, set the type field of the descriptor to OSIF_UC_NULL. It is altered by the responder whenever the file has been opened for modification or extension and is closed. This attribute is not altered unless the file is opened to allow change of the contents. It is not altered when the file attributes are changed.

osif_date_time_last_read (Date and Time of Last Read Access attribute)

Indicates when the file contents were last read in GeneralizedTime. If this parameter is not supported, set the type field of the descriptor to OSIF_UC_NULL. It is altered by the responder whenever the file has been opened for reading and is closed. This attribute is not altered unless the file is opened and it cannot be altered by the change attribute action.

osif_diagnostic (Diagnostic)

Provides more details about the information given in the action result parameter for a successful action, a transient error, or a permanent error. See Section 4.5 for more information.

osif_fadu (FADU Identity)

Specifies the target FADU to be used for file operations. The value of this parameter depends on the file operation. See Section 4.17 for more information.

osif_fadu_lock (FADU Lock)

Sets individual FADU locks on or off. If this parameter is not specified, the locks remain unchanged. Setting the locks ON changes the value from "not required" to "no access" and from "shared" to "exclusive" until the lock is set OFF, the FADU is erased, or the file is closed. Setting the lock OFF causes the lock to change back to its original value. The values for setting the locks ON and OFF are:

OSIF_FADU_LOCK_OFF	off
OSIF_FADU_LOCK_ON	on

osif_fadu_locking (Enable FADU Locking)

Indicates whether locking is on a per-FADU basis or on a file basis as a Boolean value. This parameter is available only if the storage attribute group has been negotiated and the concurrency control parameter is present.

osif_fadu_operation (FADU Operation)

Indicates the action to be taken by the filestore provider on receiving transferred data.

osif_file_availability (File Availability attribute)

Indicates the availability of the file. This parameter appears only in response PDUs and has the following possible values:

no-value-available	indicates that no value is available for this attribute by setting the type field to OSIF_UC_NULL.
actual-values	indicates when a file is available as follows: OSIF_IMMEDIATE_AVAILABILITY — immediate availability OSIF_DEFERRED_AVAILABILITY — deferred availability

osif_filename (File name attribute)

Describes a list of the file name parameters by providing a pointer to the `osif_fn` structure. This attribute is set atfile creation, but can be altered by the change attribute action. See Section 4.4 for more information.

osif_filesize (File size attribute)

Indicates the size of the file. This parameter appears only in response PDUs and has the following possible values:

no-value-available	indicates that no value is available for this attribute by setting the type field to OSIF_UC_NULL

actual-values	indicates the size of a file

osif_filestore_password (Filestore Password)

Used by the responder to authenticate the initiator identity parameter. It is a character or octet string.

osif_ftam_qual_service (FTAM Quality of Service)

Indicates the susceptibility of the external file service user to errors. This parameter has the following possible values:

OSIF_FQOS_NO_RECOVERY	no-recovery, not susceptible to errors and no error recovery provided
OSIF_FQOS_CLASS_1_RECOVERY	class-1-recovery, susceptible to errors that damage the data-transfer regime
OSIF_FQOS_CLASS_2_RECOVERY	class-2-recovery, susceptible to errors that damage the open or data-transfer regimes
OSIF_FQOS_CLASS_3_RECOVERY	class-3-recovery, susceptible to errors that damage the select, open, or data-transfer regimes, or that disconnect the association

osif_functional_units (Functional Units)

Negotiates the file service functional units (except the kernel) available from the negotiated service class for the association. This parameter has the following values:

OSIF_FU_READ	read
OSIF_FU_WRITE	write
OSIF_FU_FILE_ACCESS	file-access
OSIF_FU_LIMIT_FILE_MGMT	limited-file-management
OSIF_FU_ENH_FILE_MGMT	enhanced-file-management
OSIF_FU_GROUPING	grouping
OSIF_FU_FADU_LOCKING	FADU-locking
OSIF_FU_RECOVERY	recovery
OSIF_FU_RESTART_DATA_XFR	restart-data-transfer

osif_future_filesize (Future File size attribute)

Indicates the size in octets to which a file may grow due to modification and extension. This integer value is set at file creation, but it can be altered by the change attribute action.

osif_identity_creator (Identity of Creator attribute)

Indicates the value of the current initiator identity activity attribute at file creation as a GraphicString. This attribute cannot be altered by using the change attribute action.

osif_identity_last_attmod (Identity of Last Attribute Modifier attribute)

Indicates the value of the current initiator identity as a GraphicString whenever the change attribute action is successfully performed on one or more attributes. This attribute cannot be changed using the change attribute action.

osif_identity_last_modify (Identity of Last Modifier attribute)

Indicates the value of the current initiator identity activity attribute as a GraphicString whenever the file has been opened for modification or extension and is closed. This attribute cannot be altered by using the change attribute action.

osif_identity_last_reader (Identity of Last Reader attribute)

Indicates the value of the current initiator identity activity attribute as a GraphicString whenever the file has been opened for reading and is closed. This attribute cannot be altered using the change attribute action.

osif_implementation_information (Implementation Information)

Needed only if you want to distinguish versions of implementations on different equipment.

osif_initiator_identity (Initiator Identity)

Identifies the calling user as a GraphicString.

osif_legal_qualification (Legal Qualification attribute)

Indicates if the legal qualification for the security attribute group is available. This parameter appears only in response PDUs and has the following possible values:

no-value-available	indicates that no value is available for this attribute by setting the type field to OSIF_UC_NULL.
actual-values	indicates a value for the legal qualification attribute.

osif_local_p_addrs (Local Address (host system))

The structure containing the application entity addresses (AP-title and AE-qualifier), presentation selector, session selector, transport selector, and up to five network service access points and transport options (template and provider).

osif_peer_p_addrs (Target Address (system accepting the connection))

The structure containing the application entity addresses (AP-title and AE-qualifier), presentation selector, session selector, transport selector, and up to five network service access points and transport options (template and provider).

osif_override (Override)

Defines the action to take if the named file already exists according to one of the following values:

OSIF_OVR_CREATE_FAILURE	create-failure
OSIF_OVR_SELECT_OLD_FILE	select-old-file
OSIF_OVR_DEL_CRE_OLD_ATTRIB	delete-and-create-with-old-attributes
OSIF_OVR_DEL_CRE_NEW_ATTRIB	delete-and-create-with-new-attributes

osif_permitted_actions (Permitted Actions attribute)

Optional parameter that indicates the available actions and FADU identity groups with the following values:

OSIF_PA_READ	read
OSIF_PA_INSERT	insert
OSIF_PA_REPLACE	replace
OSIF_PA_EXTEND	extend
OSIF_PA_ERASE	erase
OSIF_PA_READ_ATTRIBUTE	read-attribute
OSIF_PA_CHANGE_ATTRIBUTE	change-attribute
OSIF_PA_DELETE_FILE	delete-file
OSIF_PA_TRAVERSAL	traversal
OSIF_PA_REVERSE_TRAVERSAL	reverse-traversal
OSIF_PA_RANDOM_ORDER	random-order

osif_pres_ctx_mgmt (Presentation Context Management)

Indicates whether the context management functional unit is used during the FTAM open and recovery procedures as a Boolean value.

osif_processing_mode (Processing Mode)

Establishes a subset of the valid actions negotiated in the select regime for use within the open regime being established. It indicates the valid actions performed as a result of access control and bulk data transfer requests and determines the filestore actions that the responding entity can perform. The possible values are:

OSIF_PM_READ	read
OSIF_PM_INSERT	insert
OSIF_PM_REPLACE	replace
OSIF_PM_EXTEND	extend
OSIF_PM_ERASE	erase

osif_prot_error_count (Protocol Error Count)

Indicates the number of returned errors. See Chapter 5 for more information.

osif_protocol_error

Contains a list of layer-specific errors. See Chapter 5 for more information.

osif_protocol_id (Protocol Version)

Indicates the protocol version. This parameter has a default value of version-1.

osif_recovery_mode (Recovery Mode)

Indicates the error recovery facilities available during the current open regime and the points at which data transfer can resume according to one of the following values:

0	none
1	at-start-of-file

2	at-any-active-checkpoint

osif_requested_access (Requested Access)

Indicates the actions performed when a file is selected or recovered according to the following values:

OSIF_AR_READ	read
OSIF_AR_INSERT	insert
OSIF_AR_REPLACE	replace
OSIF_AR_EXTEND	extend
OSIF_AR_ERASE	erase
OSIF_AR_READ_ATTRIBUTE	read-attribute
OSIF_AR_CHANGE_ATTRIBUTE	change-attribute
OSIF_AR_DELETE_FILE	delete-file

osif_returned_buffer

Is a pointer to `osif_buffer_list` structure. Buffers are returned to the user through this value. A null value can be returned. See Section 1.5 for more information.

osif_scratchpad

Used by `osifpb` to manipulate data. See Section 4.1 for more information.

osif_service_class (Service Class)

Indicates the capability of the initiator. This parameter has the following possible values:

OSIF_CLASS_UNCONST	unconstrained-class
OSIF_CLASS_MGMT	management-class
OSIF_CLASS_XFR	transfer-class
OSIF_CLASS_XFR_MGMT	transfer-and-management-class
OSIF_CLASS_ACCESS	access-class

osif_state_result (State Result)

Indicates the result of state changes. This parameter has the following values:

OSIF_SR_SUCCESS	success
OSIF_SR_FAILURE	failure

osif_storage_account (Storage Account attribute)

Identifies the accountable authority responsible for accumulated file storage charges as a GraphicString.

osif_threshold (Threshold)

Specifies the number of primitives within a group that are analyzed without failing before any part of the group can succeed.

osif_userdata

Describes the buffer containing the F-DATA as a string descriptor.

