SunLink FTAM 8.0.2 Programmer's Guide

SunSoft
A Sun Microsystems, Inc. Business
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No: 801-4962-12
Revision A, October 1994

Contents

Preface

Purpose and Audience

This manual describes five external programming interfaces that are used to develop FTAM applications such as the SunLink FTAM initiator (`osiftam`) and responder (`osiftr`). It is intended for programmers and system designers who are familiar with the SunOS™ operating system and C programming interfaces.

Refer to *Using SunLink FTAM 8.0.2* for a detailed description of the SunLink FTAM initiator and responder applications.

Before You Read This Book

You must install and configure the SunLink OSI Communications Platform (Stack) in order to use SunLink FTAM. Before reading this manual you should refer to the following related documentation:

- *SunLink OSI 8.0.2 Communication Platform Administrator's Guide* (Part No. 801-4975)

How This Book Is Organized

Chapter 1, "Architecture Overview," provides a overview of the FTAM initiator and responder modules and and introduces the five programming interfaces used to access the services that these modules provide.

Chapter 2, "FTAM Initiator External Interface Definitions," provides a detailed description of the programming interfaces to the FTAM initiator modules. These interfaces are used to develop FTAM client applications such the SunLink FTAM initiator (`osiftam`).

Chapter 3, "FTAM Responder Stubs File Definitions," provides a detailed description of the programming interface to the FTAM responder module. This interface is used to develop FTAM server applications such the SunLink FTAM responder (`osiftr`).

Chapter 4, "FTAM Diagnostic Codes," lists the diagnostic codes returned by the FTAM initiator and responder modules.

Appendix A, "Program Examples," contains an example of an FTAM client application developed using the interfaces described in this manual, and listings for the C header files `ftiuser.h` and `ftruser.h`. The example program is also provided as part the SunLink FTAM application software.

Conventions Used in this Manual

The following typographic conventions are used in this manual:

`Typewriter font`
: Represents what appears on your workstation screen and is used for program names, file names, and UNIX commands.

`Boldface typewriter font`
: Indicates user input, commands and responses to prompts that you type in exactly as they appear in the manual.

Italic font
: Indicates variables or parameters that you replace with an appropriate word or string; also used for emphasis.

`hostname%` *or* `%`
: Represents the system's prompt for a non-privileged user's account.

`hostname#` *or* `#`
Represents the system's prompt for the `root` (`superuser`) account.

Boxes

```
Contain text that represents listings, part of a file, or program
output.
```

Boxes are also used to represent interactive sessions. For example::

```
% df -k /usr
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/sd0g              155015  103090   36424    74%    /usr
```

Product Documentation

The documents in the SunLink FTAM 8.0.2 release document set are:

- *Using SunLink FTAM 8.0.2*
  (Part No. 801-4961)
- *Managing SunLink FTAM 8.0.2*
  (Part No. 802-1573)

Architecture Overview 1

This chapter provides an overview of the external interfaces to the File Transfer Access and Management (FTAM) protocols provided by SunLink FTAM. It lists the relevant ISO specifications, and describes the external interfaces (APIs) and stubs file used to develop FTAM clients and servers similar to the SunLink FTAM applications `osiftam` and `osiftr`. Refer to the *Managing SunLink FTAM 8.0.2* for a detailed description of the SunLink FTAM applications.

FTAM Specifications

The ISO FTAM specification (8571) describes a method of handling the transfer, access, and mangement of files between two communicating systems based on the concept of a *virtual file store*, which provides a mapping of diverse file systems to a common model. The FTAM specification is divided into five parts:

- **ISO 8571/1** Introduction to FTAM concepts
- **ISO 8571/2** Explains the terms, concepts, and vocabulary used by FTAM
- **ISO 8571/3** Describes the two file transfer entities
- **ISO 8571/4** Defines the rules applied to the transfer of files
- **ISO 8571/1** Describes the conformance statement

1

FTAM Initiator External Interfaces

SunLink FTAM provides four external interfaces that can be used to develop customized FTAM clients similar to the SunLink FTAM initiator application (`osiftam`):

- The **FTAM Initiator (FTI) interface** provides a set of functions that are used to access the main FTAM initiator services. These services handle file transfer, access, and management between peer entities—that is, between an FTAM initiator (client application) and an FTAM responder (server application).
- The **FTAM Remote Database (RDB) interface** provides a set of functions that are used to access the FTAM services which handle the OSI addresses required to establish an associations with remote hosts.
- The **FTAM Shadow File (SHF) interface** provides a set of functions that are used to access the FTAM shadow file services which handle the virtual file attributes that are not supported by the local file system—for example, document type or universal class number.
- The **FTAM Transfer (TRF) interface** provides a set of functions that are used to access the FTAM file recovery services which handle the recovery of interrupted file transfers based on information contained in special FTAM files called *dockets*.

Compiling and Linking FTAM Initiator Applications

To compile and link FTAM initiator applications developed using the functions defined within this chapter, you must specify the following entries in your Makefile:

```
CFLAGS  += -I /opt/SUNWconn/include
LDFLAGS += -L /opt/SUNWconn/lib_ftam
           -lintl
```

FTAM Responder Stubs File

The external interface to the SunLink FTAM responder takes the form of a stubs file that can be used to customize the SunLink FTAM responder application (`osiftr`).

The following elements are delivered as part of Sunlink FTAM:

- A stubs file (`ftr_stub.c`) that contains skeleton C functions called by the reponder process.
- A C header file (`ftruser.h`) that contains the variable definitions for inclusion in the responder program.
- A relocatable file (`osiftrapi.o`) that must be linked with the stubs file to generate the responder executable.

The functions contained in the stubs file are called by the responder process in response to FTAM activity.

For example,

> When the responder receives an `F-INITIALIZE` request (remote request for FTAM regime establishment), it calls the function `ftr_connect` to pass information regarding the connection (for example, remote user name, password, account).

In general, the FTAM responder interface is used to collect information about FTAM activity (for statistical or accounting purposes) or to prevent the execution of certain FTAM operations by returning an non-zero code that causes the responder to reject the request.

For example,

> The function `ftr_connect` is called by passing it a C data structure that contains information regarding the remote user that requested the FTAM regime establishment. A programmer can include code in the body of the function `ftr_connect` to perform strict checking on the remote user information and return an non-zero code for certain values.

Note – Note that the return codes of some functions are ignored by the responder (for example, `ftr_write`) and cannot be used in this way.

The following code example shows how the `ftr_connect` stub can be modified to reject an FTAM regime establishment request based on the remote user name.

```
int ftr_connect(ftrbndT*fbndp, ftresrT*fesrp)
{
    if( strcmp( fbndp->username, "REFUSE" ) )
        return( 1 );          /* FTAM connection REFUSED */
    else return( 0 );         /* FTAM connection ACCEPTED */
}
```

Generating a Customized Responder

To generate a customized FTAM responder process (called `osiftr_cust` in this example) based on the stubs file `ftr_stubs.c`:

1. **Create your own code to fill the skeleton functions contained in the stubs file** `ftr_stubs.c`**.**
   If you do not use the template stubs file provided with SunLink FTAM, you must remember to include the header files `ftiuser.h` and `ftruser.h` using the compiler directives:

```
#include   "ftam/ftiuser.h"
#include   "ftam/ftruser.h"
```

2. **Compile the customized stubs file and link it with the relocatable file** `/opt/SUNWconn/ftam/lib/osiftrapi.o`**.**

```
cc [ flags ...] -c -I/opt/SUNWconn/ftam/include ftr_stubs.c
cc [ flags ...] -o osiftr_cust /opt/SUNWconn/lib/osiftrapi.o
ftr_stubs.o [ libs ...] -lintl
```

   Note that you must also link with the standard internationalization library `libintl.a`. Use the `–c` flag when compiling to avoid automatic linking which will generate an error message because of the missing `main()`.

Starting a Customized Responder

The customized responder is started by the SunLink responder daemon (`osiftrd`) in the same way as the standard SunLink responder (`osiftr`); therefore, for a given network type (CONS, CNLP, or TCP/IP) you can only use *either* a standard SunLink responder *or* a customized responder.

To use a customized responder:

1. Log in as `root`, or become `superuser`.

```
hostname% su
Password: <enter your password>
hostname#
```

2. Make a backup copy of the standard responder application (`osiftr`).

```
hostname# cd /opt/SUNWconn/ftam/bin
hostname# mv osiftr osiftr.BAK
```

3. Create a symbolic link between the standard responder name and the customized responder application (where *osiftr_cust* is the full pathname in this example)

```
hostname# ln -s <osiftr_cust> /opt/SUNWconn/ftam/bin/osiftr
```

1

FTAM Initiator External Interface Definitions 2

This chapter describes the external interfaces to the FTAM initiator modules, which are used to develop FTAM initiators similar to the SunLink FTAM application `osiftam`.

Initiator Common Data Structures

All data structures, types, and constants are defined in the C header file `/opt/SUNWconn/include/ftam/ftiuser.h`. An FTAM client developed using the FTAM initiator interface must include this C header file, by using the following compiler directive:

```
#include “ftam/ftiuser.h”
```

2

Initiator Execution Status Report

Most FTAM initiator functions require the provision of a pointer to a structure of type `ftiesrT`. This output parameter contains the Execution Status Report of the operation. The structure is filled out as shown in Table 2-1:

Table 2-1 Execution Status Report

`status`	**Execution Status**. This field is set to one of the following values:	
	`FTISTNER`	Execution was successful, no error.
	`FTISTLOC`	A local or user error was detected.
	`FTISTSYS`	A local system error occurred.
	`FTISTFTM`	An FTAM service or protocol error occurred.
	`FTISTWAR`	A local warning was detected.
`errcode`	**Error Code**. Significant only when status is set to `FTISTLOC`, `FTISTSYS`, `FTISTFTM` or `FTISTWAR`. It specifies the type of error that occurred during the operation, as defined in the C header file `ftiuser.h`.	
`errtype`	**Error Type**. Significant only with recovery. It specifies if recovery is possible after the failure. This field is set to one of the following values:	
	`FTITYNRC`	Error not recoverable.
	`FTITYRCO`	Error recoverable.
`errcnxs`	**Connection Status**. It specifies the connection status after the failure. This field is set to one of the following values:	
	`FTICNXON`	Connection on.
	`FTICNXOF`	Connection off.
`diag`	**Diagnostic Code**. It gives additional information about the error:	
	When status is set to `FTISTLOC`, it is set to one of the local error diagnostic values listed in the C header file `ftiuser.h`.	
	When status is set to `FTISTSYS`, it contains the value of the system variable errno as set by the system call that failed.	
	When status is set to `FTISTFTM`, it is set to one of the values listed in Chapter 4, “FTAM Diagnostic Codes”.	
	When status is set to `FTISTWAR`, it is set to one or more of the local warning values listed in the C header file `ftiuser.h`.	
`ftamdfd`	**FTAM Diagnostic Further Details**. Significant only when status is set to `FTISTFTM`. (Reserved for future extensions.)	

Initiator External Interface Definitions Summary

The interfaces to the FTAM initiator modules are summarized in Table 2-2

Table 2-2 FTAM Initiator External Interface Definitions

Function	Page
`fti_connect()`—opens an association with a remote host .	*page 10*
`fti_create()`—creates a file on a remote host.	*page 12*
`fti_delete()`—deletes a file on a remote host.	*page 15*
`fti_dir()`—lists a remote directory using NBS-9 document type.	*page 16*
`fti_disconnect()`— closes an association with a remote host.	*page 21*
`fti_exit()`—terminates and exits the initiator process.	*page 22*
`fti_get()`—gets (or receives) a file from a remote host .	*page 23*
`fti_getid()`—returns a new transfer identifier.	*page 26*
`fti_init()`—initializes the initiator process.	*page 27*
`fti_lcd()`—changes the local directory path.	*page 28*
`fti_ratt()`—reads the attributes of a remote file.	*page 33*
`fti_rcd()`—changes the remote directory path.	*page 37*
`fti_read()`—reads part of an FTAM-2 file from a remote host.	*page 38*
`fti_recover()`—recovers an interrupted transfer.	*page 40*
`fti_rename()`—renames a file on a remote host system.	*page 43*
`fti_put()`—puts a file on (or sends a file to) a remote host.	*page 29*
`fti_pwd()`—gets the current remote directory path.	*page 32*
`fti_rdbdel()`—deletes an entry from the FTAM RDB.	*page 45*
`fti_rdbget()`—reads an entry from the FTAM RDB.	*page 46*
`fti_rdbput()`—adds a new entry to the FTAM RDB.	*page 47*
`fti_shfget()`—reads a shadow file.	*page 49*
`fti_shfput()`—creates or modifies a shadow file.	*page 51*
`fti_trf_del()`—deletes recovery information.	*page 54*
`fti_trf_list()`—returns a list of current interrupted transfers.	*page 55*
`fti_trf_show()`—reads recovery information.	*page 56*

2

FTAM Initiator (FTI) Interface Definitions

The **FTAM Initiator (FTI) interface** provides a set of functions that are used to access the main FTAM initiator services. These services handle file transfer, access, and management between peer entities—that is, between an FTAM initiator (client application) and an FTAM responder (server application).

`fti_connect()`—*open association*

The `fti_connect` function is used to establish an association between the FTAM initiator and the FTAM responder running on a remote system.

The function `fti_connect` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_connect(flaip,fbndp,frecp,fmode,fesrp)
ftilaiT   *flaip;
ftibndT   *fbndp;
int       *frecp;
int        fmode;
ftiesrT   *fesrp;
```

`flaip` Local Association Identifier. This output parameter identifies the FTAM association to be established. Its value is returned by the FTI API and must be provided by the initiator for all subsequent calls to FTI API functions that are related to this association.

`fbndp` Pointer to a structure of type `ftibndT`. This input parameter identifies the remote host system with which the association is to be established.

The structure is filled out as follows:

hostname Remote Host Name (null-terminated string). This parameter is used to retrieve the OSI addressing information required to establish the association. The remote host must be registered in the remote systems database (RSDB). See also `fti_rdbput()`.

username User Name (null terminated string). Corresponds to the *initiator identifier* in the `F-Initialize-Request`.

password User Password. Corresponds to the *filestore password* in the `F-Initialize-Request.` The password format is: <*type*><*length*><*value*>. The <*length*> may be zero and the <*type*> takes one of the following values:

`FTIGraStr` graphic string

`FTIOctStr` general string

account Account Name. Corresponds to the *account* in the `F-Initialize-Request.` May take a null value.

`frecp` Recovery Indicator. Significant only when recovery is enabled. This output parameter specifies the state of the recovery negotiation. The returned values are:

`FTIRCAVA` recovery available

`FTIRCNAV` recovery not available

`fmode` Execution Mode. Must be set to `FTIEXBLK` (blocking mode).

`fesrp` Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8.

The following error codes are returned in the Execution Status Report by the function `fti_connect`, as appropriate:

`FTIERIPA` invalid parameter

`FTIERHIC` host is closed

`FTIERHNR` host name required

`FTIERNRH` no response from host

`FTIERECH` error connecting to host

`FTIERCAE` connection already established

`FTIERRHI` error reading host information

2

`fti_create()`—*create remote file*

The function `fti_create` is used to create a new file on the remote system with which an association has been established using the function `fti_connect`. If a file of the same name exists on the remote system, the action is determined by the *override* parameter.

The function `fti_create` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_create(flaip,fcrep,fmode,fesrp)
ftilaiT   *flaip;
fticreT   *fcrep;
int        fmode;
ftiesrT   *fesrp;
```

`flaip` Local Association Identifier. This input parameter identifies the FTAM association to be established. Its value is recovered by the function `fti_connect` when the associaton is first established.

`fcrep` Pointer to a structure of type `fticreT`. Identifies the local file to be transferred to the remote system.

The structure is filled out as follows:

doctype Document Type. Specifies the type of file to be created. If the document type is specified, the function uses the corresponding shadow file template. If the document type is not specified, the document type is determined by the document type defined in the shadow file template specified by the *shfname* parameter.

`FTMDTFT1` FTAM-1 (unstructured text file)

`FTMDTFT2` FTAM-2 (sequential text file)

`FTMDTFT3` FTAM-3 (unstructured binary file)

`FTMDTIN1` INTAP-1 (INTAP record file)

`FTMDTUNO` Unknown document type

filsize Specifies the size of the new file in bytes. The `-z` option is only valid if it is supported by the remote FTAM responder (for example, the SunLink FTAM local responder does not recognize this option).

override Override Parameter. Specifies the action to be taken if the file exists.

`FTMOVNOC` No creation. Transaction fails.

`FTMOVDOA` Creation with old attributes.

`FTMOVDNA` Creation with new attributes.

`FTMOVSOF` Select old file.

peract Permitted Actions. This parameter consists of two bytes. The second byte is reserved for future extensions. The first byte is the result of a logical `OR` of the following values:

`FTMPAREA` read permission

`FTMPAINS` insert permission

`FTMPAEXT` extend permission

`FTMPAREP` replace permission

`FTMPAERA` erase permission

`FTMPADEL` delete permission

`FTMPARAT` read attributes permission

`FTMPACAT` change attributes permission

accnt Account Name. Specifies the name of the account to which access to the remote file is charged. Only significant to the remote system. May be null.

filpssd Password. Specifies a password for the creation of the remote file, if required by the remote system.The password format is: <*type*><*length*><*value*>. The <*length*> may be zero and the <*type*> takes one of the following values:

`fti_create()` continued...

`FTIGraStr` graphic string

`FTIOctStr` general string

filname Remote File Name. Specifies the name of the file to be created. The syntax must obey the rules imposed by the remote system. The complete path name is a concatenation of the internal variable set using the function `fti_rcd` and the filename specified by this parameter.

shfname Shadow File Template. Used to specify the name of a local file from which the initial attributes will be derived.

avail Availability. Always set to the constant `FTMFAIMM`.

`fmode` Execution Mode. Must be set to `FTIEXBLK` (blocking mode).

`fesrp` Pointer to an area to receive the Execution Status Report. See "Initiator Execution Status Report" on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERIPA` invalid parameter

`FTIERCNE` connection not established

`FTIERNRH` no response from host

`FTIERRFR` remote file name required

`FTIERIDT` invalid document type

`FTIERFNA` function not available (on this host)

`FTIERRCTF` error creating remote file

`FTIERRDSF` error deselecting remote file

`fti_delete()`—*delete remote file*

The function `fti_delete` is used to delete a file on a remote system with which an association has been established using the function `fti_connect`.

The function `fti_delete` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_delete(flaip,fdelp,fmode,fesrp)
ftilaiT   *flaip;
ftidelT   *fdelp;
int        fmode;
ftiesrT   *fesrp;
```

`flaip` Local Association Identifier. This input parameter identifies the FTAM association to be established. Its value is recovered by the function `fti_connect` when the associaton is first established.

`fdelp` Pointer to a structure of type `ftidelT`. Identifies the remote file to be deleted.

The structure is filled out as follows:

filname Remote File Name. Specifies the name of the file to be deleted. The syntax must obey the rules imposed by the remote system. The complete path name is a concatenation of the internal variable set using the function `fti_rcd` and the filename specified by this parameter.

accnt Account Name. Specifies the name of the account to which access to the remote file is charged. Only significant to the remote system. May be null.

filpssd Password. Specifies a password for deleting the remote file, if required by the remote system.The password format is: <*type*><*length*><*value*>. The <*length*> may be zero and the <*type*> takes one of the following values:

`FTIGraStr` graphic string

`FTIOctStr` general string

`fti_delete()` continued...

`fmode` Execution Mode. Must be set to `FTIEXBLK` (blocking mode).

`fesrp` Pointer to an area to receive the Execution Status Report. See "Initiator Execution Status Report" on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERIPA` invalid parameter

`FTIERCNE` connection not established

`FTIERNRH` no response from host

`FTIERRFR` remote file name required

`FTIERFNA` function not available (on this host)

`FTIERRSRF` error selecting remote file

`FTIERRDRF` error deleting remote file

`fti_dir()`—*list remote directory*

The function `fti_dir` is used to generate a listing of the contents of a directory on a remote system with which an association has been established using the function `fti_connect`.

The function `fti_dir` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_dir (flaip,ftrfp,fmode,fesrp)
ftilaiT   *flaip;
ftitrfT   *ftrfp;
int        fmode;
ftiesrT   *fesrp;
```

`flaip` Local Association Identifier. This input parameter identifies the FTAM association to be established. Its value is recovered by the function `fti_connect` when the associaton is first established.

`ftrfp` Pointer to a structure of type `ftitrfT`. Identifies the remote file to be transferred.

The structure is filled out as follows:

doctype Document type. Must be set to `FTMDTNB9` (NBS-9 directory).

accnt Account Name. Specifies the name of the account to which access to the remote file is charged. Only significant to the remote system. May be null.

locname Local File Name (null terminated string). Specifies a local file into which the directory listing will be written. If null, the listing will be displayed on the standard output device. The syntax must obey the rules imposed by the local system.

remname Remote Directory Name (null terminated string). Specifies the name of the remote directory. The syntax must obey the rules imposed by the remote system. The complete path name is a concatenation of the internal variable set using the function `fti_rcd` and the filename specified by this parameter.

filpssd Password. Specifies a password, if required by the remote system.The password format is: <*type*><*length*><*value*>. The <*length*> may be zero and the <*type*> takes one of the following values:

`FTIGraStr` graphic string

`FTIOctStr` general string

dform Attribute List Format. Takes the values 0 (short), 1 (long), 2 (extended).

reqattri Requested Attributes (three-byte array). Takes the following values if `dform=0`:

```
reqattri[0]=FTMAGKER0

reqattri[1]=FTMAGKER1 | FTMAFNAME

reqattri[2]=FTMAGKER2
```

`fti_dir()` continued...

Takes the following values if `dform=1` or `2`:

`reqattri[0]=FTMAGSTG0`

`reqattri[1]=FTMAGSTG1`

`reqattri[2]=FTMAGSTG2`

trfid Transfer Identifier. Significant only when recovery is enabled. Must be set to the value returned previously by the function `fti_getid`. Unpredicatble results can occur if *trfid* is not initialized correctly.

The following fields are returned; the remaining fields are not used:

result number of bytes received

elapse elapsed time (in seconds) for transfer

speed transfer speed (in Kbytes/second)

`fmode` Execution Mode. Must be set to `FTIEXBLK` (blocking mode).

`fesrp` Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERNSU` function not supported (on local system)

`FTIERIPA` invalid parameter

`FTIERCNE` connection not established

`FTIERNRH` no response from host

`FTIERRFR` remote file name required

`FTIERIDT` invalid document type

`FTIERCDK` error creating docket

`FTIEROLF` error opening local file

`FTIERORF` error opening remote file

`fti_dir()` continued...

`FTIERWDK`	error writing docket
`FTIERWLF`	error writing local file
`FTIERCLF`	error closing local file
`FTIERCRF`	error closing remote file

The attribute list format defined by the parameter dform is defined as follows:

short format (0):

```
<type><perm><class><de_size><f_size><f_name>
```

long format (1):

```
<type><perm><class><de_size><de_type><owner><f_size><lmo_dt><f_name>
```

extended format (2):

```
<type><perm><class><de_size><de_type><owner><f_size><f_name><cre_dt><lmo_dt>
<lac_dt><lam_dt>
```

<cre_dt>	Date and time of creation
<de_size>	Data element size
<de_type>	Data element type
<f_size>	Actual file size
<lac_dt>	Date and time of last read access
<lam_dt>	Date and time of last attribute modification
<lmo_dt>	Date and time of last modification

`fti_dir()` continued...

<*type*>	Specifies the FTAM document type.
	F1: Unstructured text file (FTAM-1)
	F2: Sequential text file (FTAM-2)
	F2: Unstructured binary file (FTAM-3)
	I1: INTAP record file (INTAP-1)
	N9: Directory (NBS-9)
<*owner*>	Identity of owner of FTAM creator of file
<*perm*>	The initial permissions assigned to the new file, which take precedence over the default permissions assigned in the shadow file. The permissions are:
	r: read permission
	i: insert permission
	p: replace permission
	x: extend permission
	e: erase permission
	t: read attributes permission
	c: change attributes permission
	d: delete permission
<*class*>	Integer value specifying character set:
	4: octet string
	19: printable string
	20: teletex string
	21 videotex string
	22: IA5 string (default value)
	25: graphic string

`fti_dir()` continued...

26: visible string

27: general string

`fti_disconnect()`—*close association*

The function `fti_disconnect` is used to close (or terminate) an association which has been established using the function `fti_connect`.

The function `fti_disconnect` returns zero if successful and a non-zero value unsuccessful.

```
int fti_disconnect(flaip,fmode,fesrp)
ftilaiT   *flaip;
int        fmode;
ftiesrT   *fesrp;
```

`flaip` Local Association Identifier. This input parameter identifies the FTAM association. Its value is recovered by the function `fti_connect` when the associaton is first established.

`fmode` Execution Mode. Must be set to `FTIEXBLK` (blocking mode).

`fesrp` Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERIPA` invalid parameter

`FTIERCNE` connection not established

`FTIERCRH` error closing remote host

`fti_exit()`—*exit initiator process*

The function `fti_exit` is used to close (or terminate) an association which has been established using the function `fti_connect` and to exit the initiator process

```
int fti_exit()
```

`fti_get()`—*get remote file*

The function `fti_get` is used to transfer a file from a remote system with which an association has been established using the function `fti_connect`. If a document type is specified, it must match the actual type of the remote file otherwise the transfer will fail.

The function `fti_get` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_get(flaip,ftrfp,fmode,fesrp)
ftilaiT   *flaip;
ftitrfT   *ftrfp;
int        fmode;
ftiesrT   *fesrp;
```

`flaip` Local Association Identifier. This input parameter identifies the FTAM association. Its value is recovered by the function `fti_connect` when the associaton is first established.

`ftrfp` Pointer to a structure of type `ftitrfT`. Identifies the remote file to be transferred to the local system.

The structure is filled out as follows:

doctype Document Type. Specifies the type of file to be transfered. If the document type is specified, the function uses the corresponding shadow file template. If the document type is not specified, the document type is determined by the document type defined in the shadow file template specified by the *shfname* parameter.

`FTMDTFT1` FTAM-1 (unstructured text file)

`FTMDTFT2` FTAM-2 (sequential text file)

`FTMDTFT3` FTAM-3 (unstructured binary file)

`FTMDTIN1` INTAP-1 (INTAP record file)

`FTMDTUNO` Unknown document type

mode Extend Mode. It is set to one of the following values:

`FTMTMEXT` The content of the local file will be extended with the content of the remote file.

`FTMTMREP` The content of the local file will be replaced by the content of the remote file.

accnt Account Name. Specifies the name of the account to which access to the remote file is charged. Only significant to the remote system. May be null.

locname Local File Name (null terminated string). Specifies a local file into which the contents of the remote file will be written. If null, the listing will be displayed on the standard output device. The syntax must obey the rules imposed by the local system.

remname Remote File Name (null terminated string). Specifies the name of the remote file to be transfered. The syntax must obey the rules imposed by the remote system. The complete path name is a concatenation of the internal variable set using the function `fti_rcd` and the filename specified by this parameter.

filpssd Password. Specifies a password, if one is required by the remote system.The password format is: <*type*><*length*><*value*>. The <*length*> may be zero and the <*type*> takes one of the following values:

`FTIGraStr` graphic string

`FTIOctStr` general string

recmode Recovery Mode. Significant only if recovery is enabled. Used to activate the chackpoint mechanisms for a single transfer. Must be set to one of the following values:

`FTIRMRON` Recovery mode `ON`.

`FTIRMNOR` Recovery mode `OFF`.

trfid Transfer Identifier. Significant only when recovery is enabled. Must be set to the value returned previously by the function `fti_getid`. Unpredicatble results can occur if *trfid* is not initialized correctly.

The following fields are returned; the remaining fields are not used:

result number of bytes received

elapse elapsed time (in seconds) for transfer

speed transfer speed (in Kbytes/second)

`fmode` Execution Mode. Must be set to `FTIEXBLK` (blocking mode).

`fesrp` Pointer to an area to receive the Execution Status Report. See "Initiator Execution Status Report" on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERIPA` invalid parameter

`FTIERCNE` connection not established

`FTIERNRH` no response from host

`FTIERLFR` local file name required

`FTIERRFR` remote file name required

`FTIERIDT` invalid document type

`FTIERCDK` error creating docket

`FTIEROLF` error opening local file

`FTIERORF` error opening remote file

`FTIERWDK` error writing docket

`FTIERRRF` error receiving remote file

`FTIERWLF` error writing local file

`FTIERCLF` error closing local file

`FTIERCRF` error closing remote file

`fti_getid()`—*get transfer identifier*

The function `fti_getid` is used to allocate a unique transfer identifier to the transaction. Used for recovery puposes. It is available only if the recovery functional unit is supported.

The function `fti_getid` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_getid(flaip,trfidp,fesrp)
ftilaiT   *flaip;
int       *trfidp;
ftiesrT   *fesrp;
```

`flaip` Local Association Identifier. This input parameter identifies the FTAM association. Its value is recovered by the function `fti_connect` when the associaton is first established.

`trfidp` Transfer Identifier. Output parameter used to allocate a unique transfer identfier that will be used to recover an interrupted transfer if an error occurs.

`fesrp` Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERNSU` function not supported on local system

`FTIERIPA` invalid parameter

`FTIERCNE` connection not established

`FTIERGID` error getting transfer identifier

`fti_init()`—*initialize initiator process*

The function `fti_init` is used to initialize some of the environment parameters associated with the initiator API. It must be called before any of the other initiator functions.

The function `fti_init` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_init(fadmp,fesrp)
ftiadmT   *fadmp;
ftiesrT   *fesrp;
```

`fadmp` Pointer to structure of type `ftiadmT`. This structure is used to set some of the environment parameters associated with the initiator API.

The structure is filled out as follows:

ftiinid Initialization Directory (null terminated string). Set to `/var/SUNWconn/OSIROOT/ftam/conf` by default.

ftilogd Log Files Directory (null terminated string). Set to `/var/SUNWconn/OSIROOT/ftam/log` by default.

ftishfd Shadow Files Directory (null terminated string). Set to `/var/SUNWconn/OSIROOT/ftam/shf` by default.

ftirdbd Remote Systems Database Directory (null terminated string). Set to `/var/SUNWconn/OSIROOT/ftam/rdb` by default.

ftitrfd Transfers Database Directory (null terminated string). Set to `/var/SUNWconn/OSIROOT/ftam/doc` by default.

The following field is returned by the initiator process:

log Log File Identifier. Returned if the file `fti`<*log*>`.log` has been created in the log files directory.

`fesrp` Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8.

`fti_lcd()` continued...

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERIPA` invalid parameter

`FTIWRINI` initialization warning

`fti_lcd()`—*change local working directory*

The function `fti_lcd` is used to change the current working directory on the local system. All subsequent local files names are relative to the new directory.

The function `fti_lcd` returns zero if successful and a non-zero value if not.