Chapter 4. Data Structures

The data structures described in this chapter allow the FTAM API to enter and receive FTAM parameters from the FTAM protocol machine. You will find that within the structures, ASN.1 sequences and sets are implemented as linked lists.

4.1. FTAM Parameter Block

The `osifpb` structure is the main data structure of the FTAM API. This structure contains a parameter for each parameter in any of the FTAM primitives. The FTAM primitives are distinguished by the `osif_block_type` parameter. The set of valid block types are:

Function Code	**FTAM Service Primitive**
OSIF_PBDEF_P_ABORT	F-P-ABORT
OSIF_PBDEF_U_ABORT	F-U-ABORT
OSIF_PBDEF_BG_REQ	F-BEGIN-GROUP-request, F-BEGIN-GROUP-indication
OSIF_PBDEF_BG_RSP	F-BEGIN-GROUP-response, F-BEGIN-GROUP-confirm
OSIF_PBDEF_CHAT_REQ	F-CHANGE-ATTRIBUTES-request, F-CHANGE-ATTRIBUTES-indication
OSIF_PBDEF_CHAT_RSP	F-CHANGE-ATTRIBUTES-response, F-CHANGE-ATTRIBUTES-confirm
OSIF_PBDEF_CRE_REQ	F-CREATE-request, F-CREATE-indication
OSIF_PBDEF_CRE_RSP	F-CREATE-response, F-CREATE-confirm
OSIF_PBDEF_CLOSE_REQ	F-CLOSE-request, F-CLOSE-indication
OSIF_PBDEF_CLOSE_RSP	F-CLOSE-response, F-CLOSE-confirm
OSIF_PBDEF_DATA_REQ	F-DATA-request, F-DATA-indication
OSIF_PBDEF_DATA_END_REQ	F-DATA-END-request, F-DATA-END-indication
OSIF_PBDEF_DELETE_REQ	F-DELETE-request, F-DELETE-indication
OSIF_PBDEF_DELETE_RSP	F-DELETE-response, F-DELETE-confirm
OSIF_PBDEF_DESELECT_REQ	F-DESELECT-request, F-DESELECT-indication
OSIF_PBDEF_DESELECT_RSP	F-DESELECT-response, F-DESELECT-confirm
OSIF_PBDEF_EG_REQ	F-END-GROUP-request, F-END-GROUP-indication
OSIF_PBDEF_INIT_REQ	F-INITIALIZE-request, F-INITIALIZE-indication
OSIF_PBDEF_INIT_RSP	F-INITIALIZE-response, F-INITIALIZE-confirm
OSIF_PBDEF_NODE_DE_REQ	Node descriptor data element
OSIF_PBDEF_OPEN_REQ	F-OPEN-request, F-OPEN-indication
OSIF_PBDEF_OPEN_RSP	F-OPEN-response, F-OPEN-confirm
OSIF_PBDEF_RAT_REQ	F-READ-ATTRIBUTES-request, F-READ-ATTRIBUTES-indication

Function Code	FTAM Service Primitive
OSIF_PBDEF_RAT_RSP	F-READ-ATTRIBUTES-response, F-READ-ATTRIBUTES-confirm
OSIF_PBDEF_READ_REQ	F-READ-request, F-READ-indication
OSIF_PBDEF_SEL_REQ	F-SELECT-request, F-SELECT-indication
OSIF_PBDEF_SEL_RSP	F-SELECT-response, F-SELECT-confirm
OSIF_PBDEF_TERM_REQ	F-TERMINATE-request, F-TERMINATE-indication
OSIF_PBDEF_TERM_RSP	F-TERMINATE-response, F-TERMINATE-confirm
OSIF_PBDEF_TRANSFER_END_REQ	F-TRANSFER-END-request, F-TRANSFER-END-indication
OSIF_PBDEF_TRANSFER_END_RSP	F-TRANSFER-END-response, F-TRANSFER-END-confirm
OSIF_PBDEF_WRITE_REQ	F-WRITE-request, F-WRITE-indication
OSIF_PBDEF_CANCEL_REQ	F-CANCEL-request, F-CANCEL-indication
OSIF_PBDEF_CANCEL_RSP	F-CANCEL-response, F-CANCEL-confirm
OSIF_PBDEF_LOCATE_REQ	F-LOCATE-request, F-LOCATE-indication
OSIF_PBDEF_LOCATE_RSP	F-LOCATE-response, F-LOCATE-confirm
OSIF_PBDEF_ERASE_REQ	F-ERASE-request, F-ERASE-indication
OSIF_PBDEF_ERASE_RSP	F-ERASE-response, F-ERASE-confirm

The use of the parameters is dependent on the FTAM primitive type. Any parameters that are not within the scope of the specific FTAM primitive type are ignored by the FTAM API. The relationship between the FTAM primitives and the parameters is shown in Table 3.1.

The `osifpb` structure has two parts — one for specific parameters and one for variable data. The `osif_scratchpad`descriptor is used to distinguish these parts.

For specific parameters, usage of the scratch pad for request and response primitives is a local issue for the FTAM API user. The allocated data can be passed to the FTAM API locally through static or dynamic means or the data can be referenced by the `osifpb` descriptors and stored in the scratch pad.

For variable data, the usage of the scratch pad for indication and response primitives is different. The scratch pad is filled in by the FTAM API and the underlying FTAM protocol machine. As a result, the `osifpb` structures may point into the buffer supplied by the `osif_give_buffer`call or they may point to the scratch pad. The buffers from the `osif_give_buffer` call are returned to the FTAM API user in the `osif_returned_buffer` descriptor.

The following `osifpb` structure includes all the parameters that can be used by the FTAM API.

```
struct osifpb {

    unsigned int                osif_block_type;  /* block identifier  */
    unsigned int                osif_block_size;  /* size of the block */
    struct osif_buffer_list     *osif_returned_buffer;
    struct osif_prot_err        osif_protocol_error;
    int                         osif_prot_error_count;

    struct osif_faduac          osif_access_context;
```

```
struct osif_access_ctl      osif_access_control;
struct osif_apwd            osif_access_passwords;
struct osif_sdesc           osif_account;
struct osif_bdesc           osif_action_result;
struct osif_bdesc           osif_activity_ident;
struct osif_sdesc           osif_application_context;
struct osif_sdesc           osif_arc_length;
struct osif_bdesc           osif_attribute_groups;
struct osif_bdesc           osif_attribute_names;
struct osif_bdesc           osif_character_sets;
struct osif_charging_pb     *osif_charging;
struct osif_bdesc           osif_checkpoint_window;
struct osif_cc              osif_concurrency_control;
struct osif_ct              osif_contents_type;
struct osif_ctl             *osif_contents_type_list;
struct osif_sdesc           osif_create_password;
struct osif_sdesc           osif_date_time_creation;
struct osif_sdesc           osif_date_time_last_attmod;
struct osif_sdesc           osif_date_time_last_modif;
struct osif_sdesc           osif_date_time_last_read;
struct osif_sdesc           osif_delete_password;
struct osif_access_ctl      osif_delete_values;
struct osif_sdesc           osif_define_context;
struct osif_diagnostics_pb  *osif_diagnostic;
struct osif_sdesc           osif_encryption_name;
struct osif_faduid          osif_fadu;
struct osif_bdesc           osif_fadu_lock;
struct osif_sdesc           osif_fadu_locking;
struct osif_bdesc           osif_fadu_operation;
struct osif_fn              *osif_filename;
struct osif_bdesc           osif_filesize;
struct osif_bdesc           osif_file_availability;
struct osif_sdesc           osif_filestore_password;
struct osif_sdesc           osif_ftam_coded;
struct osif_bdesc           osif_ftam_qual_service;
struct osif_bdesc           osif_functional_units;
struct osif_bdesc           osif_future_filesize;
struct osif_sdesc           osif_initiator_identity;
struct osif_sdesc           osif_identity_creator;
struct osif_sdesc           osif_identity_last_attmod;
struct osif_sdesc           osif_identity_last_modify;
struct osif_sdesc           osif_identity_last_reader;
struct osif_sdesc           osif_implementation_information;
struct osif_access_ctl      osif_insert_values;
struct osif_sdesc           osif_legal_qualification;
struct osif_ae_entry        osif_local_p_addrs;
struct osif_bdesc           osif_override;
struct osif_ae_entry        osif_peer_p_addrs;
struct osif_bdesc           osif_permitted_actions;
struct osif_bdesc           osif_pres_ctx_mgmt;
struct osif_bdesc           osif_processing_mode;
struct osif_sdesc           osif_protocol_id;
struct osif_sdesc           osif_remove_context;
struct osif_bdesc           osif_requested_access;
struct osif_bdesc           osif_recovery_mode;
struct osif_bdesc           osif_service_class;
struct osif_bdesc           osif_session_version;
struct osif_bdesc           osif_state_result;
```

```
    struct osif_sdesc           osif_storage_account;
    struct osif_bdesc           osif_threshold;
    struct osif_sdesc           osif_user_coded;
    struct osif_sdesc           osif_userdata;  /* User data used to */
    int                         osif_reserved; /* Reserved for alignment */
    unsigned char               osif_scratchpad[ SCRATCHPAD_SIZE ];
};
```

More information on the descriptors and their associated parameters is found in Section 3.2 and Section 3.4.