```
int fti_lcd(flaip,flcdp,fmode,fesrp)
ftilaiT   *flaip;
ftilcdT   *flcdp;
int        fmode;
ftiesrT   *fesrp;
```

`flaip` Local Association Identifier. This input parameter identifies the FTAM association. Its value is recovered by the function `fti_connect` when the associaton is first established.

`flcdp` Pointer to a structure of type `ftilcdT`. The structure is filled out as follows:

dirname Local Directory Name (null terminated string).

`fmode` Execution Mode. Must be set to `FTIEXBLK` (blocking mode).

`fesrp` Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8. The status cannot be `FTISTFTM`.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERIPA` invalid parameter

`FTIERIPN` invalid path name

`fti_put()`—*put local file*

The function `fti_put` is used to transfer a file from the local system to a remote system with which an association has been established using the function `fti_connect`. If the remote file exists, the action is determined by the *override* and *mode* parameters.

The function `fti_put` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_put(flaip,ftrfp,fmode,fesrp)
ftilaiT   *flaip;
ftitrfTi  *ftrfp;
int        fmode;
ftiesrT   *fesrp;
```

`flaip` Local Association Identifier. This input parameter identifies the FTAM association. Its value is recovered by the function `fti_connect` when the associaton is first established.

`ftrfp` Pointer to a structure of type `ftitrfT`. Identifies the remote file to be transferred to the local system.

The structure is filled out as follows:

doctype Document Type. Specifies the type of file to be transfered. If the document type is specified, the function uses the corresponding shadow file template. If the document type is not specified, the document type is determined by the document type defined in the shadow file template specified by the *shfname* parameter.

`FTMDTFT1` FTAM-1 (unstructured text file)

`FTMDTFT2` FTAM-2 (sequential text file)

`FTMDTFT3` FTAM-3 (unstructured binary file)

`FTMDTIN1` INTAP-1 (INTAP record file)

`FTMDTUNO` Unknown document type

`fti_put()` continued...

override Override Parameter. Specifies the action to be taken if the remote file exists.

`FTMOVNOC` No creation. Transaction fails.

`FTMOVDOA` Creation with old attributes.

`FTMOVDNA` Creation with new attributes.

`FTMOVSOF` Select old file.

mode Extend Mode. It is set to one of the following values:

`FTMTMEXT` The content of the local file will be extended with the content of the remote file.

`FTMTMREP` The content of the local file will be replaced by the content of the remote file.

accnt Account Name. Specifies the name of the account to which access to the remote file is charged. Only significant to the remote system. May be null.

locname Local File Name (null terminated string). Specifies a local file to be transfered to the remote system. The syntax must obey the rules imposed by the local system.

remname Remote File Name (null terminated string). Specifies the name of the remote file. The syntax must obey the rules imposed by the remote system.

filpssd Password. Specifies a password, if one is required by the remote system.The password format is: <*type*><*length*><*value*>. The <*length*> may be zero and the <*type*> takes one of the following values:

`FTIGraStr` graphic string

`FTIOctStr` general string

`fti_put()` continued...

recmode Recovery Mode. Significant only if recovery is enabled. Used to activate the chackpoint mechanisms for a single transfer. Must be set to one of the following values:

`FTIRMRON` Recovery mode `ON`.

`FTIRMNOR` Recovery mode `OFF`.

filsize Size of the new file in bytes. The `-z` option is only valid if it is supported by the remote FTAM responder (for example, the SunLink FTAM local responder does not recognize this option).

trfid Transfer Identifier. Significant only when recovery is enabled. Must be set to the value returned previously by the function `fti_getid`. Unpredicatble results can occur if *trfid* is not initialized correctly.

The following fields are returned; the remaining fields are not used:

result number of bytes received

elapse elapsed time (in seconds) for transfer

speed transfer speed (in Kbytes/second)

`fmode` Execution Mode. Must be set to `FTIEXBLK` (blocking mode).

`fesrp` Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERIPA` invalid parameter

`FTIERCNE` connection not established

`FTIERNRH` no response from host

`FTIERLFR` local file name required

`FTIERRFR` remote file name required

FTIERIDT	invalid document type
FTIERRDS	error reading default shadow file
FTIEROLF	error opening local file
FTIERORF	error opening remote file
FTIERRLF	error reading local file

fti_pwd()—*return remote working directory*

The function fti_pwd is used to return the current working directory on the remote system. It returns zero if successful and a non-zero value if unsuccessful.

```
int fti_pwd(flaip,fpwdp,fmode,fesrp)
ftilaiT   *flaip;
ftipwdT   *fpwdp;
int        fmode;
ftiesrT   *fesrp;
```

flaip — Local Association Identifier. This input parameter identifies the FTAM association. Its value is recovered by the function fti_connect when the associaton is first established.

fpwdp — Pointer to a structure of type ftipwdT. The structure is filled out as follows:

dirname	Remote Directory Name (null terminated string).

fmode — Execution Mode. Must be set to FTIEXBLK (blocking mode).

fesrp — Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

FTIERIPA	invalid parameter

`fti_ratt()`—*read remote attributes*

The function `fti_ratt` is used to read the attributes of a file on a remote system with which an association has been established using the function `fti_connect`.

The function `fti_ratt` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_ratt(flaip,fratp,fmode,frarp,fesrp)
ftilaiT   *flaip;
ftiratT   *fratp;
int        fmode;
ftiesrT   *fesrp;
```

`flaip` Local Association Identifier. This input parameter identifies the FTAM association. Its value is recovered by the function `fti_connect` when the associaton is first established.

`fratp` Pointer to a structure of type `ftiratT`. Identifies the remote file to be transferred.

The structure is filled out as follows:

accnt Account Name. Specifies the name of the account to which access to the remote file is charged. Only significant to the remote system. May be null.

filname Remote File Name (null terminated string). Specifies a remote file for which the attributes will be read. The syntax must obey the rules imposed by the remote system.

filpssd Password. Specifies a password, if one is required by the remote system.The password format is: <*type*><*length*><*value*>. The <*length*> may be zero and the <*type*> takes one of the following values:

`FTIGraStr` graphic string

`FTIOctStr` general string

reqattri Requested Attributes (three-byte array).

The first byte (index 0) is a logical OR of the following values:

FTMAFNAME file name

FTMAPERAC permitted actions

FTMACOTYP content type

FTMASTACC storage account (*)

FTMADTCRE date and time created (*)

FTMADTLMO date and time of last modification

FTMADTLRA date and time of last read access

FTMADTLAM date and time of last attribute modification

The second byte (index 1) is a logical OR of the following values:

FTMAIDLCR identity of creator

FTMAIDLMO identity of last modifier (*)

FTMAIDLRE identity of last reader (*)

FTMAIDLAM identity of last attribute modifier (*)

FTMAFAVAI file availability (*)

FTMAFSIZE file size

FTMAFFSIZ future file size (*)

FTMAACCTL access control (*)

The third byte (index 2) is a logical OR of the following values:

FTMALQUAL legal qualification (*)

FTMAPRUSE private use (*)

The remaining fields are not used.

`fti_ratt()` continued...

(*) indicates that the attribute is not available or not supported on UNIX systems.

`fmode` Execution Mode. Must be set to `FTIEXBLK` (blocking mode).

The following fields are returned; the remaining fields are not used:

result number of bytes received

elapse elapsed time (in seconds) for transfer

speed transfer speed (in Kbytes/second)

`fmode` Execution Mode. Must be set to `FTIEXBLK` (blocking mode).

`frarp` Pointer to a structure of type `ftirasT`. This output parameter will contain the *read attribute response* of the `read-attribute` operation. This structure is described in the C header file `ftiuser.h`.

special formats

<*length*><*type*><*value*>

<*length*> specifies the length of the <*value*> field (one octet)

<*type*> specifies whether the value of the attribute is available or not (one octet):

0 no value available

1 value not supported

2 value available

<*value*> value field of length <*length*>

dates and times

<*length*><*avail*><*year*><*month*><*day*><*hour*><*minutes*>

<*length*> length of value (one octet)

<*avail*> availability of value (one octet)

<*year*> year (zero to four octets)

`fti_ratt()` continued...

<*month*> month (zero to two octets)

<*day*> day (zero to two octets)

<*hour*> hour (zero to two octets)

<*minutes*> minutes (zero to two octets)

identities

<*length*><*avail*><*identity*>

<*length*> length of value (one octet)

<*avail*> availability of value (one octet)

<*identity*> value of length <*length*>

`fesrp` Pointer to an area to receive the Execution Status Report. See "Initiator Execution Status Report" on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERIPA` invalid parameter

`FTIERCNE` connection not established

`FTIERNRH` no response from host

`FTIERRFR` remote file name required

`FTIERFNA` function not available on this system

`FTIERDSF` error deselecting remote file

`fti_rcd()`—*change remote working directory*

The function `fti_rcd` is used to change the current working directory on the remote system. All subsequent remote files names are relative to the new directory.

The function `fti_rcd` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_rcd(flaip,frcdp,fmode,fesrp)
ftilaiT   *flaip;
ftircdT   *frcdp;
int        fmode;
ftiesrT   *fesrp;
```

`flaip` Local Association Identifier. This input parameter identifies the FTAM association. Its value is recovered by the function `fti_connect` when the associaton is first established.

`frcdp` Pointer to a structure of type `ftircdT`. The structure is filled out as follows:

dirname Local Directory Name (null terminated string).

`fmode` Execution Mode. Must be set to `FTIEXBLK` (blocking mode).

`fesrp` Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8. The status cannot be `FTISTFTM`.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERIPA` invalid parameter

`fti_read()`—*read remote file*

The function `fti_read` is used to read all or part of a file from a remote system with which an association has been established using the function `fti_connect`. It is only available if the file access functional unit is supported.

Because the function `fti_read` is typically used for small file transfers, the recovery mode is automatically set to `OFF`.

The function `fti_read` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_read(flaip,ftrfp,fmode,fesrp)
ftilaiT   *flaip;
ftitrfT   *ftrfp;
int        fmode;
ftiesrT   *fesrp;
```

`flaip` Local Association Identifier. This input parameter identifies the FTAM association. Its value is recovered by the function `fti_connect` when the associaton is first established.

`ftrfp` Pointer to a structure of type `ftitrfT`. Identifies the remote file to be transferred to the local system.

The structure is filled out as follows:

doctype Document Type. Specifies the type of file to be created.

`FTMDTFT2` FTAM-2 (sequential text file)

mode Extend Mode. It is set to one of the following values:

`FTMTMEXT` The content of the local file will be extended with the content of the remote file.

`FTMTMREP` The content of the local file will be replaced by the content of the remote file.

accnt Account Name. Specifies the name of the account to which access to the remote file is charged. Only significant to the remote system. May be null.

`fti_read()` continued...

locname Local File Name (null terminated string). Specifies a local file into which the content of the remote file will be written. The syntax must obey the rules imposed by the local system.

remname Local File Name (null terminated string). Specifies the name of the remote file. The syntax must obey the rules imposed by the remote system. The complete path name is a concatenation of the internal variable set using the function `fti_rcd` and the filename specified by this parameter.

filpssd Password. Specifies a password, if one is required by the remote system.The password format is: <*type*><*length*><*value*>. The <*length*> may be zero and the <*type*> takes one of the following values:

`FTIGraStr` graphic string

`FTIOctStr` general string

frfadu From FADU (integer). Specifies the start of the block to be read. The first FADU (first text line) is number 1.

tofadu To FADU (integer). Specifies the end of the block to be read. The last FADU (last text line) is number -1.

trfid Transfer Identifier. Significant only when recovery is enabled. Must be set to the value returned previously by the function `fti_getid`. Unpredicatble results can occur if *trfid* is not initialized correctly.

The following fields are returned; the remaining fields are not used:

result number of bytes received

elapse elapsed time (in seconds) for transfer

speed transfer speed (in Kbytes/second)

`fmode` Execution Mode. Must be set to `FTIEXBLK` (blocking mode).

`fesrp` Pointer to an area to receive the Execution Status Report. See "Initiator Execution Status Report" on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERNSU`	function not supported on local system
`FTIERIPA`	invalid parameter
`FTIERCNE`	connection not established
`FTIERNRH`	no response from host
`FTIERRFR`	remote file name required
`FTIERIDT`	invalid document type
`FTIERCDK`	error creating docket
`FTIEROLF`	error opening local file
`FTIERORF`	error opening remote file
`FTIERWDK`	error writing docket
`FTIERRRF`	error receiving remote file
`FTIERWLF`	error writing local file
`FTIERRDF`	error reading local file
`FTIERCLF`	error closing local file
`FTIERCRF`	error closing remote file

`fti_recover()`—*recover interrupted transfer*

The function `fti_recover` is used to recover a file transfer that has been interrupted by an error or by request. It is only available if the recovery functional unit is supported.

The function `fti_recover` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_recover(flaip,docketp,fmode,fesrp)
ftilaiT   *flaip;
ftidockT  *docketp;
int        fmode;
ftiesrT   *fesrp;
```

`fti_recover()` continued...

`flaip` Local Association Identifier. This input parameter identifies the FTAM association. Its value is recovered by the function `fti_connect` when the associaton is first established.

`docketp` Pointer to a structure of type `ftidockT`. Identifies the interrupted file transfer to be recovered.

The structure is filled out as follows:

id Transfer Identifier. Specifies the transfer to be recovered.

The following fields are returned; the remaining fields are not used:

result number of bytes received

elapse elapsed time (in seconds) for transfer

speed transfer speed (in Kbytes/second)

`fmode` Execution Mode. Must be set to `FTIEXBLK` (blocking mode).

`fesrp` Pointer to an area to receive the Execution Status Report. See "Initiator Execution Status Report" on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERNSU` function not supported on local system

`FTIERIPA` invalid parameter

`FTIERCNE` connection not established

`FTIERNRH` no response from host

`FTIERFNA` function not available on this host

`FTIERRDK` error reading docket

`FTIERWDK` error writing docket

`FTIEROLF` error opening local file

`FTIERWLF` error writing local file

`fti_recover()` continued...

`FTIERRDF`	error reading local file
`FTIERRTR`	error recovering transfer
`FTIERSLF`	error sending local file
`FTIERRRF`	error receiving remote file
`FTIERCLF`	error closing local file
`FTIERCRF`	error closing remote file

`fti_rename()`—*rename remote file*

The function `fti_rename` is used to rename a file on a remote system with which an association has been established using the function `fti_connect`. If a file named *newname* exists the file is not overwritten and the operation fails.

The function `fti_rename` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_rename(flaip,frenp,fmode,fesrp)
ftilaiT   *flaip;
ftirenT   *frenp;
int        fmode;
ftiesrT   *fesrp;
```

`flaip` Local Association Identifier. This input parameter identifies the FTAM association. Its value is recovered by the function `fti_connect` when the associaton is first established.

`frenp` Pointer to a structure of type `ftirenT`. Identifies the remote file to be transferred to the local system.

accnt	Account Name. Specifies the name of the account to which access to the remote file is charged. Only significant to the remote system. May be null.
filname	Old File Name (null terminated string). Specifies a remote file to be renamed. The syntax must obey the rules imposed by the remote system.

`fti_rename()` continued...

filpssd Password. Specifies a password, if one is required by the remote system.The password format is: <*type*><*length*><*value*>. The <*length*> may be zero and the <*type*> takes one of the following values:

`FTIGraStr` graphic string

`FTIOctStr` general string

newname New File Name (null terminated string). Specifies a new name for the remote file. The syntax must obey the rules imposed by the remote system. The full name is a concatenation of the internal variable set using the function `fti_rcd` and the filename specified by this parameter.

`fmode` Execution Mode. Must be set to `FTIEXBLK` (blocking mode).

`fesrp` Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERIPA` invalid parameter

`FTIERCNE` connection not established

`FTIERNRH` no response from host

`FTIERRFR` remote file name required

`FTIERSRF` error selecting remote file

`FTIERDSF` error deselecting remote file

`FTIERRNF` error renaming remote file

FTAM Remote Database (RDB) Interface Definitions

The **FTAM Remote Database (RDB) interface** provides a set of functions that are used to access the FTAM services which handle the OSI addresses required to establish an associations with remote hosts.

`fti_rdbdel()`—*delete entry from RDB*

The function `fti_rdbdel` is used to delete an entry from the remote systems database that contains a list of the remote systems recognized by the FTAM initiator application.

The function `fti_rdbdel` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_rdbdel(frhnp,localonly,fesrp)
X         *frhnp;
int        localonly;
ftiesrT   *fesrp;
```

`frhnp` Remote Host Name (null terminated string). Specifies the entry to be deleted. Maximum number of characters is given by `FTIMXHRN`.

`localonly` Must be set to 1.

`fesrp` Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERIPA` invalid parameter

`FTIERUDN` unknown host name

`FTIERUDH` unable to delete host

`fti_rdbget()`—*get entry from RDB*

The function `fti_rdbget` is used to read the OSI address elements for an entry in the remote systems database that contains a list of the remote systems recognized by the FTAM initiator application.

The function `fti_rdbget` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_rdbget(frdbp,localonly,fesrp)
ftirdbT   *frdbp;
intr       localonly;
ftiesrT   *fesrp;
```

`frdbp` Pointer to a structure of type `ftirdbT`. The host name must be provided by the user in the *hostname* field. The structure is filled out as follows:

- *pstlen* Presentation Selector Length. When set to 0 (zero) the presentation selector should be ignored.
- *pstsel* Presentation Selector Value.
- *seslen* Session Selector Length. When set to 0 (zero) the session selector should be ignored.
- *sessel* Session Selector Value.
- *trslen* Transport Selector Length. When set to 0 (zero) the transport selector should be ignored.
- *trssel* Transport Selector Value.
- *netlen* Network Address Length. When set to 0 (zero) the network address should be ignored.
- *netsel* Network Address Value.
- *aetlen* Application Entity Title (AET) Length. When set to 0 (zero) the presentation selector should be ignored.
- *pstsel* Application Entity Title (AET) Value.
- *subnet* Subnetwork Number.

state Current State. Specifies whether the remote host can be reached or not. It is set to one of the following values:

`FTIHSOPE` remote host is `OPEN`

`FTIHSCLO` remote host is `CLOSED`

`localonly` Must be set to 1.

`fesrp` Pointer to an area to receive the Execution Status Report. See "Initiator Execution Status Report" on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERIPA` invalid parameter

`FTIERUDN` unknown host name

`FTIERRHI` error reading host information

`fti_rdbput()`—*put entry in RDB*

The function `fti_rdbget` is used to add an entry in the remote systems database that contains a list of the remote systems recognized by the FTAM initiator application.

The function `fti_rdbput` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_rdbput(frdbp,localonly,fesrp)
ftirdbT   *frdbp;
int        rlocalonly;
ftiesrT   *fesrp;
```

`frdbp` Pointer to a structure of type `ftirdbT`. The host name must be provided by the user in the *hostname* field.

The structure is filled out as follows:

pstlen Presentation Selector Length. When set to 0 (zero) the presentation selector is ignored.

`fti_rdbput()` continued...

pstsel Presentation Selector Value.

seslen Session Selector Length. When set to 0 (zero) the session selector is ignored.

sessel Session Selector Value.

trslen Transport Selector Length. When set to 0 (zero) the transport selector is ignored.

trssel Transport Selector Value.

netlen Network Address Length. When set to 0 (zero) the network address is ignored.

netsel Network Address Value.

aetlen Application Entity Title (AET) Length. When set to 0 (zero) the presentation selector is ignored.

pstsel Application Entity Title (AET) Value.

subnet Subnetwork Number. Takes a value between 0 and `FTIMXSBN-1`

state Current State. Specifies whether the remote host can be reached or not. It is set to one of the following values:

`FTIHSOPE` remote host is `OPEN`

`FTIHSCLO` remote host is `CLOSED`

`localonly` Must be set to 1.

`fesrp` Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERIPA` invalid parameter

`FTIERUDN` unknown host name

`FTIERWHI` error writing host information

FTAM Shadow File (SHF) Interface Definitions

The **FTAM Shadow File (SHF) interface** provides a set of functions that are used to access the FTAM shadow file services which handle the virtual file attributes that are not supported by the local file system—for example, document type or universal class number.

`fti_shfget()`—*get attributes from local shadow file*

The function `fti_shfget` is used to read the attributes from a shadow file associated with a local file.

The function `fti_shfget` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_shfget(filename,fshfp,fesrp)
char      *filename;
ftishfT   *fshfp;
ftiesrT   *fesrp;
```

`filname` Pointer to the local file name, a null terminated string of `FTIMXSTR` characters maximum length.

`fshfp` Pointer to a structure of type `ftishfT`. Returns the attributes recovered from the shadow file.

The structure is filled out as follows:

doctype Document Type. Specifies the type of file. It is set to one of the following values:

`FTMDTFT1` FTAM-1 (unstructured text file)

`FTMDTFT2` FTAM-2 (sequential text file)

`FTMDTFT3` FTAM-3 (unstructured binary file)

`FTMDTIN1` INTAP-1 (INTAP record file)

`fti_shfget()` continued...

peract Permitted Actions. This parameter consists of two bytes. The second byte is reserved for future extensions. The first byte is the result of a logical `OR` of the following values:

`FTMAREA`	read permission
`FTMAINS`	insert permission
`FTMAEXT`	extend permission
`FTMAREP`	replace permission
`FTMAERA`	erase permission
`FTMADEL`	delete permission
`FTMARAT`	read attributes permission
`FTMACAT`	change attributes permission

keypos Not Used

parm.choice Not Used

parm.class Universal Class Number. It is set to one of the following values:

`FTMUCPRT`	printable string
`FTMUCTXT`	teletext string
`FTMUCVXT`	videotext string
`FTMUCIA5`	IA5 string
`FTMUCGRP`	graphic string
`FTMUCVIS`	visible string
`FTMUCGEN`	general string

For FTAM-3 files, the only possible value is:

`FTMUCOCT`	octet string

parm.length1 Maximum String Length.

parm.length2 String Significance. It is set to one of the following values:

`FTMPDFIX` fixed

`FTMPDVAR` variable

`FTMPDNOT` not significant

`fesrp` Pointer to an area to receive the Execution Status Report. See "Initiator Execution Status Report" on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERIPA` invalid parameter

`FTIERRSF` error reading shadow file

`fti_shfput()`—*put attributes in local shadow file*

The function `fti_shfput` is used to create a new shadow file or modify an existing shadow file.

The function `fti_shfput` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_shfput(filename,fshfp,fesrp)
char      *filename;
ftishfT   *fshfp;
ftiesrT   *fesrp;
```

`filname` Pointer to the local file name, a null terminated string of `FTIMXSTR` characters maximum length.

`fshfp` Pointer to a structure of type `ftishfT`. Defines the attributes to be put in the shadow file. The structure is filled out as follows:

doctype Document Type. Specifies the type of file. It is set to one of the following values:

`fti_shfput()` continued...

	`FTMDTFT1`	FTAM-1 (unstructured text file)
	`FTMDTFT2`	FTAM-2 (sequential text file)
	`FTMDTFT3`	FTAM-3 (unstructured binary file)
	`FTMDTIN1`	INTAP-1 (INTAP record file)
peract	Permitted Actions. This parameter consists of two bytes. The second byte is reserved for future extensions. The first byte is the result of a logical `OR` of the following values:	
	`FTMAREA`	read permission
	`FTMAINS`	insert permission
	`FTMAEXT`	extend permission
	`FTMAREP`	replace permission
	`FTMAERA`	erase permission
	`FTMADEL`	delete permission
	`FTMARAT`	read attributes permission
	`FTMACAT`	change attributes permission
keypos	Not Used	
parm.choice	Not Used	
parm.class	Universal Class Number. It is set to one of the following values:	
	`FTMUCPRT`	printable string
	`FTMUCTXT`	teletext string
	`FTMUCVXT`	videotext string
	`FTMUCIA5`	IA5 string
	`FTMUCGRP`	graphic string

FTMUCVIS visible string

FTMUCGEN general string

For FTAM-3 files, the only possible value is:

FTMUCOCT octet string

parm.length1 Maximum String Length.

parm.length2 String Significance. It is set to one of the following values:

FTMPDFIX fixed

FTMPDVAR variable

FTMPDNOT not significant

`fesrp` Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

FTIERIPA invalid parameter

FTIERWSF error writing shadow file

2

FTAM Transfer (TRF) Interface Definitions

The **FTAM Transfer (TRF) interface** provides a set of functions that are used to access the FTAM file recovery services which handle the recovery of interrupted file transfers based on information contained in special FTAM files called *dockets*.

`fti_trf_del()`—*delete docket information*

The function `fti_trf_del` is used to delete the recovery information (docket) associated with an interrupted transfer. It is only available if the recovery functional unit is supported.

The function `fti_trf_del` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_trf_del(trfid,fesrp)
int        trfid;
ftiesrT   *fesrp;
```

`trfid` Transfer Identifier. Identifies the interrupted transfer for which the associated recovery information is to be deleted.

`fesrp` Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERNSU` function not supported on local system

`FTIERIPA` invalid parameter

`FTIERDDK` error deleting docket

`fti_trf_list()`—*list transfer identifiers*

The function `fti_trf_list` is used to list the transfer identifiers for the current interrupted transfers. It is only available if the recovery functional unit is supported.

The function `fti_trf_list` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_trf_list(trfidp,morep,fesrp)
int       *trfidp;
char      *morep;
ftiesrT   *fesrp;
```

`trfidp` Transfer Identifier. The value of this field is significant only if `fti_trf_list` returns successfully.

`morep` More Indicator. If the value of this field is `FTIMORE`, then another call to `fti_trf_list` will return the next available transfer identifier.

`fesrp` Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERNSU` function not supported on local system

`FTIERIPA` invalid parameter

`FTIWRTRL` transfer list warning

`fti_trf_show()`—*show docket information*

The function `fti_trf_show` is used to read the recovery information (docket) associated with an interrupted transfer. It is only available if the recovery functional unit is supported.

The function `fti_trf_show` returns zero if successful and a non-zero value if unsuccessful.

```
int fti_trf_show(trfid,docketp,fesrp)
int        trfid;
ftidockT  *docketp;
ftiesrT   *fesrp;
```

`trfid` Transfer Identifier. Identifies the interrupted transfer for which the associated recovery information is to be read.

`docketp` Pointer to a structure of type `ftidocT`. Returns the information read from the docket. The structure is filled out as follows:

ir Docket Owner. The docket may be owned by the initiator (`i`) or the responder (`r`).

id Transfer Identifier.

hostname Remote Host Name (null terminated string).

username User Name (null terminated string).

locname Local File Name (null terminated string).

cpi Last Checkpoint received.

acpi Last Checkpoint Acknowledged.

act_state Activity State. It can take the following values:

`0` data transfer starting

`2` data transfer in progress

`3` data transfer finishing

`4` data transfer finished

doctype Document Type. Specifies the type of file. It is set to one of the following values:

`FTMDTFT1` FTAM-1 (unstructured text file)

`FTMDTFT2` FTAM-2 (sequential text file)

`FTMDTFT3` FTAM-3 (unstructured binary file)

`FTMDTIN1` INTAP-1 (INTAP record file)

The remianing fields are reserved for future extensions.

`fesrp` Pointer to an area to receive the Execution Status Report. See “Initiator Execution Status Report” on page 8.

The following error codes are returned in the Execution Status Report as appropriate:

`FTIERNSU` function not supported on local system

`FTIERIPA` invalid parameter

2

Initiator Error and Status Codes

API Event Codes

```
FTINOEVT     0  FTI NO EVenT received
FTIEVABO     1  FTI EVent code, ABOrt
FTIEVOPE     2  FTI EVent code, OPEn
FTIEVCLO     3  FTI EVent code, CLOse
FTIEVPUT     4  FTI EVent code, PUT
FTIEVGET     5  FTI EVent code, GET
FTIEVCRE     6  FTI EVent code, CREate
FTIEVDEL     7  FTI EVent code, DELete
FTIEVREN     8  FTI EVent code, REName
FTIEVRAT     9  FTI EVent code, Read ATtribute
```

Execution Status

```
FTISTNER     0  No ERror
FTISTLOC     1  LOCal or user error
FTISTSYS     2  local SYStem error
FTISTFTM     3  FTaM service or protocol error
FTISTUFE     4  Unexpected Ftam event
FTISTWAR     5  local WARning
```

Error Types

```
FTITYNRC     0  Not ReCoverable error
FTITYRCO     1  RECoverable error
```

Error Connexion Status

```
FTICNXON     1  CoNneXion ON
FTICNXOF     0  CoNneXion OFf
```

Error codes

```
FTIERIEV     1  FTI ERror, Incorrect Environment Variabl
FTIERNSU     2  FTI ERror, function Not SUpported
FTIERIPA     3  FTI ERror, Invalid PArameter
             4  Not used
FTIERCNE     5  FTI ERror, Connection Not Established
FTIERCAE     6  FTI ERror, Connection Already Establishe
FTIERLFR     7  FTI ERror, Local file name required
FTIERRFR     8  FTI ERror, Remote file name required
FTIERUHN    11  FTI ERror, Unknown host name
FTIEROLF    12  FTI ERror, Opening Local File
FTIERORF    13  FTI ERror, Opening Remote File
FTIERFCH    14  FTI ERror, Failed to Connect to remote Host
FTIERCLF    15  FTI ERror, Closing Local File
FTIERCRF    16  FTI ERror, Closing Remote File
FTIERSLF    17  FTI ERror, Sending Local File
FTIERRRF    18  FTI ERror, Receiving Remote File
FTIERFNA    19  FTI ERror, Function Not Available
FTIERCRH    23  FTI ERror, Closing Remote Host
FTIERWHI    24  FTI ERror, Writing Host Information
FTIERRSF    25  FTI ERror, Reading Shadow File
FTIERRDS    26  FTI ERror, Reading Default Shadow File
FTIERIPN    29  FTI ERror, Invalid Path Name
FTIERRHI    30  FTI ERror, Reading Host Information
FTIERWSF    33  FTI ERror, Writing Shadow File
FTIERRLF    31  FTI ERror, Reading Local File
FTIERWLF    32  FTI ERror, Writing Local File
FTIERDRF    35  FTI ERror, Deleting Remote File
FTIERCTF    36  FTI ERror, Creating Remote File
FTIERRNF    37  FTI ERror, Renaming Remote File
FTIERRRA    38  FTI ERror, Reading Remote File Attribute
FTIERIDT    40  FTI ERror, Invalid Document Type
FTIERHNR    42  FTI ERror, Host name required
```

```
FTIERUDH    51  FTI ERror, Unable to Delete Host
FTIERHIC    55  FTI ERror, Host is closed
FTIERRDF    63  FTI ERror, Reading Remote File
FTIERSRF    66  FTI ERror, Selecting Remote File
FTIERDSF    67  FTI ERror, Deselecting Remote File
FTIERGID    68  FTI ERror, Getting transfer IDentifier
FTIERRTR    69  FTI ERror, Recovering TRansfer
FTIERCDK    70  FTI ERRor, Creating DocKet
FTIERWDK    71  FTI ERRor, Writing  DocKet
FTIERRDK    72  FTI ERror, Reading  Docket
FTIERDDK    73  FTI ERror, Deleting Docket
```