4.2. String Descriptor Specification

String parameters can be either character, bit, or octet strings. The following `osif_sdesc` structure is used to specify the string descriptor.

```
struct osif_sdesc {
unsigned char *address;
unsigned short length;
unsigned char type;
unsigned char class;
};
```

Field descriptions:

address	A pointer to the specified string.
length	The length of the specified string.
type	Defines the type of the string. For example, OSIF_UC_GRAPHIC or OSIF_UC_OCTET.
class	Defines the class of the string. For internal use only; users do not need to specify any value.

4.3. Binary Descriptor Specification

Binary parameters are integers. The following `osif_bdesc` structure is used to specify the binary descriptor.

```
struct osif_bdesc {
unsigned value;
unsigned short length;
unsigned char type;
unsigned char class;
};
```

Field descriptions:

value	The value of the specified integer.
length	The length of the integer in bytes.
type	Defines the type of the integer.
class	Defines the class of the integer. For internal use only; users do not need to specify any value.

4.4. File Names

File names can be specified as a sequence of graphic strings. Most profiles restrict file names to one element. The following `osif_fn`structure is used to pass a file name to the FTAM protocol machine as a Null terminated linked list.

```
struct osif_fn {
struct osif_fn *next;
struct osif_sdesc filename;
};
```

Field descriptions:

next	A pointer to the next filename element.
filename	The `osif_sdesc` string descriptor describing the filename.

4.5. Diagnostics

Diagnostics are returned as part of a response primitive. Diagnostics can be passed as a sequence that is a Null terminated linked list. The `osif_diagnostics_pb` structure follows.

```
struct osif_diagnostics_pb {
struct osif_diagnostics_pb *next
struct osif_bdesc diagnostic_type;
struct osif_bdesc error_identifier;
struct osif_bdesc error_observer;
struct osif_bdesc error_source;
struct osif_bdesc suggested_delay;
struct osif_sdesc further_details;
};
```

Field descriptions:

next	A pointer to the next diagnostic in the sequence.
diagnostic_type	One of the following values describing the type: OSIF_INFORMATIVE_ERROR — informative OSIF_TRANSIENT_ERROR — transient OSIF_PERMANENT_ERROR — permanent
error_identifier	A value describing the error that matches the diagnostic errors found in ISO 8571-3. For your convenience, these values and corresponding information are listed in Appendix B.
error_observer	One of the following values indicating the observer of the error: OSIF_INITIATING_USER — initiating file service user OSIF_INITIATING_FPM — initiating file protocol machine OSIF_RESPONDING_FPM — the responding file protocol machine

	OSIF_RESPONDING_USER — the responding file service user (filestore)
error_source	One of the following values indicating the presumed source of the error. OSIF_NO_CATEGORIZATION — no categorization possible OSIF_SUPPORTING_SERVICE — service supporting the file protocol machines
suggested_delay	The integer describing the suggested delay.
further_details	The character string describing any extra information about the error that the implementation wishes to provide.

4.6. Contents Type Lists

Contents type lists are part of the F-INITIALIZE service primitive. They are Null terminated linked lists of `osif_ctl` structures. They describe the abstract syntaxes supported by FTAM implementations and are used for negotiating the abstract syntaxes between cooperating FTAM providers. The contents type list element contains either a document type name or an abstract syntax name. If both are specified in the same `osif_ctl` structure, then the error OSIF_BAD_CNTTYLST is returned.

```
struct osif_ctl {
struct osif_ctl *next
struct osif_sdesc abstract_syntax_name;
struct osif_sdesc document_name;
};
```

Field descriptions:

next	A pointer to the next contents type list.
abstract_syntax_name	The character string describing the abstract syntax name.
document_name	The character string describing the document type name.

4.7. Contents Type Parameter

The contents type parameter is an optional parameter of the F-CREATE and F-OPEN primitives. The contents type specifies either a document type name with its associated document parameters or an abstract syntax name/constraint set name pair. Do not specify both in the same `osif_ct` structure.

```
struct osif_ct {
struct osif_sdesc abstract_syntax_name;
struct osif_sdesc constraint_set_name;
struct osif_sdesc document_name;
struct osif_dt_subparms document_param;
};
```

Field descriptions:

abstract_syntax_name	The character string describing the abstract syntax name.
constraint_set_name	The character string describing the constraint set name.

document_name	The character string describing the document type name, for example, FTAM-1 or NBS-9.
document_param	The document type parameters for the specified document.

4.8. Document Type Parameters

Document type parameters are specified as part of the contents type structure `osif_ct`. These parameters are associated with each specification of the document type and describe the contents of the document type.

```
struct osif_dt_subparms {
struct osif_bdesc max_string_length;
struct osif_bdesc string_significance;
struct osif_bdesc universal_class;
struct osif_bdesc attribute_names;
struct osif_bdesc max_record_length;
struct osif_bdesc record_significance;
};
```

Field descriptions:

max_string_length	An integer describing the string length.
string_significance	An integer describing the significance of strings.
universal_class	An integer describing the type of strings found in the document.
attribute_names	For specifying an NBS-9 document type.
max_record_length	An integer describing the maximum length of records found in the document.
record_significance	An integer describing the significance of records in the document.

4.9. Application-Entity Entry

The application-entity entry is included as a parameter of the F-Initialize primitive. Some of the fields of this structure are used for FTAM API Version 3.0 and some are used for FTAM API Version 3.2.

```
struct osif_ae_entry {
struct osif_ae_addr         ae_addr;
struct osif_sdesc           p_address;      /* psap.ssap.tsap. */
struct osif_nsap_entry      nsap_queue[OSIF_MAX_NSAPS];
struct osif_nsap_queue      *nsap_queue_ptr;
struct osif_template_queue  *template_queue_ptr;
};
```

Field descriptions:

ae_addr	The structure containing the AP-title and AE-qualifier.
p_address	The field containing a character string specifying the psel.ssel.tsel. address. Field descriptions:

	• psel — is the presentation selector • ssel — is the session selector • tsel — is the transport selector
nsap_queue	The structure containing network selectors and transport options for FTAM API Version 3.0.
nsap_queue_ptr	A pointer to a linked list of structures containing network selectors and transport providers for FTAM API Version 3.2.
template_queue_ptr	A pointer to a linked list of transport template names for FTAM API Version 3.2.

4.10. Application-Entity Address

The applications-entity address consists of both an application-entity qualifier and an application-title:

```
struct osif_ae_addr {
struct osif_sdesc   ae_qualifier
struct osif_sdesc   ap_title;
}:
```

Field descriptions:

ae_qualifier	The character string describing the AE-qualifier.
ap_title	The character string describing the AP-title. If integers are being used, this must be a pointer to an integer.

4.11. Network Selector and Transport Options Queue (Version 3.0 Only)

The network selector and transport options queue is defined as follows:

```
struct osif_nsap_entry {
struct osif_sdesc   nsap;
struct osif_sdesc   template;
struct osif_sdesc   provider;
};
```

Field descriptions:

nsap	The network service access point (NSAP). You can define up to five NSAPs and use multihoming to establish a connection. Each time a connection attempt fails, the initiator uses the next NSAP until either a connection is established or no NSAPs remain.
template	A character string defining which transport template is used. By default, the transport template called "default" is used.
provider	A character string specifying either the transport provider called "OSI" (for OSI transport services), or "RFC1006" (for TCP/IP services). By default, the transport provider called "OSI" is used.