Local (Non-UNIX) Errors

```
ERRDTYP      1  Illegal document type parameters
ERRMODE      2  Invalid mode in F-Open-ind
ERRSUBSQ     3  Subsequent error
ERRFNAME     4  No file name in F-Create-ind
ERRCFNAME    5  No file name in F-ChAtt-ind
ERRDTPABS    6  No composite attributes in F-Create-ind
ERRIOI       7  Unknown user (F-Init-ind)
ERRUID       8  No user identity or cannot set groupe id
                or user id. or cannot change directory
ERRSRC       9  Invalid service class in F-Init-ind
ERRPSWD     10  Invalid password in F-Init-ind
ERRDTTYP    11  Invalid Data Element type
ERRDTINV    12  Invalid Universal Class
ERRDTSZ     13  Invalid Data Element size
ERRTIMEOUT  14  Time-out
ERRRECLG    15  Invalid INTAP_1 record length
ERRRDEOF    16  Cannot read at end of file
ERRFADUI    17  FADU id. error in F-Read-ind
ERRLFADU    18  Cannot locate FADU
ERRTRFIL    19  Cannot truncate file
```

```
ERRSFNLG    20  File name too long
ERRDCTYP    21  Invalid document type
ERROVERW    22  Invalid overwrite parameter
ERRNODTY    23  No document type in F-Open-cnf
ERRFTYPE    24  Unsupported Special local file type
ERRSTRLG    25  Invalid FTAM_1 or FTAM_2 string length
ERRCHECK    26  Recovery error - Bad checkpoint
ERRDOCK     27  Recovery error - Corrupted docket
ERRNODOCK   28  Recovery error - No docket
ERRRECPT    29  Recovery error - No recovery point
ERRCONTYP   30  Recovery error - Content type inconsist.
ERRUNSPEC   31  Recovery error - Unspecific
```

Warning codes

```
FTIWRINI     1   WaRning on INItialisation
FTIWRTRL    30   WaRning on Transfer List
WARINID    0x000000001   Init directory not found
WARLOGD    0x000000002   Log directory not found
WARSHFD    0x000000004   Shf directory not found
WARRDBD    0x000000008   Rdb directory not found
WARTRFD    0x000000010   Trf directory not found
WARINIF    0x000000020   Init file not found
WARTRFF    0x000000040   Transfer Identifier file not found
WARNODA    0x000000021   No Docket Available
```

2

FTAM Responder Stubs File Definitions 3

This chapter describes the functions contained in the FTAM responder stubs file, which is used to develop FTAM servers similar to the SunLink FTAM application `osiftr`.

Note that throughout this chapter the term *remote system* is used to denote the system on which the FTAM responder is running.

Responder Common Data Structures

All data structures, types, and constants are defined in the C header file `/opt/SUNWconn/include/ftam/ftruser.h`, which is delivered as part of the SunLink FTAM application. A customized FTAM server developed using the FTAM responder stubs file must include this C header file, by using the following compiler directive:

```
#include “ftam/ftruser.h”
```

3

Responder Execution Status Report

Some FTAM responder functions return an Execution Status Report (of type `ftresrT`). This output parameter is used for operations that can be refused by the customized responder application. The responder behavior will be determined according the return of the function call:

- If the function returns successfully, the Execution Status Report will be ignored and responder will complete the operation.
- Otherwise, the Execution Status Report will be used to generate an appropriate diagnostic for the failure, and a negative response will be sent to the remote host.

The structure must be filled by the API user as follows:

`status`: **Execution status**.
This field is set to one of the following values:

`FTRSTNER` Execution was successful. No error.

`FTRSTFTM` Execution was unsuccessful. FTAM service error.

`diag`: **Diagnostic code**.
Gives information about the error. Used by the responder function to generate the appropriate FTAM diagnostic. It must be set to one of the values listed in Chapter 4, "FTAM Diagnostic Codes."

`ftamdfd`: **FTAM diagnostic further details**.
Contains the graphic string referred to in the FTAM protocols as "diagnostic further details".

Responder Stubs File Definitions Summary

The interfaces to the FTAM responder module are summarized in Table 3-1

Table 3-1 FTAM Initiator External Interface Definitions

Function	**Page**
`ftr_connect()`—association connect request	page 66
`ftr_create()`—file create request	page 67
`ftr_delete()`—file delete request	page 69
`ftr_deselect()`—file deselect request	page 69
`ftr_disconnect()`—association disconnect request	page 70
`ftr_ratt()`—read attributes request	page 70
`ftr_read()`—file read request	page 70
`ftr_recover()`—transfer recover request	page 71
`ftr_rename()`—file rename request	page 72
`ftr_select()`—file select request	page 73
`ftr_transfer_end()`—end of transfer indication	page 74
`ftr_write()`—file write request	page 75

3

Responder Stubs File Definitions

The following definitions are used to develop customized FTAM responder processes based on the SunLink FTAM responder (`osiftr`).

`ftr_connect()`—*association connect request*

The function `ftr_connect` is called when the responder receives an association open request from an FTAM initiator. The responder performs several protocol (functional units, service classes) and security checks (user identity, password) before calling `ftr_connect`.

The function `ftr_connect` returns zero if successful and a non-zero value if unsuccessful.

```
int ftr_connect(fbndp,fesrp)
ftrbndT *fbndp;
ftresrT *fesrp;
```

`fbndp` Pointer to a structure of type `ftrbndT`. This input parameter identifies the remote host system which requested the association.

The structure is filled out by the responder as follows:

username User Name. Corresponds to the *initiator identifier* in the `F-Initialize-Indication`.

password User Password. Corresponds to the *filestore password* in the `F-Initialize-Indication`. The password format is: <*type*><*length*><*value*>. The <*length*> may be zero and the <*type*> takes one of the following values:

`FTRgraStr` graphic string

`FTROctStr` octet string

account Account Name. Corresponds to the *account* in the `F-Initialize-Indication`. May take a null value.

`fesrp` Pointer to an area to receive the Execution Status Report. See "Responder Execution Status Report" on page 64.

`ftr_create()`—*file create request*

The function `ftr_create` is called when an FTAM initiator attemptes to create a file on the remote system. The file creation attempt can be refused by the responder.

The function `ftr_create` returns zero if successful and a non-zero value if unsuccessful.

```
int ftr_create(fcrep,fesrp)
ftrcreT *fcrep;
ftresrT *fesrp;
```

`fcrep`: Pointer to a structure of type `ftrcrT` that identifies the local file to be created. The structure is filled out by the reponder as follows:

filname Remote File Name. Specifies the name of the file to be created.

doctype Document Type. Specifies the type of file to be created.

`FTMTFT1` FTAM-1 (unstructured text file)

`FTMTFT2` FTAM-2 (sequential text file)

`FTMTFT3` FTAM-3 (unstructured binary file)

`FTMTIN1` INTAP-1 (INTAP record file)

filsize Specifies the size of the new file in bytes. The `-z` option is only valid if it is supported by the remote FTAM responder (for example, the SunLink FTAM local responder does not recognize this option).

override Override Parameter. Specifies the action to be taken if the file exists.

`FTMOVNOC` No creation. Transaction fails.

`FTMOVDOA` Creation with old attributes.

`FTMOVDNA` Creation with new attributes.

`FTMOVSOF` Select old file.

`ftr_create()` continued...

account Account Name. Specifies the name of the account to which access to the remote file is charged.

peract Permitted Actions. This parameter consists of two bytes. The second byte is reserved for future extensions. The first byte is the result of a logical `OR` of the following values:

- `FTMAREA` read permission
- `FTMAINS` insert permission
- `FTMAEXT` extend permission
- `FTMAREP` replace permission
- `FTMAERA` erase permission
- `FTMADEL` delete permission
- `FTMARAT` read attributes permission
- `FTMACAT` change attributes permission

reqaccess Requested Access. This parameter is the result of a logical OR of the following values:

- `FTMACREA` read access requested
- `FTMACINS` insert access requested
- `FTMACREP` replace access requested
- `FTMACERA` erase access requested
- `FTMACDEL` delete access requested
- `FTMACRAT` read attribute access requested
- `FTMACCAT` change attribute access requested

cfilpssd Create File Password. Reserved for future extensions.

reafilpssd Read File Password. Reserved for future extensions.

infilpssd Insert File Password. Reserved for future extensions.

exfilpssd Extend File Password. Reserved for future extensions.

refilpssd Replace File Password. Reserved for future extensions.

erfilpssd Erase File Password. Reserved for future extensions.

defilpssd Delete File Password. Reserved for future extensions.

rafilpssd Read File Attributes Password. Reserved for future extensions.

cafilpssd Change File Attributes Password. Reserved for future extensions.

`fesrp` Pointer to an area to receive the Execution Status Report. See "Responder Execution Status Report" on page 64.

`ftr_delete()`—*file delete request*

The function `ftr_delete` is called when an FTAM initiator attempts to delete a file on the remote system.

This function is always successful and returns are ignored by the responder.

```
int ftr_delete()
```

`ftr_deselect()`—*file deselect request*

The function `ftr_deselect` is called when an FTAM initiator attempts to deselect a file on the remote system.

This function is always successful and returns are ignored by the responder.

```
int ftr_deselect()
```

ftr_disconnect()—*association disconnect request*

The function `ftr_disconnect` is called when the responder receives an association close request from an FTAM initiator.

This function is always successful and returns are ignored by the responder.

```
int ftr_disconnect(status)
int     status;
```

`status` Association End Status. This paramter is set to the following values:

`FTRASNER` association closed successfully

`FTRASABT` association aborted.

ftr_ratt()—*read attributes request*

The function `ftr_ratt` is called when an FTAM initiator attempts to read the attributes of the selected file on the remote system.

This function is always successful and returns are ignored by the responder.

```
int ftr_ratt()
```

ftr_read()—*file read request*

The function `ftr_read` is called when an FTAM initiator attempts to read data from the selected file on the remote system.

This function is always successful and returns are ignored by the responder.

```
int ftr_read(freap)
ftrtrfT   *freap;
```

`freap` Pointer to a structure of type `ftrtrfT` that identifies the transfer in progress. The structure is filled out by the responder as follows:

trfid local transfer identifier (of the transfer in progress)

ftr_recover()—*transfer recover request*

The function ftr_recover is called when an FTAM initiator attempts to recover an interrupted file transfer. The responder can refuse file recovery.

The function ftr_recover returns zero if successful and a non-zero value if unsuccessful.

```
int ftr_recover(frcop,fesrp)
ftrrcoT *frcop;
ftresrT *fesrp;
```

frcop Pointer to a structure of type ftrrcoT that identifies the interrupted transfer. The structure is filled out by the reponder as follows:

trfid Local Transfer Identifier. Specifies the interrupted file transfer.

filname Remote File Name. Specifies the name of the file being transferred.

doctype Document Type. Specifies the type of file being transferred

FTMTFT1 FTAM-1 (unstructured text file)

FTMTFT2 FTAM-2 (sequential text file)

FTMTFT3 FTAM-3 (unstructured binary file)

FTMTIN1 INTAP-1 (INTAP record file)

FTMTNB9 NBS-9 (file directory listing)

account Account Name. Specifies the name of the account to which access to the remote file is to be charged.

reqaccess Requested Access. This parameter is the result of a logical OR of the following values:

FTMACREA read access requested

FTMACINS insert access requested

FTMACREP replace access requested

`ftr_recover()` continued...

	`FTMACERA`	erase access requested
	`FTMACDEL`	delete access requested
	`FTMACRAT`	read attribute access requested
	`FTMACCAT`	change attribute access requested
reafilpssd		Read File Password. Reserved for future extensions.
infilpssd		Insert File Password. Reserved for future extensions.
exfilpssd		Extend File Password. Reserved for future extensions.
refilpssd		Replace File Password. Reserved for future extensions.
erfilpssd		Erase File Password. Reserved for future extensions.
defilpssd		Delete File Password. Reserved for future extensions.
rafilpssd		Read File Attributes Password. Reserved for future extensions.
cafilpssd		Change File Attributes Password. Reserved for future extensions.
`fesrp`		Pointer to an area to receive the Execution Status Report. See "Responder Execution Status Report" on page 64.

`ftr_rename()`—*file rename request*

The function `ftr_rename` is called when an FTAM initiator attempts to rename the selected file on the remote system.

This function is always successful and returns are ignored by the responder.

```
int ftr_rename()
```

`ftr_select()`—*file select request*

The function `ftr_select` is called when an FTAM initiator attempts to select a file on the remote system.

The function `ftr_select` returns zero if successful and a non-zero value if unsuccessful.

```
int ftr_select(fselp,fesrp)
ftrselT *fselp;
ftresrT *fesrp;
```

`fselp`: Pointer to a structure of type `ftrselT` that identifies the file to be selected. The structure is filled out by the reponder as follows:

filname Remote File Name. Specifies the name of the file to be selected.

doctype Document Type. Specifies the type of file to be selected.

`FTMTFT1` FTAM-1 (unstructured text file)

`FTMTFT2` FTAM-2 (sequential text file)

`FTMTFT3` FTAM-3 (unstructured binary file)

`FTMTIN1` INTAP-1 (INTAP record file)

`FTMTNB9` NBS-9 (file directory listing)

account Account Name. Specifies the name of the account to which access to the remote file is to be charged.

reqaccess Requested Access. This parameter is the result of a logical OR of the following values:

`FTMACREA` read access requested

`FTMACINS` insert access requested

`FTMACREP` replace access requested

`FTMACERA` erase access requested

`FTMACDEL` delete access requested

`ftr_select()` continued...

	`FTMACRAT`	read attribute access requested
	`FTMACCAT`	change attribute access requested
reafilpssd		Read File Password. Reserved for future use.
infilpssd		Insert File Password. Reserved for future use.
exfilpssd		Extend File Password. Reserved for future use.
refilpssd		Replace File Password. Reserved for future use.
erfilpssd		Erase File Password. Reserved for future use.
defilpssd		Delete File Password. Reserved for future use.
rafilpssd		Read File Attributes Password. Reserved for future use.
cafilpssd		Change File Attributes Password. Reserved for future use.

`fesrp` Pointer to an area to receive the Execution Status Report. See "Responder Execution Status Report" on page 64.

`ftr_transfer_end()`—*end of transfer indication*

The function `ftr_transfer_end` is called when the responder is notified that the transfer has ended.

This function is always successful and returns are ignored by the responder.

```
int ftr_transfer_end(fstat,status)
long      fstat;
int       status;
```

`fstat` Number of bytes actually transmitted.

`status` Transfer End Status. This parameter is set to one of the following:

`FTRTRNER`	transfer ended successfully.
`FTRTRCAN`	transfer cancelled

`ftr_write()`—*file write request*

The function `ftr_write` is called when an FTAM initiator attempts to write data into the selected file on the remote system.

This function is always successful and returns are ignored by the responder.

```
int ftr_write(fwrip)
ftrtrfT   *fwrip;
```

`fwrip` Pointer to a structure of type `ftrtrfT` that identifies the transfer in progress. The structure is filled out by the responder as follows:

trfid local transfer identifier (of the transfer in progress)

3

FTAM Diagnostic Codes 4

This chapter lists the diagnostic codes returned by the functions used to develop FTAM client and FTAM server applications.

4

General FTAM Diagnostics

Codes `0000` to `0011` are used to return general diagnostic information as shown in Table 4-1:

Table 4-1 General FTAM Diagnostics

0000	No reason
0001	Responder error (unspecific)
0002	System shutdown
0003	FTAM management problem (unspecific)
0004	FTAM management, bad account
0005	FTAM management, security not passed
0006	Delay may be encountered
0007	Initiator error (unspecific)
0008	Subsequent error
0009	Temporal insufficiency of resources
0010	Access request violates VFS security
0011	Access request violates local security

Protocol and Supporting Service Diagnostics

Codes `1000` to `1017` are used to return diagnostic information regarding FTAM protocols and services as shown in Table 4-2:

Table 4-2 Protocol and Supporting Service Diagnostics

1000	Conflicting parameter value
1001	Unsupported parameter value
1002	Mandatory parameter not set
1003	Unsupported parameter
1004	Duplicated parameter
1005	Illegal parameter type
1006	Unsupported parameter type
1007	FTAM protocol error (unspecific)
1008	FTAM protocol error, procedure error
1009	FTAM protocol error, functional unit error
1010	FTAM protocol error, corruption error
1011	Lower layer failure
1012	Lower layer addressing error
1013	Timeout
1014	System shutdown
1015	Illegal grouping sequence
1016	Grouping threshold violation
1017	Specific PDU request inconsistent with the current requested access

4

Association-Related Diagnostics

Codes `2000` to `2021` are used to return diagnostic information associations between the a client and a server as shown in Table 4-3:

Table 4-3 Association-Related Diagnostics

2000	Association with user not allowed
2001	(not assigned)
2002	Unsupported service class
2003	Unsupported functional unit
2004	Attribute group error (unspecific)
2005	Attribute group not supported
2006	Attribute group not allowed
2007	Bad account
2008	Association management (unspecific)
2009	Association management, bad address
2010	Association management, bad account
2011	Checkpoint window error, too large
2012	Checkpoint window error, too small
2013	Checkpoint window error, unsupported
2014	Communications QoS nor supported
2015	Initiator identity unacceptable
2016	Context management refused
2017	Rollback not available
2018	Contents type list cut by responder
2019	Contents type list cut by Presentation service
2020	Invalid filestore password
2021	Incompatible service class

Selection Regime-Related Diagnostics

Codes `3000` to `3030` are used to return diagnostic information regarding the file selection regime as shown in Table 4-4:

Table 4-4 Selection Regime-Related Diagnostics

3000	Filename not found
3001	Selection attributes not matched
3002	Initial attributes not possible
3003	Bad attribute name
3004	Non-existent file
3005	File already exists
3006	File cannot be created
3007	File cannot be deleted
3008	Concurrency control not available
3009	Concurrency control not supported
3010	Concurrency control not possible
3011	More restrictive lock
3012	File busy
3013	File not available
3014	Access control not available
3015	Access control not supported
3016	Access control inconsistent
3017	Filename truncated
3018	Initial attributes altered
3019	Bad account
3020	Override selected existing file
3021	Override deleted and recreated file with old attributes
3022	Override deleted and recreated file with new attributes
3023	Create override, not possible
3024	Ambiguous file specification

Table 4-4 Selection Regime-Related Diagnostics

3025	Invalid create password
3026	Invalid delete password on override
3027	Bad attribute value
3028	Requested access violates permitted actions
3029	Functional unit not available for requested access
3030	File created but not selected

File Management-Related Diagnostics

Codes `4000` to `4007` are used to return diagnostic information regarding file management as shown in Table 4-5:

Table 4-5 File Management-Related Diagnostics

4000	Attribute non existent
4001	Attribute cannot be read
4002	Attribute cannot be changed
4003	Attribute not supported
4004	Bad attribute name
4005	Bad attribute value
4006	Attribute partially supported
4007	Additional set attribute value not distinct

4

Access-Related Diagnostics

Codes `5000` to `5041` are used to return diagnostic information regarding file access as shown in Table 4-6:

Table 4-6 Access Related Diagnostics

5000	Bad FADU (unspecific)
5001	Bad FADU - size error
5002	Bad FADU - type error
5003	Bad FADU - poorly specified
5004	Bad FADU - bad location
5005	FADU does not exist
5006	FADU not available (unspecific)
5007	FADU not available for reading
5008	FADU not available for writing
5009	FADU not available for location
5010	FADU not available for erasure
5011	FADU cannot be insterted
5012	FADU cannot be replaced
5013	FADU cannot be located
5014	Bad data element type
5015	Operation not available
5016	Operation not supported
5017	Operation inconsistent
5018	Concurrency control not available
5019	Concurrency control not supported
5020	Concurrency control inconsistent
5021	Processing mode not available
5022	Processing mode not supported
5023	Processing mode inconsistent
5024	Access context not available

Table 4-6 Access Related Diagnostics

5025	Access context not supported
5026	Bad write (unspecific)
5027	Bad read (unspecific)
5028	Local failure (unspecific)
5029	Local failure - filespace exhausted
5030	Local failure - data corrupted
5031	Local failure - device failure
5032	Future file size exceeded
5034	Future file size increased
5035	Functional unit invalid in processing mode
5036	Contents type inconsistent
5037	Contents type simplified
5038	Duplicate FADU name
5039	Damage to select/open regime
5040	FADU locking not available on file
5041	FADU locked by another user

4

Recovery-Related Diagnostics

Codes `6000` to `6017` are used to return diagnostic information regarding the recovery of interrupted files as shown in Table 4-7:

Table 4-7 Recovery-Related Diagnostics

6000	Bad checkpoint (unspecific)
6001	Activity not unique
6002	Checkpoint outside window
6003	Activity no longer exists
6004	Activity not recognized
6005	No docket
6006	Corrupt docket
6007	File waiting restart
6008	Bad recovery point
6009	Non-existent recovery point
6010	Recovery mode not available
6011	Recovery mode inconsistent
6012	Recovery mode reduced
6013	Access control not available
6014	Access control not supported
6015	Access control inconsistent
6016	Contents type inconsistent
6017	Contents type simplified

Program Examples A

This appendix includes listings for the example headers and example initiator application that are delivered as part of the SunLink FTAM application software.

A

FTAM Initiator Application Example

The following example is included in the SunLink FTAM product in the file `/opt/SUNWconn/ftam/examples/ex_client.c`. The `Makefile` and `README` files associated with this example are also shown.

ex_client.c

```
#ident "@(#)ex_client.c 1.11 94/01/25 SMI"
/*
 * Copyright 1994 Sun Microsystems, Inc. All Rights Reserved
 */

/*****************************************************************************/
/*** Name         :      ex_client.c                                       ***/
/***                                                                       ***/
/*** Descritpion  :      FTAM Initiator API Programming use Example        ***/
/***                                                                       ***/
/*** Date         :      February 1993                                     ***/
/*****************************************************************************/

/*****************************************************************************/
/***                           STANDARD INCLUDE                            ***/
/*****************************************************************************/
#include <stdio.h>
#include <string.h>
#include <sys/systeminfo.h>
/*****************************************************************************/
/***                           FTAM USER INCLUDE                           ***/
/*****************************************************************************/
#include "ftiuser.h"


/*****************************************************************************/
/***                           GENERAL CONSTANTS                           ***/
/*****************************************************************************/

#define      MAXSTRLEN      64
#define      RDB_FILE_NAME  "SUN_llc1"     /* name of .rdb file to use for*/
                                           /* loopback FTAM connection    */
#define      RDBTEST        "APITEST"      /* name of .rdb file created   */
                                           /* for API test                */

/*****************************************************************************/
```

```
#ident "@(#)ex_client.c 1.11 94/01/25 SMI"
/***                          GLOBAL VARIABLES                          ***/
/*****************************************************************************/
static char user[FTIMXSTR];                 /* Input parm: user_name     */
static char pass[FTIMXSTR];                 /* Input parm: password      */
static char remfile[] = {"FTAM_CREATE"};    /* Name of created file      */
static char renamedfile[] = {"FTAM_RENAME"}; /* Name of renamed file     */
static char hostname[FTIMXRHN];             /* temporary rdb file name   */

/*****************************************************************************/
/***                         EXTERN DECLARATIONS                         ***/
/*****************************************************************************/
extern  char    *optarg;

/*****************************************************************************/
/***                          LOCAL FUNCTIONS                            ***/
/*****************************************************************************/
void      print_error ();
void      usage ( char * );
int       Ascii_Hex ( unsigned char *, unsigned char * );

main( int argc, char *argv[] )
{

        ftiadmT         adm;            /* Administrative structure     */
        ftirdbT         rdb;            /* Rdb operations structure     */
        ftibndT         bnd;            /* Bind structure               */
        fticreT         crefile;        /* Create file structure        */
        ftidelT         delfile;        /* Delete file structure        */
        ftitrfT         trf;            /* File transfert structure     */
        ftirenT         renfile;        /* Rename file structure        */
        ftidockT        dock;           /* Docket structure             */
        ftilaiT         lai      = 0;   /* Local identifier             */
        ftiesrT         esr;            /* Status report                */
        int             recneg;         /* Recovery negociation result  */
        int             trfid;          /* Transfer identifier number   */
        char            *dir;           /* Environment variable         */
        int             i;              /* General counter              */
        int             netlen;         /* Network Address length       */
        char            netadd[MAXSTRLEN];      /* Store Network Address*/
        long            n_bytes;        /* To get Network Address info  */
        char            host_info[256]; /*     "    "       "      "    */

/*---------------------------------------------------------------------*/
/* GET INPUT Parameters and Checking                                   */
```