4.12. Network Selector and Transport Provider Queue (Version 3.2 Only)

The network selector and transport provider queue is defined as follows:

```
struct osif_nsap_queue {
struct osif_nsap_queue *next;
struct osif_sdesc      nsap;
struct osif_sdesc      provider;
struct osif_bdesc      network_svc;
};
```

Field descriptions:

next	A pointer to the next NSAP queue entry.
nsap	The network service access point. You can define up to five NSAPs and use multihoming to establish a connection. See Section 1.9.1 for a description of how the NSAPs and transport templates are used.
provider	A character string specifying either the transport provider called "OSI" (for OSI transport services), or "RFC1006" (for TCP/IP services). By default, the transport provider called "OSI" is used.
network_svc	A constant used to indicate whether the NSAP is appropriate for CLNS, CONS, or RFC1006 network service. Valid values are the OSAK constants OSAK_C_CLNS, OSAK_C_CONS, or OSAK_C_RFC1006.

4.13. Transport Template Queue (Version 3.2 Only)

The transport template queue is defined as follows:

```
struct osif_template_queue {
struct osif_template_queue *next;
struct osif_sdesc          template_name;
};
```

Field descriptions:

next	A pointer to the next template queue entry.
template_name	A character string defining which transport template is used. By default, the transport template called"default" is used. See Section 1.9 for a description of how the NSAPs and transport templates are used.

4.14. Concurrency Control

The concurrency control parameter is found in the F-SELECT, F-CREATE, and F-OPEN primitives. It is used by initiators to request locks on actions performed on remote files. The `osif_cc` structure follows.

```
struct osif_cc {
struct osif_bdesc change_attrib_cc;
struct osif_bdesc delete_file_cc;
struct osif_bdesc erase_cc;
struct osif_bdesc extend_cc;
struct osif_bdesc insert_cc;
struct osif_bdesc read_attrib_cc;
struct osif_bdesc read_cc;
struct osif_bdesc replace_cc;
};
```

Field descriptions:

change_attrib_cc	The bit string describing the concurrency key for change attributes.
delete_file_cc	The bit string describing the concurrency key for delete.
erase_cc	The bit string describing the concurrency key for erase.
extend_cc	The bit string describing the concurrency key for extend.
insert_cc	The bit string describing the concurrency key for insert.
read_attrib_cc	The bit string describing the concurrency key for read attributes.
read_cc	The bit string describing the concurrency key for read.
replace_cc	The bit string describing the concurrency key for replace.

Each field above may have one of these values:

OSIF_CC_NOT_REQUIRED	not required
OSIF_CC_SHARED	shared
OSIF_CC_EXCLUSIVE	exclusive
OSIF_CC_NO_ACCESS	no-access

4.15. Access Control

The following `osif_access_ctl` structure allows you to specify the security required for file operations.

```
struct osif_access_ctl {
struct osif_access_ctl *next;
struct osif_sdesc no_value_avail
struct osif_bdesc action_list;
struct osif_cc concurrency_access;
struct osif_sdesc identity;
struct osif_apwd passwords;
struct osif_ae_addr location;
};
```

Field descriptions:

*next	Points to the next access control structure in the list.
action_list	Lists the actions (read, insert, replace, extend, erase, read attribute, change attribute, and delete file) that must be matched with the access request attributes.

concurrency_access	The optional concurrency key value that corresponds to concurrency locks (not required, shared, exclusive, and no access) for each action.
identity	An optional value that must match the initiator identity for the association.
passwords	An optional value that lists a password for each action that must match the corresponding password in the access passwords attribute.
location	An optional application-entity title value that must match the application-entity title attribute.

4.16. Access Passwords

Access passwords are part of the access control structure and are part of the F-CREATE and F-READ-ATTRIBUTES primitives. The following `osif_apwd` structure provides the mechanism for setting access passwords as required by the security attribute group.

```
struct osif_apwd {
struct osif_sdesc chng_attrib_password;
struct osif_sdesc delete_password;
struct osif_sdesc erase_password;
struct osif_sdesc extend_password;
struct osif_sdesc insert_password;
struct osif_sdesc read_attrib_password;
struct osif_sdesc read_password;
struct osif_sdesc replace_password;
};
```

Field descriptions:

chng_attrib_password	The character or octet string describing the password for change attributes.
delete_password	The character or octet string describing the password for delete.
erase_password	The character or octet string describing the password for erase.
extend_password	The character or octet string describing the password for extend.
insert_password	The character or octet string describing the password for insert.
read_attrib_password	The character or octet string describing the password for read attributes.
read_password	The character or octet string describing the password for read.
replace_password	The character or octet string describing the password for replace.

Note that the type field of the `osif_sdesc` structure must be set to OSIF_UC_GRAPHIC or OSIF_UC_OCTET depending on the semantics of the password.

4.17. File Access Data Unit

The following `osif_faduid` structure specifies the target FADU to be used for file operations.

```
struct osif_faduid {
struct osif_bdesc fadu_number;
struct osif_bdesc fadu_ref_begin_end;
struct osif_bdesc fadu_ref_first_last;
```

```
struct osif_bdesc fadu_ref_relative;
struct osif_bdesc name_list;
struct osif_bdesc single_name;
};
```

Field descriptions:

fadu_number	Specifies the selected node by its number in the preorder traversal sequence for the file access structure.
fadu_ref_begin_end	Indicates that the "next" FADU in the preorder traversal list will be the first one in the file structure if this parameter is set to OSIF_FADU_ID_BEGIN, or that the "previous" FADU is the last FADU in the file structure if it is set to OSIF_FADU_ID_END.
fadu_ref_first_last	Identifies the first FADU in the preorder traversal sequence for the file structure if this parameter is set to OSIF_FADU_ID_FIRST, or the last FADU if it is set to OSIF_FADU_ID_LAST.
fadu_ref_relative	Identifies the location of FADUs in terms of "previous," "current," and "next" in relation to the currently identified FADU and the preorder traversal sequence of the file access structure by setting this value to OSIF_FADU_ID_PREVIOUS, OSIF_FADU_ID_CURRENT, and OSIF_FADU_ID_NEXT respectively.
name_list	Specifies a path of FADU identifiers from the root node of the file to the node to be located.
single_name	Identifies the specified FADU.

4.18. File Access Data Unit Access Context

The following `osif_faduac` structure is used to specify the file access structure for read operations.

```
struct osif_faduac {
struct osif_bdesc fadu_context;
struct osif_bdesc fadu_level;
};
```

Field descriptions:

fadu_context	Indicates one of the following file access structures: OSIF_ACC_CTX_HA — Hierarchical all data units (HA) OSIF_ACC_CTX_HN — Hierarchical no data units (HN) OSIF_ACC_CTX_FA — Flat all data units (FA) OSIF_ACC_CTX_FL — Flat one level data units (FL) OSIF_ACC_CTX_FS — Flat single data unit (FS) OSIF_ACC_CTX_UA — Unstructured all data units (UA)

	OSIF_ACC_CTX_US — Unstructured single data unit (US)
fadu_level	An optional value used only if FL access context is selected.

4.19. Charging

The following `osif_charging_pb` structure is a Null terminated linked list that passes cost information attributed to the account during the regime being released.

```
struct osif_charging_pb {
struct osif_charging_pb *next;
struct osif_sdesc charging_unit;
struct osif_bdesc charging_value;
struct osif_sdesc resource_identifier;
};
```

Field descriptions:

*next	A pointer to the next charging structure.
charging_unit	A GraphicString charging unit.
charging_value	An integer charging value.
resource_identifier	A GraphicString resource identifier.

4.20. User Buffer

The `osif_buffer_list` structure is for user data buffers. Information in these buffers is referenced by the `osifpb` structure. These buffers are returned to the user as a result of a successful `osif_get_event` or `osif_deassign_port` call.

```
struct osif_buffer_list {
struct osif_buffer_list *next;
int buffer_length;
char *bufferptr;
};
```

Field descriptions:

*next	A pointer to the next buffer list structure.
buffer_length	An integer describing the length of the buffer.
*bufferptr	A pointer to the beginning of the user buffer.

Chapter 5. Function Calls

This chapter describes the following FTAM application programming interface (FTAM API) function calls:

- osif_assign_port
- osif_deassign_port
- osif_get_event
- osif_give_buffer
- osif_send

A success or failure value is returned as an indicator. Specific information detailing the cause of a failure is returned in the `error_code` argument. If the `error_code`argument is set to `osif_protocol_error`, then the `osif_protocol_error` vector contains a list of layer-specific errors in a null terminated list. The `osif_protocol_error_count` variable will be set to the number of returned errors.

The rest of this chapter describes the calls and refers to `osifpb` and its descriptors which were described in Section 4.1 and Section 3.4.

osif_assign_port

osif_assign_port — Creates a communication port.