A

```
#ident "@(#)ex_client.c 1.11 94/01/25 SMI"
/*------------------------------------------------------------------------*/


        if( argc != 5 )
                usage(argv[0]);

        /* Get arguments */
        while(( i = getopt( argc, argv, "u:p:")) != EOF) {
                switch(i) {
                case 'u':
                    /* Username */
                    if( (int) strlen( &optarg[0] ) > FTIMXSTR ) {
                          printf( "Invalid username length\n");
                          exit( 1 );
                    }
                    strcpy( &user[0], &optarg[0] );
                    break;
                case 'p':
                    /* Password */
                    if( (int) strlen( &optarg[0] ) > FTIMXSTR ) {
                          printf( "Invalid password length\n");
                          exit( 1 );
                    }
                    strcpy( &pass[0], &optarg[0] );
                    break;
                default :
                        usage(argv[0]);
                    break;
                }
        }


/*------------------------------------------------------------------------*/
/* ENVIRONMENT VARIABLES Initialization                                   */
/*------------------------------------------------------------------------*/

        /* FTIINID: looks first into environment, if already defined    */
        dir = (char *)getenv("FTIINID") ;
        if (dir) {
              /* if yes, copy its value into fti_init structure */
              strcpy(adm.ftiinid,dir);
              printf("FTIINID : %s\n",dir);
        }
        /* else use default installation value */
```

```
#ident "@(#)ex_client.c 1.11 94/01/25 SMI"
        else strcpy(adm.ftiinid,"\0");

        /* FTILOGD: same as above */
        dir = (char *)getenv("FTILOGD") ;
        if (dir) {      /* yes, it is defined */
              strcpy(adm.ftilogd,dir);
              printf("FTILOGD : %s\n",dir);
        }
        else strcpy(adm.ftilogd,"\0");

        /* FTISHFD */
        dir = (char *)getenv("FTISHFD") ;
        if (dir) {      /* yes, defined */
              strcpy(adm.ftishfd,dir);
               printf("FTISHFD : %s\n",dir);
        }
        else strcpy(adm.ftishfd,"\0");

        /* FTITRF */
        dir = (char *)getenv("FTITRFD");
        if (dir) {      /* yes, defined */
              strcpy(adm.ftitrfd,dir);
              printf("FTITRFD : %s\n",dir);
        }
        else strcpy(adm.ftitrfd,"\0");

        /* NOTE: FTIRDBD take default installation value */
        strcpy(adm.ftirdbd, "\0");

/*--------------------------------------------------------------------------*/
/* Call FTAM Library Initialization function                                */
/*--------------------------------------------------------------------------*/

        printf("fti INITIALIZATION\n\t------------------> ");
        if (fti_init(&adm,&esr)) {
              /* error */
              printf("fti_init FAILED\n");
              print_error( &esr );
              exit(1);
        }
        else
              printf("fti_init SUCCEEDED\n");

/*--------------------------------------------------------------------------*/
```

```
#ident "@(#)ex_client.c 1.11 94/01/25 SMI"
/* Enter into Remote Systems Database input network address             */
/*--------------------------------------------------------------------------*/

        /* Name of "rdb" file to use: SUN_llc1 for loopback */
        strcpy( &rdb.hostname[0], RDB_FILE_NAME );
        /* Network Address assignement */
#ifdef HOSTNAME
        /* Use hostname as NSAP suffix -> TO CHANGE */
              n_bytes = sysinfo( SI_HOSTNAME, host_info, 256 );
              if( n_bytes != -1 )
                    strcpy( netadd, host_info );
#else
        /* Network address: for loopback use 49-hostid-01 */
              n_bytes = sysinfo( SI_HW_SERIAL, host_info, 256 );
              if( n_bytes != -1 ) {
                    sprintf( host_info, "49%x01", atoi(host_info) );
                    /* Convert from ASCII to Hexadecimal */
                    Ascii_Hex( &host_info[0], &netadd[0] );
              }
#endif
              else {      /* error in sysinfo */
                 printf( "Cannot build Network Address for loopback. Exit\n" );
                    exit(1);
              }


        /* Get info already stored for selectors we wont change */
        printf("Getting RDB file %s\n\t------------------> ", rdb.hostname );
        if (fti_rdbget(&rdb, 1, &esr)) {
              /* error */
              printf("fti_rdbget FAILED. Exit\n");
              print_error( &esr );
              exit(1);
        }
        else
              printf("Rdb file read SUCCESSFULLY\n");

        strcpy( &rdb.netadd[0], &netadd[0] );
        /* Change rdb file name */
        strcpy( &rdb.hostname[0], RDBTEST );
        rdb.netlen = (int) strlen( &rdb.netadd[0] );
        /* Now try to update ".rdb" file in database */
        printf("Creating RDB file %s for loopabck connection\n\t-------------
-----> ", rdb.hostname );
```

```
#ident "@(#)ex_client.c 1.11 94/01/25 SMI"
        if (fti_rdbput(&rdb, 1, &esr)) {
              /* error */
              printf("fti_rdbput FAILED. Exit\n");
              print_error( &esr );
              exit(1);
        }
        else
              printf("Rdb file created SUCCESSFULLY\n");
/*--------------------------------------------------------------------------*/
/* ESTABLISH A CONNECTION with a remote host system                         */
/*--------------------------------------------------------------------------*/


        /* Enter host and username */
        strcpy(bnd.hostname, rdb.hostname);
        strcpy(bnd.username,user);

        /* Init password parameter */
        /* NOTE: 1st char *must* be zero */
        bnd.password[0] = 0;
        bnd.password[1] = (int) strlen( pass );
        strcpy( &bnd.password[2], pass );
        bnd.account[0] = '\0';

       printf("Try to connect with %s\n\t------------------> ", bnd.hostname);
        if (fti_connect(&lai,&bnd,&recneg,FTIEXBLK,&esr)) {
              /* error */
              printf("fti_connect FAILED\n");
              /* Remove ".rdb" file created for test */
              strcpy( hostname, RDBTEST );
              if (fti_rdbdel(hostname, 1, &esr)) {
                    /* error */
                    printf("fti_rdbdel FAILED. Exit\n");
                    print_error( &esr );
                    exit(1);
              }
              print_error( &esr );
              exit(1);
        }
        else /* OK */
              printf("fti_connect SUCCEEDED\n");

        /* Check RECOVERY Negotiation Result */
        if (recneg != FTIRCAVA)      /* Failed */
```

```
#ident "@(#)ex_client.c 1.11 94/01/25 SMI"
              printf("\t\tRECOVERY NEGOTIATION FAILED\n");

/*--------------------------------------------------------------------*/
/* CREATE a file on remote host                                       */
/*--------------------------------------------------------------------*/

        printf("Try to CREATE %s file on %s host\n\t------------------> ",
remfile, bnd.hostname );

        strcpy(crefile.filname,remfile);
        crefile.doctyp     = FTMDTUNO;  /* type unknown */
        crefile.filsize    = 100;       /* max size     */
        crefile.override   = FTMOVDNA;  /* del + create with new att */
        crefile.peract[0]  = FTMPAREA | FTMPAINS | FTMPAREP | FTMPAEXT;
       crefile.peract[0] |= FTMPAERA | FTMPARAT | FTMPACAT | FTMPADEL; /* all
permitted */
        crefile.avail      = FTMFAIMM;
        crefile.accnt[0]   = '\0';
        crefile.filpssd[1] = 0x00;
        crefile.acctrl[0]  = '\0';
        crefile.shfname[0] = '\0';

        if (fti_create(&lai,&crefile,FTIEXBLK,&esr)){
              /* Error */
              printf("fti_create FAILED\n");
              print_error( &esr );
        }
        else      /* OK */
              printf("fti_create SUCCEEDED\n");

/*--------------------------------------------------------------------*/
/* RENAME a file on remote host                                       */
/*--------------------------------------------------------------------*/

      printf("Try to RENAME %s file with %s\n\t------------------> ", remfile,
renamedfile);

        strcpy( renfile.filname, remfile );
        strcpy( renfile.newname, renamedfile );
        renfile.accnt[0]   = '\0';
        renfile.filpssd[1] = 0x00;

        if (fti_rename(&lai,&renfile,FTIEXBLK,&esr)){
              /* Error */
```

```
#ident "@(#)ex_client.c 1.11 94/01/25 SMI"
              printf("fti_rename FAILED\n");
              print_error( &esr );
        }
        else      /* OK */
              printf("fti_rename SUCCEEDED\n");

/*-----------------------------------------------------------------------*/
/* PUT a file from the local system to remote host                       */
/*-----------------------------------------------------------------------*/

        trf.doctyp   = FTMDTFT3;        /* type FTAM-3                  */
        trf.override = FTMOVDNA;        /* del + create with new att    */
        trf.mode     = FTMTMEXT;        /* extend content of remote     */
        sprintf( trf.locname, "%s.o", argv[0] );
        strcpy( trf.remname, remfile );

        if (recneg == FTIRCAVA)       {      /* Recovery was negotiated */
              if (fti_getid(&lai,&trfid,&esr)) { /* Error */
                    printf("fti_getid FAILED\n");
                    print_error( &esr );
                    printf("Cannot put %s on %s\n", argv[0], remfile);
                    goto exit_ex_client;
              }
              trf.trfid = trfid;     /* get transfer idf. no. */
        }
        trf.recmode = FTIRMNOR;      /* recovery mode off     */

        printf("Try to PUT file %s.o in file %s\n\t------------------> ",
argv[0], remfile);

        if (fti_put(&lai,&trf,FTIEXBLK,&esr)) {
              /* Error */
              printf("fti_put FAILED\n");
              if ((esr.errtype == FTITYRCO) && (esr.errcnxs == FTICNXON)) {
                    dock.id = trfid;
                    if (!fti_recover(&lai,&dock,FTIEXBLK,&esr)) {
                          printf("fti_recover FAILED\n");
                          print_error( &esr );
                          exit(1);
                    }
              }
              else      print_error( &esr );
        }
        else { /* OK */
```

```
#ident "@(#)ex_client.c 1.11 94/01/25 SMI"
            printf("fti_put SUCCEEDED\n");
            printf("\tBytes sent : %d\n",trf.result);
            printf("\tTime elapsed : %f seconds\n",trf.elapse);
        }

/*------------------------------------------------------------------------*/
/*  DELETE all created/put files on remote host                           */
/*------------------------------------------------------------------------*/

        printf("Try to DELETE %s file on %s host\n\t------------------>
",renamedfile,bnd.hostname);

        strcpy(delfile.filname,renamedfile);
        delfile.filpssd[1] = 0x00;
        delfile.accnt[0]   = '\0';

        if (fti_delete(&lai,&delfile,FTIEXBLK,&esr)){
              /* Error */
              printf("fti_delete FAILED\n");
              print_error( &esr );
        }
        else      /* OK */
              printf("fti_delete SUCCEEDED\n");

        printf("Try to DELETE %s file on %s host\n\t------------------>
",remfile,bnd.hostname);

        strcpy(delfile.filname,remfile);
        delfile.filpssd[1] = 0x00;
        delfile.accnt[0]   = '\0';

        if (fti_delete(&lai,&delfile,FTIEXBLK,&esr)){
              /* Error */
              printf("fti_delete FAILED\n");
              print_error( &esr );
        }
        else      /* OK */
              printf("fti_delete SUCCEEDED\n");

/*------------------------------------------------------------------------*/
/* CLOSE the CONNECTION AND EXIT                                          */
/*------------------------------------------------------------------------*/

exit_ex_client:      printf("Try to DISCONNECT\n\t------------------> ");
```

```
#ident "@(#)ex_client.c 1.11 94/01/25 SMI"

              if (fti_disconnect(&lai,FTIEXBLK,&esr)){
                    /* Error */
                    printf("fti_disconnect FAILED\n");
                    print_error( &esr );
              }
              else      /* OK */
                  printf("fti_disconnect SUCCEEDED\n");

              /* Remove ".rdb" file created for test */
              strcpy( hostname, RDBTEST );
              if (fti_rdbdel(hostname, 1, &esr)) {
                    /* error */
                    printf("fti_rdbdel FAILED. Exit\n");
                    print_error( &esr );
                    exit(1);
              }
              else
                    printf("Rdb file %s deleted SUCCESSFULLY\n", hostname);

              fti_exit();

              printf("END OF EXAMPLE.....\n");
}

/*
 *      PRINT_ERROR:
 *      ------------
 *      prints out the error status report structure in case an API function
 *      returns an error code.
 *      PARAMETERS: -> pointer to Execution Status Report structure to print
 */
void      print_error(
                    ftiesrTp      err_ptr
                  )
{
        char      st_status[MAXSTRLEN];

        fprintf( stderr, "Execution Status Report:\n" );
        switch( err_ptr->status ) {
        case      FTISTLOC: /* Local Error */
              strcpy( st_status, "LOCAL Error");
              break;
        case      FTISTSYS: /* System Error */
```

```
#ident "@(#)ex_client.c 1.11 94/01/25 SMI"
              strcpy( st_status, "SYSTEM Error");
              break;
        case      FTISTFTM: /* FTAM Error */
              strcpy( st_status, "FTAM Error");
              break;
        case      FTISTUFE: /* Unexpected FTAM Event Error */
              strcpy( st_status, "Unexpected FTAM Event");
              break;
        case      FTISTWAR: /* Local Warning */
              strcpy( st_status, "LOCAL WARNING");
              break;
        default            : /* Should not be possible */
              strcpy( st_status, "NO ERROR!");
        } /* switch status */
        fprintf( stderr, "Status\t\t= %s\n", st_status);

        fprintf( stderr, "Error Code\t= %d\n", err_ptr->errcode);
        fprintf( stderr, "Error Type\t= %s\n", (err_ptr->errtype) ?
"RECOVERABLE"
                                                    : "NOT RECOVERABLE");
      fprintf( stderr, "Cnx Status\t= %s\n", (err_ptr->errcnxs) ? "ON" : "OFF"
);
        if( err_ptr->status == FTISTFTM )
              fprintf( stderr, "Diagnostic\t= %.4d\n", err_ptr->diag);
        else if( err_ptr->status == FTISTSYS )
            fprintf( stderr, "Diagnostic\t= %s\n", strerror( err_ptr->diag) );
        else       fprintf( stderr, "Diagnostic\t= %.8x\n", err_ptr->diag);

} /* print_error end */

/*
^L
 *      ASCII_HEX:
 *      ----------
 *      convert an ASCII string of hexadecimal digits into its hexadecimal
 *      corresponding format.
 *      INPUT PARAMETERS:
 *      from    --->    input ASCII string
 *      to      --->    result of conversion (OUT string)
 *      RETURNS:
 *      0 (OK) or 1 (error)
 */
int
Ascii_Hex(
```

```
#ident "@(#)ex_client.c 1.11 94/01/25 SMI"
                unsigned char   *from,
                unsigned char   *to
        )
{
        int l;

        for (l=0; (*from)!='\0'; ) {
                if( !isxdigit(*from) )
                        return( 1 );     /* *not* hexa character */
                *to = (*from<'A') ? ((*from++)-'0') << 4 : (toupper(*from++)-
'A'+10) << 4 ;
                if (*from){
                        if (!isxdigit(*from))
                                return( 1 );
                   *to++ |= (*from<'A') ? ((*from++)-'0') : (toupper(*from++)-
'A' +10);

                }
               else *to++ |= 0x0F;   /* pads even bytes no. with "0x0F" byte */
                l++;
        }       /* for l */
        *to = 0x00;
        return( 0 );
} /* Ascii_Hex end */

/*
^L
 *      USAGE:
 *      ------
 *      print example usage.
 */
void usage(char *prog_n)
{
        fprintf( stderr, "Usage: %s -u User_name -p Password\n", prog_n );
        exit(1);
} /* usage end */
```

A

Makefile

```
#
#ident "@(#)Makefile 1.8 93/07/20 SMI"
#
#         Copyright 1994 Sun Microsystems, Inc. All Rights Reserved
#
#         Makefile for generation of  FTAM HL API programming example
#

PROGRAM = ex_client

# FTAM example sources, library, includes home directory
FTAM_HOME = /opt/SUNWconn/ftam

# FTAM Library path
LIB_PATH = ${FTAM_HOME}/lib

# FTAM User include path
INCL =    ${FTAM_HOME}/include

SOURCES = ex_client.c

OBJECTS = \
          $(SOURCES:%.c=%.o)

#
# Compiler flags.
#

MYDEFS =
CFLAGS += -g -w
CPPFLAGS += -I$(INCL) $(MYDEFS)
LDFLAGS += -L$(LIB_PATH)
LDLIBS += -lnsl -lintl -lftam


#
# Standard targets.
#

all:      $(PROGRAM)

clean:
          rm -f $(OBJECTS) $(PROGRAM)
```

```
#

$(PROGRAM): $(OBJECTS)
          @echo -------------- $@ link begins --------------
          @$(CC) $(LDFLAGS) $(OBJECTS) $(LDLIBS) -o $@
          @echo -------------- $@ link end    --------------

.c.o:
          @echo -------------- $@ compilation begins -------
          @$(CC) $(CFLAGS) $(CPPFLAGS) -c $<
          @echo -------------- $@ compilation end    -------

# Dependencies

ex_client.c: ${INCL}/ftiuser.h
          @touch $@
```

A

README

```
#ident "@(#)README 1.8 94/01/18 SMI"
#
#       Copyright 1994 Sun Microsystems, Inc. All Rights Reserved
#
#         README file for generation of an example of FTAM HL API use
#

The current directory contains :

A -- a programming example using the FTAM High Level API (Initiator).
B -- a stubs for generating customized FTAM Reponders, using the FTAM HL API
     ( Responder ).

                                - o -

A)        FTAM HL API Initiator Programming Example

The example implements an FTAM Client (or Initiator) performing a given set
of file access, management and transfer operations (see below).
The example will open a connection in loopback on the same machine where
it is executed. Therefore, to run it the SunLink 8.0.1 FTAM Responder Daemon
osiftrd on clnp (LAN) must be runnning on the machine.

Follow these steps to build and run the "ex_client" program:

a) type "make" to compile and generate the executable

b) start on the local machine the FTAM daemon (osiftrd) over "clnp" (if not
   already started) :
   either
   as root, type: "/etc/rc.3/S147osiftrd start" (start all osiftrd daemons)
   or
   as root, type "/opt/SUNWconn/ftam/bin/osiftrd_start clnp".

   The following message will be displayed:
   ## FTAM Responder Daemon (osiftrd) on network <clnp> started.

c) at last, to execute the ex_client program, type

                    ex_client -u User_name -p Password

   where:
```

```
#ident "@(#)README 1.8 94/01/18 SMI"
                    - "User_name" is the login name of a user which is present
                      into "passwd" file on the local machine

                    - "Password" is his/her password

The example will perform the following operations ( suited status messages
will be displayed on std. output ):

1) In order to establish an FTAM connection in loopback on the local machine,
   a temporary rdb file is created, by copying all the OSI address elements in
   the template "SUN_llc1", which is automatically installed under
   "/var/SUNWconn/OSIROOT/ftam/rdb" directory.

2) Establish a connection in loopback, by using the "APITEST" rdb file.

3) Create an empty file (called FTAM_CREATE, type F3)

4) Rename this file (from FTAM_CREATE to FTAM_RENAME)

6) Put a file from the local to the remote system ( put ex_client.o to
   FTAM_CREATE )

5) Delete all created files ( FTAM_CREATE and FTAM_RENAME )

7) Close the connection

8) Remove the APITEST rdb file

In case of error during one of these operations a suited message is displayed
on standard output along with the "Execution Status Report", data structure
passed as parameter to each API function and containing different information
about possible error conditions during the function execution
(e.g., connection status, error code, etc.)

It should be noted that the remote directory where the test files are
created/deleted/etc. is the 'home' directory of the user "User_name" on the
local machine.

                              - o -

B)        FTAM HL API Responder Use

The file "ftr_stubs.c" is a stubs file for generating customized FTAM
Responders.
```

```
#ident "@(#)README 1.8 94/01/18 SMI"
It contains the skeleton of all customizable Responder functions.
Some of these functions, depending on their return code, may inhibit the
successfull completion of the corresponding FTAM operations.
So, if the programmer wants to prevent the execution of a given FTAM operation,
(s)he has to made the corresponding function returning a non-zero value.

To generate a 'customized Reponder' do the following steps
(see also "SunLink FTAM 8.0.1 Programmer's Guide", Chapt. 1, pag. 3)

1) Edit the file 'ftr_stubs.c' and customize the FTAM functions as wanted.

2) Compile and link it to the relocatable file 'osiftrapi.o', which is
   installed under '/opt/SUNWconn/lib'.

3) To use the customized Responder, you must perform a symbolic link ( or do a
   copy ) of your Responder 's executable, say 'my_osiftr' :

   as root, type
   mv /opt/SUNWconn/ftam/bin/osiftr /opt/SUNWconn/ftam/bin/osiftr.BAK
   - this will make a copy of the standard Responder -

   ln -s /opt/SUNWconn/ftam/examples/my_osiftr /opt/SUNWconn/ftam/bin/osiftr
  - the customized responder program's name must be 'osiftr' (as the std. one)
    and it must be installed in the same directory then all the ftam executables
     (by default '/opt/SUNWconn/ftam/bin')

NOTE:
     you do not need to stop and restart the osiftrd daemon, *but* on a given
     network, once performed the above step #3, your customized Responder will
     be used instead of the standard one on *every* network (cons, clnp and
     tcp/ip).
```