Syntax

`status=osif_assign_port(port_id,pb_ptr,port_flags,error_code)`

Argument	Data Type	Passing Mechanism	Access
port_id	unsigned longword	by reference	write only
pb_ptr	`osifpb` structure	by reference	read only
port_flags	unsigned longword	by value	read only
error_code	unsigned longword	by reference	write only

C Binding

```
osif_assign_port(port_id,pb_ptr,port_flags,error_code)

unsigned *port_id;
struct osifpb *pb_ptr;
unsigned port_flags;
unsigned *error_code;
```

Arguments

port_id

This argument is a reference to a communication port. It is used as an identifier to map FTAM events to a specific process. Subsequent API functions must use this identifier.

pb_ptr

This argument is a pointer to the `osifpb` structure, which is used to pass values to the API. This argument is required if the program acts as a responder. This argument must be zero (null pointer) if the program acts as an initiator.

port_flags

This argument indicates if the initiator or the responder is using `osif_assign_port` function call. It accepts the following values:

OSIF_ASSIGN_INITIATOR	initiator
OSIF_ASSIGN_RESPONDER	responder (active)
OSIF_ASSIGN_REDIRECT	responder (passive)

For more information about active and passive FTAM responders, see Section 1.8.

error_code

The `error_code` argument provides further information if the status returned from the call is OSIF_FAILURE.

Description

This function call is used to establish a port for communication. A port identifier is returned to the caller to be used in subsequent calls involving the particular association. A port must be assigned for each FTAM initialization regime to be established. This call allows users to open connections to the FTAM protocol machine. It can be used by an initiator or a responder. If the initiator is using this call, the `pb_ptr` argument has no values that need to be passed to the API and the `error_code` argument is set if an error occurs. If the responder is using this call, the `pb_ptr` argument must be passed to the API with the following fields filled in (note that the local NSAP does not have to be specified):

Local AE-qualifier	osif_local_p_addrs.ae_addr.ae_qualifier.address osif_local_p_addrs.ae_addr.ae_qualifier.length
Local AP-title	osif_local_p_addrs.ae_addr.ap_title.address osif_local_p_addrs.ae_addr.ap_title.length
Local presentation address	osif_local_p_addrs.p_address.address osif_local_p_addrs.p_address.length

Return Values

OSIF_FAILURE	A port could not be assigned. The value returned in the argument `error_code` provides further details. Possible values are: OSIF_NOMEM — There was not enough memory for the operation OSIF_NOPORT — The call did not have a port identifier
OSIF_SUCCESS	A port was assigned.

Examples

This example illustrates the use of the `osif_assign_port` function by an initiator.

```
unsigned  status;
unsigned  error_status;
unsigned  port_id;

status = osif_assign_port ( &port_id,
                            NULL,
                            OSIF_ASSIGN_INITIATOR,
                            &error_status );
```

This example illustrates the use of the `osif_assign_port` function by a responder. In this case, the local AE-qualifier and AP-title are null.

```
unsigned  status;
unsigned  error_status;
unsigned  port_id;

struct osifpb assign_pb;

memset(&assign_pb, 0, sizeof(assign_pb) );

assign_pb.osif_local_p_addrs.p_address.address =
    (unsigned char *)LOCAL_P_ADDRESS;
assign_pb.osif_local_p_addrs.p_address.length =
    strlen (LOCAL_P_ADDRESS);

status = osif_assign_port( &port_id,
                           &assign_pb,
                           OSIF_ASSIGN_RESPONDER,
                           &error_status );
```

osif_deassign_port

osif_deassign_port — Destroys a communication port.

Syntax

`status=osif_deassign_port(port_id,user_buffer_listptr,port_flags,error_code)`

Argument	Data Type	Passing Mechanism	Access
port_id	unsigned longword	by value	read only
user_buffer_listptr	pointer to `osif_buffer_list` structure	by reference	write only
port_flags	unsigned longword	by value	read only
error_code	unsigned longword	by reference	write only

C Binding

`osif_deassign_port(port_id,user_buffer_listptr,port_flags,error_code)`

```
unsigned port_id;
struct osif_buffer_list **user_buffer_listptr;
unsigned port_flags;
unsigned *error_code;
```

Arguments

port_id

This argument is a reference to a communication port.

user_buffer_listptr

This argument contains a list of the buffers previously owned by the FTAM API that are being returned to the user upon deassignment of the port.

port_flags

This argument has a value of OSIF_ASSIGN_INITIATOR or OSIF_ASSIGN_ RESPONDER and indicates if the initiator or the responder is using the `osif_deassign_port` call.

This argument should be the same as the `port_flags` argument passed to the `osif_assign_port` call, except that if OSIF_ASSIGN_REDIRECT was used on `osif_assign_port`, OSIF_ASSIGN_RESPONDER should be used here.

error_code

The `error_code` argument provides further information if the status returned from the call is OSIF_FAILURE.

Description

This function call is used to destroy a communication port. A reference to the port to be destroyed is provided by the caller. This call allows users to close connections to the FTAM protocol machine.

Return Values

OSIF_FAILURE	The port could not be deassigned. The value returned in the argument `error_code` provides further details. A possible value is: OSIF_INVPORT — The call contained an invalid port identifier
OSIF_SUCCESS	The port was deassigned.

Example

This example illustrates the use of the `osif_deassign_port` function by an initiator.

```
unsigned status;
unsigned error_status;
unsigned port_id;
struct osif_buffer_list  *buffer_list;
struct osif_buffer_list  *buf_entry;
```

```
struct osif_buffer_list  *tmp_buf_entry;

status = osif_deassign_port ( port_id,
                              &buffer_list,
                              OSIF_ASSIGN_INITIATOR,
                              &error_status );

if ( buffer_list )
{
    for ( buf_entry = buffer_list; buf_entry;)
    {
    free( buf_entry->bufferptr );
    tmp_buf_entry = buf_entry;
    buf_entry = buf_entry->next;
    free( tmp_buf_entry );
    }
}
```

osif_get_event

osif_get_event — Solicits inbound events from the FTAM API.

Syntax

`status=osif_get_event(port_id,pb_ptr,timeout,error_code)`

Argument	Data Type	Passing Mechanism	Access
port_id	unsigned longword	by value	read only
pb_ptr	`osifpb` structure	by reference	write only
timeout	signed longword	by value	read only
error_code	unsigned longword	by reference	write only

C Binding

```
osif_get_event(port_id,pb_ptr,timeout,error_code)

unsigned port_id;
struct osifpb *pb_ptr;
long timeout;
unsigned *error_code;
```

Arguments

port_id

This argument is the reference of the communication port which is being solicited for reception of inbound events.

pb_ptr

This argument is a reference to the `osifpb` user parameter block supplied by the user to the FTAM API. The FTAM API fills in the appropriate parameters to describe the contents of the received protocol data unit (PDU). For example, the `osif_block_type` parameter determines the type of FTAM event

received. If parameters have default values, they are applied by the FTAM API if the parameter is not present in the PDU.

This call can be used to allow the reuse of buffers. If the `*osif_returned_buffer` parameter is filled in, the FTAM API user may reuse any of the buffers on the list. However, buffers must not be reused until all the information in the received `osifpb` has been processed or information will be lost.

timeout

This argument is the time in seconds indicating how long the `osif_get_event` call should wait before returning to the user. If timeout is a positive value, it indicates the time in seconds to wait for the call to complete. A timeout value of OSIF_WAIT_INFINITE indicates a synchronous call and the call blocks indefinitely until an event is received. A timeout value of OSIF_WAIT_NONE indicates a poll and the call returns immediately.

error_code

The `error_code` argument provides further information if the status returned from the call is OSIF_FAILURE.

Description

This function is used to solicit inbound events on a specified port. This call receives an incoming FTAM event in either synchronous or asynchronous mode, depending on the value of the timeout argument.

For FTAM primitives, this call will populate the `osifpb` user parameter block with the function code corresponding to the type of primitive received as well as the primitive-specific attribute values (or length/address pairs in the case of strings).

For FTAM data, this call will:

- Copy the file data to a suitably-sized user-supplied buffer (if available).
- Fill in the user parameter block with the F-DATA function code.
- Fill in the data address parameter of the F-DATA user parameter block with the address of the buffer containing the file data.

The `osif_get_event` call must be used in conjunction with the `osif_give_buffer` function call. The `osif_give_buffer` call provides user-supplied buffers to the FTAM API which uses these buffers to generate FTAM events. If no buffers have been supplied through the `osif_give_buffer` call, then the `osif_get_event` call will fail.

Return Values

OSIF_FAILURE	The FTAM protocol data unit could not be received. The value returned in the argument `error_code` provides further details. Possible values are: OSIF_INVPORT — The call contained an invalid port identifier OSIF_NO_EVENT — The event was not found during the timeout period OSIF_NOBUFFS — Buffers were unavailable or not large enough

	OSIF_NOMEM — There was not enough memory for the operation OSIF_XPORTFAILURE — There were failures at the Transport level OSIF_RECOVERY_EVENT - There is a recovery in progress
OSIF_SUCCESS	A PDU was successfully received and passed to the caller.

Example

This example illustrates the use of the `osif_get_event` function.

```
unsigned port_id;
unsigned status;
unsigned error_status;
struct osifpb f_initialize_response;

memset(&f_initialize_response, 0, sizeof(f_initialize_response));

status = osif_get_event( port_id,
                         &f_initialize_response,
                         OSIF_WAIT_INFINITE,
                         &error_status );
```

osif_give_buffer

osif_give_buffer — Posts a linked list of buffers to the port for reception of inbound events.

Syntax

`status=osif_give_buffer(port_id,user_buffer_listptr,error_code)`

Argument	Data Type	Passing Mechanism	Access
port_id	unsigned longword	by value	read only
user_buffer_listptr	`osif_buffer_list` structure	by reference	read only
error_code	unsigned longword	by reference	write only

C Binding

```
osif_give_buffer(port_id,user_buffer_listptr,error_code)

unsigned port_id;
struct osif_buffer_list *user_buffer_listptr;
unsigned *error_code;
```

Arguments

port_id

This argument is a reference to the port to which the buffers are being posted.

user_buffer_listptr

This argument is the address of a linked list of buffers being posted to the interface for use in receiving inbound events. Buffers are returned to the user as data (never as parameter blocks) on reception of inbound events (`osif_get_event`) or when the port is deassigned (`osif_deassign_port`).

error_code

The `error_code` argument provides further information if the status returned from the call is OSIF_FAILURE.

Description

This routine is used to post buffers to a port for the purpose of receiving inbound events. These buffers are used by the interface to return inbound events to the user. The buffers are returned to the user on either `osif_get_event` calls or on an `osif_deassign_port` call. Buffers returned on an `osif_deassign_port` call are unused and the contents are undefined.

Return Values

OSIF_FAILURE	The buffers could not be posted to the interface. The value returned in the argument `error_code` provides further details. Possible values are: OSIF_INVPORT — The call contained an invalid port identifier OSIF_NOMEM — There was not enough memory for the operation
OSIF_SUCCESS	The buffers were successfully posted to the interface.

Example

This example illustrates the use of the `osif_give_buffer` function.

```
unsigned status;
unsigned error_status;
unsigned port_id;
struct osif_buffer_list *buffer_list;
int i;

for ( i=0; i<5; i++ )
{
    buffer_list = ( struct osif_buffer_list *)
        malloc( sizeof( struct osif_buffer_list ));

    if (!buffer_list)
        exit(0);

    buffer_list->next = 0;
    buffer_list->buffer_length = 8*1024;
    buffer_list->bufferptr = (char *) malloc( 8*1024 );

    status = osif_give_buffer( port_id,
                               buffer_list,
                               &error_status );
}
```

osif_send

osif_send — Sends an FTAM protocol data unit to the cooperating implementation.

Syntax

`status=osif_send(port_id,pb_ptr,error_code)`

Argument	Data Type	Passing Mechanism	Access
port_id	unsigned longword	by value	read only
pb_ptr	osifpb structure	by reference	read only
error_code	unsigned longword	by reference	write only

C Binding

```
osif_send(port_id,pb_ptr,error_code)

unsigned port_id;
struct osifpb *pb_ptr;
unsigned *error_code;
```

Arguments

port_id

This argument is the reference of the communication port on which to send the protocol data unit.

pb_ptr

This argument is the address of the parameter block whose contents are to be encoded and sent to the peer entity.

error_code

The `error_code` argument provides further information if the status returned from the call is OSIF_FAILURE.

Description

This routine is used to send a protocol data unit (PDU) to a cooperating implementation. Upon invocation, it validates each parameter of the osifpb user parameter block. If any of the parameters in `osifpb` are invalid, the interface returns with an error. If all parameters are valid, then an FTAM PDU is encoded and posted to the lower layers.

This call is used to create all the request and response FTAM PDUs. All the parameters required by the specific FTAM service primitive must be in the `osifpb` that is passed to this call.

Return Values

OSIF_FAILURE	The FTAM protocol data unit could not be sent. The value returned in the argument `error_code` provides further details. Possible values are listed in Appendix A.

OSIF_SUCCESS	A PDU was successfully encoded and posted to the lower layers.

Example

This example illustrates the use of the `osif_send` function.

```
unsigned port_id;        /* port id */
char *init_id = "username";          /* initiator id */
char *fs_passwd = "password";        /* filestore password */
unsigned status;                     /* Call completion status */
unsigned error_status;               /* Additional status info value */
struct osifpb f_initialize_request;  /* Request parameter block */

struct osif_ctl ctlblk_1;            /* Temp structures for building */
struct osif_ctl ctlblk_2;            /* a contents type list */
struct osif_ctl ctlblk_3;
/*
* Zero fill the parameter block and
* contents type list blocks
*/
memset (&f_initialize_request, 0, sizeof(f_initialize_request));
memset (&ctlblk_1, 0, sizeof(ctlblk_1));
memset (&ctlblk_2, 0, sizeof(ctlblk_2));
memset (&ctlblk_3, 0, sizeof(ctlblk_3));
/*
* Set the f-initialize function code in the parameter block
*/
f_initialize_request.osif_block_type = OSIF_PBDEF_INIT_REQ;
f_initialize_request.osif_block_size = sizeof(f_initialize_request);
/*
* Store the systems presentation address in the parameter block
*/
f_initialize_request.osif_local_p_addrs.p_address.address =
    (unsigned char *)LOCAL_P_ADDRESS;
f_initialize_request.osif_local_p_addrs.p_address.length =
    strlen (LOCAL_P_ADDRESS);
f_initialize_request.osif_local_p_addrs.nsap_queue[0].nsap.address =
    (unsigned char *)LOCAL_NSAP;
f_initialize_request.osif_local_p_addrs.nsap_queue[0].nsap.length =
    strlen (LOCAL_NSAP);
f_initialize_request.osif_local_p_addrs.nsap_queue[0].template.address =
    (unsigned char *)LOCAL_TEMPLATE;
f_initialize_request.osif_local_p_addrs.nsap_queue[0].template.length =
    strlen (LOCAL_TEMPLATE);
f_initialize_request.osif_local_p_addrs.nsap_queue[0].provider.address =
    (unsigned char *)LOCAL_PROVIDER;
f_initialize_request.osif_local_p_addrs.nsap_queue[0].provider.length =
    strlen (LOCAL_PROVIDER);
/*
* Store the remote systems presentation address in the parameter block
*/
f_initialize_request.osif_peer_p_addrs.p_address.address =
    (unsigned char *)REMOTE_P_ADDRESS;
f_initialize_request.osif_peer_p_addrs.p_address.length =
    strlen (REMOTE_P_ADDRESS);
f_initialize_request.osif_peer_p_addrs.nsap_queue[0].nsap.address =
    (unsigned char *)REMOTE_NSAP;
```

```
f_initialize_request.osif_peer_p_addrs.nsap_queue[0].nsap.length =
    strlen (REMOTE_NSAP);
f_initialize_request.osif_peer_p_addrs.nsap_queue[0].template.address =
    (unsigned char *)REMOTE_TEMPLATE;
f_initialize_request.osif_peer_p_addrs.nsap_queue[0].template.length =
    strlen (REMOTE_TEMPLATE);
f_initialize_request.osif_peer_p_addrs.nsap_queue[0].provider.address =
    (unsigned char *)REMOTE_PROVIDER;
f_initialize_request.osif_peer_p_addrs.nsap_queue[0].provider.length =
    strlen (REMOTE_PROVIDER);
/*
* Propose the transfer, management, and transfer and management
* service class. Note that the FTAM responder will choose the
* service class it will support for this association.
*/
f_initialize_request.osif_service_class.length = 4;
f_initialize_request.osif_service_class.value =
      ( OSIF_CLASS_XFR_MGMT |
        OSIF_CLASS_MGMT |
        OSIF_CLASS_XFR );
/*
* Propose the read, write, limited file management, enhanced file
* management and grouping functional units.
*/
f_initialize_request.osif_functional_units.length = 4;
f_initialize_request.osif_functional_units.value =
      ( OSIF_FU_READ |
        OSIF_FU_WRITE |
        OSIF_FU_LIMIT_FILE_MGMT |
        OSIF_FU_ENH_FILE_MGMT |
        OSIF_FU_GROUPING );
/*
* Propose the storage attribute group
*/
f_initialize_request.osif_attribute_groups.length = 4;
f_initialize_request.osif_attribute_groups.value =
        OSIF_ATG_STORAGE | OSIF_ATG_SECURITY;
/*
* Set the FTAM Quality of Service to no recovery
*/
f_initialize_request.osif_ftam_qual_service.length = 4;
f_initialize_request.osif_ftam_qual_service.value =
        OSIF_FQOS_NO_RECOVERY;
/*
* Tell the responder which files type we can support
* add FTAM-1, FTAM-2 and FTAM-3 to the contents_type_list
*/
ctlblk_1.document_name.address = (unsigned char *)"FTAM-1";
ctlblk_1.document_name.length = strlen( ctlblk_1.document_name.address );
ctlblk_1.next = &ctlblk_2;

ctlblk_2.document_name.address = (unsigned char *)"FTAM-2";
ctlblk_2.document_name.length = strlen( ctlblk_2.document_name.address );
ctlblk_2.next = &ctlblk_3;

ctlblk_3.document_name.address = (unsigned char *)"FTAM-3";
ctlblk_3.document_name.length = strlen( ctlblk_3.document_name.address );
ctlblk_3.next = 0;
```

```
f_initialize_request.osif_contents_type_list = &ctlblk_1;
/*
* Store the filestore password and the initiator id in the
* parameter block.
* Note that the type field of the sdesc structure is used for the
* filestore password. This is because the password could be encoded
* as either a graphic string or an octet string. The type field
* tells the asn1 encoder how to encode this parameter.
*/
f_initialize_request.osif_filestore_password.address =
    (unsigned char *)fs_passwd;
f_initialize_request.osif_filestore_password.length =
    strlen(f_initialize_request.osif_filestore_password.address);
f_initialize_request.osif_filestore_password.type = OSIF_UC_GRAPHIC;

f_initialize_request.osif_initiator_identity.address =
    (unsigned char *) init_id;
f_initialize_request.osif_initiator_identity.length =
    strlen( f_initialize_request.osif_initiator_identity.address);
/*
* Send the f-initialize request to the remote responder
*/
status = osif_send( port_id,
                    &f_initialize_request,
                    &error_status );
```

Appendix A. Error Messages

The following table is a list of messages specific to the API. It also includes a short explanation of the error message.

These are returned in the `error_code` argument of the `osif_send` call.

Message	Meaning
OSIF_BAD_ACCCNTX	Bad access context
OSIF_BAD_ACCCNTRL	Bad access control
OSIF_BAD_ACCPWD	Bad access password
OSIF_BAD_ACCOUNT	Bad account
OSIF_BAD_ACTID	Bad activity identifier
OSIF_BAD_ACTRES	Bad action result
OSIF_BAD_APPCNTX	Bad application contexts
OSIF_BAD_ARCLEN	Bad arc length
OSIF_BAD_ATTNAM	Bad attribute name
OSIF_BAD_ATTRGRP	Bad attribute groups
OSIF_BAD_CCCNTRL	Bad concurrency control
OSIF_BAD_CHARGE	Bad charging
OSIF_BAD_CHATPWD	Bad change attribute password
OSIF_BAD_CHKPWIN	Bad checkpoint window
OSIF_BAD_CNTTYLST	Bad contents type list
OSIF_BAD_CNTTYPE	Bad contents type
OSIF_BAD_CREPWD	Bad create password
OSIF_BAD_DELCNTX	Bad delete contexts
OSIF_BAD_DELPWD	Bad delete password
OSIF_BAD_DELVAL	Bad delete value
OSIF_BAD_DIAG	Bad diagnostic
OSIF_BAD_DTCRE	Bad date and time of creation
OSIF_BAD_DTLATMD	Bad date and time of last attribute modification
OSIF_BAD_DTLMOD	Bad date and time of last modification
OSIF_BAD_DTLSTRD	Bad date and time of last read
OSIF_BAD_ERAPWD	Bad erase password
OSIF_BAD_EXTPWD	Bad extend password
OSIF_BAD_FADU	Bad file access data unit
OSIF_BAD_FADULK	Bad FADU lock
OSIF_BAD_FADULKG	Bad enable FADU locking
OSIF_BAD_FADUOP	Bad FADU operation
OSIF_BAD_FILEAV	Bad file availability
OSIF_BAD_FILENM	Bad file name

Message	Meaning
OSIF_BAD_FILESZ	Bad file size
OSIF_BAD_FSPWD	Bad filestore password
OSIF_BAD_FQOS	Bad FTAM quality of service
OSIF_BAD_FUNITS	Bad functional units
OSIF_BAD_FUTFISZ	Bad future file size
OSIF_BAD_IDCRE	Bad identity of creator
OSIF_BAD_IDLATMD	Bad identity of last attribute modifier
OSIF_BAD_IDLMOD	Bad identity of last modifier
OSIF_BAD_IDLREAD	Bad identity of last reader
OSIF_BAD_IMPINFO	Bad implementation information
OSIF_BAD_INITID	Bad initiator identity
OSIF_BAD_INSPWD	Bad insert password
OSIF_BAD_INVAL	Bad insert values
OSIF_BAD_LAEQUAL	Bad local AE-qualifier
OSIF_BAD_LAPTITLE	Bad local AP-title
OSIF_BAD_LEQUAL	Bad legal qualification
OSIF_BAD_LPADDR	Bad local presentation address
OSIF_BAD_OVRRIDE	Bad override
OSIF_BAD_PERACT	Bad permitted actions
OSIF_BAD_PCTXMGT	Bad presentation context management
OSIF_BAD_PROMODE	Bad processing mode
OSIF_BAD_PROTID	Bad protocol version
OSIF_BAD_PRVUSE	Bad private use
OSIF_BAD_RAEQUAL	Bad remote AE-qualifier
OSIF_BAD_RAPTITLE	Bad remote AP-title
OSIF_BAD_RDATPWD	Bad read attribute password
OSIF_BAD_RDPWD	Bad read password
OSIF_BAD_RECMODE	Bad recovery mode
OSIF_BAD_REMCNTX	Bad remove contexts
OSIF_BAD_REQACC	Bad requested access
OSIF_BAD_RPADDR	Bad remote presentation address
OSIF_BAD_RPLPWD	Bad replace password
OSIF_BAD_SRVCLASS	Bad service class
OSIF_BAD_STOACC	Bad storage account
OSIF_BAD_STRES	Bad state result
OSIF_BAD_THRES	Bad threshold
OSIF_BAD_USRDATA	Bad user data
OSIF_BADITEMSIZE	The call contained a bad item size

Message	Meaning
OSIF_FAILURE	The operation failed
OSIF_INVPORT	The call contained an invalid port identifier
OSIF_NO_EVENT	The event was not found during the timeout period
OSIF_NOBUFFS	Buffers were unavailable or not large enough
OSIF_NOMEM	There was not enough memory for the operation
OSIF_NOPORT	The call did not have a port identifier
OSIF_PARAMNORD	The parameter could not be read
OSIF_PARAMNOWRT	The parameter could not be written
OSIF_PROTOCOL_ERROR	There are layer-specific protocol errors
OSIF_RECOVERY_EVENT	There is a recovery in progress
OSIF_SUCCESS	The operation succeeded
OSIF_XPORTFAILURE	There were failures at the Transport level

Appendix B. Diagnostic Errors

This chapter provides all the constants returned as error identifiers with the `osif_diagnostic` described in Section 3.4.

The identifiers and reason codes from ISO 8571-3 are provided with the following list of constants.

Identifier	Constant	Reason
General FTAM Diagnostics		
0	OSIF_GEN_NOREASON	No reason
1	OSIF_GEN_RESPERR	Responder error (unspecific)
2	OSIF_GEN_SYSSHUT	System shutdown
3	OSIF_GEN_MGMT	FTAM management problem (unspecific)
4	OSIF_GEN_MGMACCT	FTAM management, bad account
5	OSIF_GEN_MGMSECURITY	FTAM management, security not passed
6	OSIF_GEN_DELAY	Delay may be encountered
7	OSIF_GEN_INITERR	Initiator error (unspecific)
8	OSIF_GEN_SUBSERR	Subsequent error
9	OSIF_GEN_INSFRSRC	Temporal insufficiency of resources
10	OSIF_GEN_VFSSEC	Access request violates VFS security
11	OSIF_GEN_LCLSEC	Access request violates local security
Protocol and Supporting Service Related Diagnostics		
1000	OSIF_PRO_CNFPRMVAL	Conflicting parameter values
1001	OSIF_PRO_UNSPRMVAL	Unsupported parameter values
1002	OSIF_PRO_MNDPARAM	Mandatory parameter not set
1003	OSIF_PRO_UNSPARAM	Unsupported parameter
1004	OSIF_PRO_DUPPARAM	Duplicated parameter
1005	OSIF_PRO_ILLPRMTYP	Illegal parameter type
1006	OSIF_PRO_UNSPRMTYP	Unsupported parameter types
1007	OSIF_PRO_PROT	FTAM protocol error (unspecific)
1008	OSIF_PRO_PROTPROC	FTAM protocol error, procedure error
1009	OSIF_PRO_PROTFUNC	FTAM protocol error, functional unit error
1010	OSIF_PRO_PROTCORR	FTAM protocol error, corruption error
1011	OSIF_PRO_LWRLYR	Lower layer failure
1012	OSIF_PRO_LWRLYRADRS	Lower layer addressing error
1013	OSIF_PRO_TIMEOUT	Timeout
1014	OSIF_PRO_SYSSHUT	System shutdown
1015	OSIF_PRO_ILLGRP	Illegal grouping sequence
1016	OSIF_PRO_GRPTHRSH	Grouping threshold violation
1017	OSIF_PRO_PDUINC	Specific PDU request inconsistent with the current requested access

Identifier	Constant	Reason
Association Related Diagnostics		
2000	OSIF_ASC_ASCNOTALL	Association with user not allowed
2001	OSIF_ASC_NOTDEFINED	(not assigned)
2002	OSIF_ASC_SRVCCLS	Unsupported service class
2003	OSIF_ASC_FUNCUNI	Unsupported functional unit
2004	OSIF_ASC_ATTGRP	Attribute group error (unspecific)
2005	OSIF_ASC_ATTGRPNS	Attribute group not supported
2006	OSIF_ASC_ATTGRPNA	Attribute group not allowed
2007	OSIF_ASC_BADACCT	Bad account
2008	OSIF_ASC_ASCMGM	Association management (unspecific)
2009	OSIF_ASC_ASCMGMADRS	Association management - bad address
2010	OSIF_ASC_ASCMGMACCT	Association management - bad account
2011	OSIF_ASC_CHKWINDLRG	Checkpoint window error - too large
2012	OSIF_ASC_CHKWINDSML	Checkpoint window error - too small
2013	OSIF_ASC_CHKWINDUNS	Checkpoint window error - unsupported
2014	OSIF_ASC_COMMQOS	Communications QoS not supported
2015	OSIF_ASC_INITID	Initiator identity unacceptable
2016	OSIF_ASC_CTXMGMT	Context management refused
2017	OSIF_ASC_ROLLBACK	Rollback not available
2018	OSIF_ASC_CTLCUTRESP	Contents type list cut by responder
2019	OSIF_ASC_CTLCUTPRES	Contents type list by presentation service
2020	OSIF_ASC_INVPWD	Invalid filestore password
2021	OSIF_ASC_INCSVC	Incompatible service classes
Selection Related Diagnostics		
3000	OSIF_SEL_FILNOTFND	File name not found
3001	OSIF_SEL_SELATTR	Selection attributes not matched
3002	OSIF_SEL_INITATT	Initial attributes not possible
3003	OSIF_SEL_BADATTNAM	Bad attribute name
3004	OSIF_SEL_NONEXFILE	Non-existent file
3005	OSIF_SEL_FILEXISTS	File already exists
3006	OSIF_SEL_FILNOCREATE	File cannot be created
3007	OSIF_SEL_FILNODELETE	File cannot be deleted
3008	OSIF_SEL_CONCTLNA	Concurrency control not available
3009	OSIF_SEL_CONCTLNS	Concurrency control not supported
3010	OSIF_SEL_CONCTLNP	Concurrency control not possible
3011	OSIF_SEL_MORERESLOCK	More restrictive lock
3012	OSIF_SEL_FILEBUSY	File busy
3013	OSIF_SEL_FILENA	File not available

Identifier	Constant	Reason
3014	OSIF_SEL_ACSCTLNA	Access control not available
3015	OSIF_SEL_ACSCTLNS	Access control not supported
3016	OSIF_SEL_ACSCTLINC	Access control inconsistent
3017	OSIF_SEL_FILNAMTRNC	File name truncated
3018	OSIF_SEL_INITATTALT	Initial attributes altered
3019	OSIF_SEL_BADACCT	Bad account
3020	OSIF_SEL_SELECTOLD	Override selected existing file
3021	OSIF_SEL_RECROLD	Override deleted and recreated file with old attributes
3022	OSIF_SEL_RECRNEW	Create override deleted and recreate file with new attributes
3023	OSIF_SEL_OVERRIDE	Create override - not possible
3024	OSIF_SEL_AMBFILSPEC	Ambiguous file specification
3025	OSIF_SEL_INVCREPWD	Invalid create password
3026	OSIF_SEL_INVDELPWD	Invalid delete password on override
3027	OSIF_SEL_BADATTVAL	Bad attribute value
3028	OSIF_SEL_RQSTACCS	Requested access violates permitted actions
3029	OSIF_SEL_FUNCUNIT	Functional unit not available for requested access
3030	OSIF_SEL_CREATED	File created but not selected
File Management Related Diagnostics		
4000	OSIF_MNG_ATTNONEX	Attribute nonexistent
4001	OSIF_MNG_ATTNOREAD	Attribute cannot be read
4002	OSIF_MNG_ATTNOCHNG	Attribute cannot be changed
4003	OSIF_MNG_ATTNS	Attribute not supported
4004	OSIF_MNG_BADATTNAM	Bad attribute name
4005	OSIF_MNG_BADATTVAL	Bad attribute value
4006	OSIF_MNG_ATTPARSUP	Attribute partially supported
4007	OSIF_MNG_ATTVALND	Additional set attribute value not distinct
Access Related Diagnostics		
5000	OSIF_ACC_BADFADU	Bad FADU (unspecific)
5001	OSIF_ACC_BADFADUSIZ	Bad FADU - size error
5002	OSIF_ACC_BADFADUTYP	Bad FADU - type error
5003	OSIF_ACC_BADFADUPS	Bad FADU - poorly specified
5004	OSIF_ACC_BADFADULOC	Bad FADU - bad location
5005	OSIF_ACC_FADUNONEXI	FADU does not exist
5006	OSIF_ACC_FADUNA	FADU not available (unspecific)
5007	OSIF_ACC_FADUNARD	FADU not available for reading
5008	OSIF_ACC_FADUNAWR	FADU not available for writing

Identifier	Constant	Reason
5009	OSIF_ACC_FADUNALOC	FADU not available for location
5010	OSIF_ACC_FADUNAERA	FADU not available for erasure
5011	OSIF_ACC_FADUNOINS	FADU cannot be inserted
5012	OSIF_ACC_FADUNORPL	FADU cannot be replaced
5013	OSIF_ACC_FADUNOLOC	FADU cannot be located
5014	OSIF_ACC_BADDETYP	Bad data element type
5015	OSIF_ACC_OPERNA	Operation not available
5016	OSIF_ACC_OPERNS	Operation not supported
5017	OSIF_ACC_OPERINC	Operation inconsistent
5018	OSIF_ACC_CONCTLNA	Concurrency control not available
5019	OSIF_ACC_CONCTLNS	Concurrency control not supported
5020	OSIF_ACC_CONCTLINC	Concurrency control inconsistent
5021	OSIF_ACC_PRCMODNA	Processing mode not available
5022	OSIF_ACC_PRCMODNS	Processing mode not supported
5023	OSIF_ACC_PRCMODINC	Processing mode inconsistent
5024	OSIF_ACC_ACSCTXNA	Access context not available
5025	OSIF_ACC_ACSCTXNS	Access context not supported
5026	OSIF_ACC_BADWRITE	Bad write (unspecific)
5027	OSIF_ACC_BADREAD	Bad read (unspecific)
5028	OSIF_ACC_LCLERR	Local failure (unspecific)
5029	OSIF_ACC_LCLFILSPACE	Local failure - filespace exhausted
5030	OSIF_ACC_LCLDATCORR	Local failure - data corrupted
5031	OSIF_ACC_LCLDEVFAIL	Local failure - device failure
5032	OSIF_ACC_FUTSIZEXC	Future file size exceeded
	OSIF_ACC_UNDEFINED	
5034	OSIF_ACC_FUTSIZINC	Future file size increased
5035	OSIF_ACC_FUNCUNIT	Functional unit invalid in processing mode
5036	OSIF_ACC_CNTTYPINC	Contents type inconsistent
5037	OSIF_ACC_CNTTYPSMPL	Contents type simplified
5038	OSIF_ACC_DUPFADUNAM	Duplicate FADU name
5039	OSIF_ACC_DMGSELOPEN	Damage to select/open regime
5040	OSIF_ACC_FADULOCKNA	FADU locking not available on file
5041	OSIF_ACC_FADULOCKED	FADU locked by another user
Recovery Related Diagnostics		
6000	OSIF_REC_BADCHKPNT	Bad checkpoint (unspecific)
6001	OSIF_REC_ACTVNOTUNI	Activity not unique
6002	OSIF_REC_CHKOUTWIND	Checkpoint outside window
6003	OSIF_REC_ACTVNOEXIST	Activity no longer exists

Identifier	Constant	Reason
6004	OSIF_REC_ACTVNORECOG	Activity not recognized
6005	OSIF_REC_NODOCKET	No docket
6006	OSIF_REC_CORDOCKET	Corrupt docket
6007	OSIF_REC_WAITRESTART	File waiting restart
6008	OSIF_REC_BADRECPNT	Bad recovery point
6009	OSIF_REC_NONEXRECPNT	Non-existent recovery point
6010	OSIF_REC_RECMODNA	Recovery mode not available
6011	OSIF_REC_RECMODINC	Recovery mode inconsistent
6012	OSIF_REC_RECMODRED	Recovery mode reduced
6013	OSIF_REC_ACSCTLNA	Access control not available
6014	OSIF_REC_ACSCTLNS	Access control not supported
6015	OSIF_REC_ACSCTLINC	Access control inconsistent
6016	OSIF_REC_CNTTYPINC	Contents type inconsistent
6017	OSIF_REC_CNTTYPSMPL	Contents type simplified