FTAM Initiator Header (`ftiuser.h`*)*

The content of the C header file `ftiuser.h` which must be included in all applications developed using the FTAM Initiator interfaces is listed on the following pages:

```
#ident "@(#)ftiuser.h 1.14 94/01/11 SMI"
/*
          Copyright 1993 Sun Microsystems, Inc. All Rights Reserved

          (c) Copyright 1991 Marben Produit.    All rights reserved
*/

#ifndef   _FTIUSER_H
#define   _FTIUSER_H

#ifdef  __cplusplus
extern "C" {
#endif

/*-----------------------------------------------------------------**
**                            General Constants                    **
**-----------------------------------------------------------------*/

#include  <sys/param.h>/* used for MAXPATHLEN*/
#define FTIMXDIR  MAXPATHLEN/* MaXimum DIRectory Length    */
#define FTIMXSTR 128/* MaX General Purpose Length*/
#define FTIMXDAT 20 /* MaXimum Date and Time Length*/
#define FTIMXDFD 128/* MaXimum FTAM Diagnostic */

/*-----------------------------------------------------------------**
**                            Remote Host Database Constants**
**        should not be changed!                                   **
**-----------------------------------------------------------------*/

#define FTIMXRHN  9  /* MaXimum Host Name Length            */
#define FTIMXPSL 33 /* MaXimum Presentation Selector Length*/
#define FTIMXSSL 33 /* MaXimum Session Selector Length    */
#define FTIMXTSL 33 /* MaXimum Transport Selector Length   */
#define FTIMXNAL 41 /* MaXimum Network Address Length      */
#define FTIMXATL 21 /* MaXimum AET Length */
#define FTIMXSBN 32 /* MaXimum SuBnet Number               */


/*-----------------------------------------------------------------**
```

```
#ident "@(#)ftiuser.h 1.14 94/01/11 SMI"
**                              Document Types                          **
**--------------------------------------------------------------------*/

#define FTMDTUNO  0 /* Document Type - UNknOwn*/
#define FTMDTFT1  1 /* FTAM_1  Document Type*/
#define FTMDTFT2  2 /* FTAM_1  Document Type*/
#define FTMDTFT3  3 /* FTAM_1  Document Type*/
#define FTMDTNB9  9 /* NBS_9   Document Type*/
#define FTMDTIN1 10 /* INTAP_1 Document Type*/

/*--------------------------------------------------------------------**
**                              Override Parameters           **
**--------------------------------------------------------------------*/

#define FTMOVNOC 0  /* NO Creation, Create Failure*/
#define FTMOVSOF 1  /* Select Old File  */
#define FTMOVDOA 2  /* Deleted and created OLD attr.*/
#define FTMOVDNA 3  /* Deleted and created NEW attr.*/

/*--------------------------------------------------------------------**
**                              Processing Mode                       **
**--------------------------------------------------------------------*/

#define FTMTMREP 0x20/* REPLACE mode              */
#define FTMTMEXT 0x10/* EXTEND mode               */

/*--------------------------------------------------------------------**
**                              Permitted Actions             **
**--------------------------------------------------------------------*/

#define FTMPAREA 0x80/* READ                      */
#define FTMPAINS 0x40/* INSERT                    */
#define FTMPAREP 0x20/* REPLACE                   */
#define FTMPAEXT 0x10/* EXTEND                    */
#define FTMPAERA 0x08/* ERASE                     */
#define FTMPARAT 0x04/* READ-ATTRIBUTE  */
#define FTMPACAT 0x02/* CHANGE ATTRIBUTE*/
#define FTMPADEL 0x01/* DELETE file               */

/*--------------------------------------------------------------------**
**                              Universal Class Numbers       **
**--------------------------------------------------------------------*/

#define FTMUCOCT 04  /* OctetString     */
```

```
#ident "@(#)ftiuser.h 1.14 94/01/11 SMI"
#define FTMUCPRT 19  /* PrintableString */
#define FTMUCTXT 20  /* TeletexString   */
#define FTMUCVXT 21  /* VideotexString  */
#define FTMUCIA5 22  /* IA5String       */
#define FTMUCGRP 25  /* GraphicString   */
#define FTMUCVIS 26  /* VisibleString   */
#define FTMUCGEN 27  /* GeneralString   */

/*-----------------------------------------------------------------**
**                            String Significance                  **
**-----------------------------------------------------------------*/

#define FTMPDVAR  0  /* variable               */
#define FTMPDFIX  1 /* fixed                     */
#define FTMPDNOT  2 /* not significant  */

/*-----------------------------------------------------------------**
**                            Passwords Types                       **
**-----------------------------------------------------------------*/

#define FTIGraStr 0x00/* Graphic String */
#define FTIOctStr 0x01/* Octet String            */

/*-----------------------------------------------------------------**
**                            Requested Attributes                 **
**-----------------------------------------------------------------*/

#define FTMAFNAME 0x80/* read-filename  */
#define FTMAPERAC 0x40/*  " permitted-actions*/
#define FTMACOTYP 0x20/*  " contents_type*/
#define FTMASTACC 0x10/*  " storage-account*/
#define FTMADTCRE 0x08/*  " date and time of creation */
#define FTMADTLMO 0x04/*  " date and time of last     *
                                        *       modification    */
#define FTMADTLRA 0x02/*  " date and time of last *
                                        *       read access     */
#define FTMADTLAM 0x01/*  " date and time of last*
                                        *       attribute modification*/
#define FTMAIDLCR 0x80/*  " identity of last creator*/
#define FTMAIDLMO 0x40/*  " identity of last modifier*/
#define FTMAIDLRE 0x20/*  " identity of last reader*/
#define FTMAIDLAM 0x10/*  " identity of last *
                                        *       attribute modifier*/
#define FTMAFAVAI 0x08/*  " file availability*/
```

A

```
#ident "@(#)ftiuser.h 1.14 94/01/11 SMI"
#define FTMAFSIZE 0x04/*  " filesize              */
#define FTMAFFSIZ 0x02/*  " future file size*/
#define FTMAACCTL 0x01/*  " access-control*/

/*--------------------------------------------------------------**
**                            Kernel Group Attributes (Masks)**
**--------------------------------------------------------------*/

#define FTMAGKER0 (unsigned char)0xe0/* kernel group byte 0*/
#define FTMAGKER1 (unsigned char)0x00/*         byte 1*/
#define FTMAGKER2 (unsigned char)0x00/* byte 2*/

/*--------------------------------------------------------------**
**                            Storage Group Attributes (Masks)**
**--------------------------------------------------------------*/

#define FTMAGSTG0 (unsigned char)0x1f/* storage group byte 0*/
#define FTMAGSTG1 (unsigned char)0xfe/* byte 1*/
#define FTMAGSTG2 (unsigned char)0x00/* byte 2*/

/*--------------------------------------------------------------**
**                            File Availability             **
**--------------------------------------------------------------*/

#define FTMFAIMM  0x00/* IMMEDIATE availability*/

/*--------------------------------------------------------------**
**                            Execution Mode                  **
**--------------------------------------------------------------*/

#define FTIEXNBL 0  /* Execution mode: NON Blocking */
#define FTIEXBLK 1  /* Execution mode: BLOCKING     */

/*--------------------------------------------------------------**
**                            Host State                      **
**--------------------------------------------------------------*/

#define FTIHSOPE 1  /* Host State: OPEN */
#define FTIHSCLO 0  /* Host State: CLOSED*/

/*--------------------------------------------------------------**
**                            Recovery Mode                   **
**--------------------------------------------------------------*/
```

```
#ident "@(#)ftiuser.h 1.14 94/01/11 SMI"
#define FTIRMRON 1  /* Recovery Mode: Recovery ON*/
#define FTIRMNOR 0  /* Recovery Mode: NO Recovery*/

/*-----------------------------------------------------------------**
**                            Recovery Availability         **
**-----------------------------------------------------------------*/

#define FTIRCAVA 1  /* Recovery AVAILABLE*/
#define FTIRCNAV 0  /* Recovery NOT Available*/

/*-----------------------------------------------------------------**
**                            More Indicator                       **
**-----------------------------------------------------------------*/

#define FTINOMO  0  /* NO MORE                    */
#define FTIMORE  1  /* MORE                       */

/*-----------------------------------------------------------------**
**                            API Event Codes (multi-connections)**
**-----------------------------------------------------------------*/

#define FTINOEVT 0  /* NO EVENT received*/
#define FTIEVABO 1  /* Event code: Abort*/
#define FTIEVOPE 2  /* EVent code: Open */
#define FTIEVCLO 3  /* EVent code: Close*/
#define FTIEVPUT 4  /* EVent code: Put  */
#define FTIEVGET 5  /* EVent code: Get  */
#define FTIEVCRE 6  /* EVent code: Create*/
#define FTIEVDEL 7  /* EVent code: Delete*/
#define FTIEVREN 8  /* EVent code: Rename*/
#define FTIEVRAT 9  /* EVent code: Read Attribute*/

/*-----------------------------------------------------------------**
**                            Execution Status              **
**-----------------------------------------------------------------*/

#define FTISTNER 0  /* NO Error                   */
#define FTISTLOC 1  /* LOCAL or user error*/
#define FTISTSYS 2  /* local SYSTEM error*/
#define FTISTFTM 3  /* FTAM service or protocol err*/
#define FTISTUFE 4  /* Unexpected FtAM event*/
#define FTISTWAR 5  /* local WARNING    */

/*-----------------------------------------------------------------**
```

A

```
#ident "@(#)ftiuser.h 1.14 94/01/11 SMI"
**                          Error Types                          **
**---------------------------------------------------------------*/

#define FTITYNRC 0  /* NOT Recoverable error*/
#define FTITYRCO 1  /* RECOVERABLE     error*/

/*---------------------------------------------------------------**
**                          Error Connexion Status        **
**---------------------------------------------------------------*/

#define FTICNXON 1  /* CoNneXion ON              */
#define FTICNXOF 0  /* CoNneXion OFF    */

/*---------------------------------------------------------------**
**                          Error Codes                          **
**---------------------------------------------------------------*/

#define FTIERIEV 1  /* FTI ERror, Incorrect Environment Variable */
#define FTIERNSU 2  /* FTI ERror, function Not SUpported         */
                    /*            on local host                  */
#define FTIERIPA 3  /* FTI ERror, Invalid PArameter              */
/*               4     not used                                   */
#define FTIERCNE 5  /* FTI ERror, Connection Not Established     */
#define FTIERCAE 6  /* FTI ERror, Connection Already Established */
#define FTIERLFR 7  /* FTI ERror, Local file name required       */
#define FTIERRFR 8  /* FTI ERror, Remote file name required      */
/*                   9     not used                                 *
 *                   10    not used                                 */
#define FTIERUHN 11 /* FTI ERror, Unknown host name              */
#define FTIEROLF 12 /* FTI ERror, Opening Local File             */
#define FTIERORF 13 /* FTI ERror, Opening Remote File            */
#define FTIERFCH 14 /* FTI ERror, Failed to Connect to remote    */
                    /*            host. Error connecting to host */
#define FTIERCLF 15 /* FTI ERror, Closing Local File             */
#define FTIERCRF 16 /* FTI ERror, Closing Remote File            */
#define FTIERSLF 17 /* FTI ERror, Sending Local File             */
#define FTIERRRF 18 /* FTI ERror, Receiving Remote File          */
#define FTIERFNA 19 /* FTI ERror, Function Not Available         */
                    /*            on remote host                 */
/*                   20    not used                                 *
 *                   21    not used                                 *
 *                   22    not used                                 */
#define FTIERCRH 23 /* FTI ERror, Closing Remote Host            */
#define FTIERWHI 24 /* FTI ERror, Writing Host Information       */
```

```
#ident "@(#)ftiuser.h 1.14 94/01/11 SMI"
#define FTIERRSF 25 /* FTI ERror, Reading Shadow File            */
#define FTIERRDS 26 /* FTI ERror, Reading Default Shadow File    */
/*                   27    not used                                 *
 *                   28    not used                                 */
#define FTIERIPN 29 /* FTI ERror, Invalid Path Name              */
#define FTIERRHI 30 /* FTI ERror, Reading Host Information       */
#define FTIERRLF 31 /* FTI ERror, Reading Local File             */
#define FTIERWLF 32 /* FTI ERror, Writing Local File             */
#define FTIERWSF 33 /* FTI ERror, Writing Shadow File            */
/*                   34    not used                                    */
#define FTIERDRF 35 /* FTI ERror, Deleting Remote File           */
#define FTIERCTF 36 /* FTI ERror, Creating Remote File           */
#define FTIERRNF 37 /* FTI ERror, Renaming Remote File           */
#define FTIERRRA 38 /* FTI ERror, Reading Remote File Attributes */
/*                   39    not used                                 */
#define FTIERIDT 40 /* FTI ERror, Invalid Document Type          */
/*                   41    not used                                 */
#define FTIERHNR 42 /* FTI ERror, Host name required             */
/*                   43 - 50 not used                         */
#define FTIERUDH 51 /* FTI ERror, Unable to Delete Host          */
/*                   52    not used                                 *
 *                   53    not used                                 *
 *                   54    not used                                 */
#define FTIERHIC 55 /* FTI ERror, Host is closed                 */
/*                   56 - 62 not used                         */
#define FTIERRDF 63 /* FTI ERror, Reading Remote File            */
/*                   64    not used                                 *
 *                   65    not used                                 */
#define FTIERSRF 66 /* FTI ERror, Selecting Remote File          */
#define FTIERDSF 67 /* FTI ERror, Deselecting Remote File        */
#define FTIERGID 68 /* FTI ERror, Getting transfer IDentifier    */
#define FTIERRTR 69 /* FTI ERror, Recovering TRansfer            */
#define FTIERCDK 70 /* FTI ERRor, Creating DocKet                */
#define FTIERWDK 71 /* FTI ERRor, Writing  DocKet                */
#define FTIERRDK 72 /* FTI ERror, Reading  Docket                */
#define FTIERDDK 73 /* FTI ERror, Deleting Docket                */

/*----------------------------------------------------------------**
**                           Local (non-Unix) Errors       **
**----------------------------------------------------------------*/

#define ERRDTYP     1 /* Illegal document type parameters*/
#define ERRMODE     2 /* Invalid mode in F-Open-ind*/
#define ERRSUBSQ3 /* Subsequent error   */
```

A

```
#ident "@(#)ftiuser.h 1.14 94/01/11 SMI"
#define ERRFNAME4 /* No file name in F-Create-ind*/
#define ERRCFNAME5 /* No file name in F-ChAtt-ind*/
#define ERRDTPABS       6 /* No composite attributes in*
                          *                F-Create-ind    */
#define ERRIOI          7 /* Unknown user (F-Init-ind)*/
#define ERRUID          8 /* No user identity or cannot set*
                          *      group id. or user id. or*
                          *      cannot change directory*/
#define ERRSRC          9 /* Invalid service class in F-Init-ind*/
#define ERRPSWD        10 /* Invalid password in F-Init-ind*/
#define ERRDTTYP       11 /* Invalid Data Element type*/
#define ERRDTINV       12 /* Invalid Universal Class*/
#define ERRDTSZ        13 /* Invalid Data Element size*/
#define ERRTIMEOUT     14 /* Time-out   */
#define ERRRECLG       15 /* Invalid INTAP_1 record length*/
#define ERRRDEOF       16 /* Cannot read at end of file*/
#define ERRFADUI       17 /* FADU id. error in F-Read-ind */
#define ERRLFADU       18 /* Cannot locate FADU          */
#define ERRTRFIL       19 /* Cannot truncate file       */
#define ERRSFNLG       20 /* File name too long          */
#define ERRDCTYP       21 /* Invalid document type       */
#define ERROVERW       22 /* Invalid overwrite parameter */
#define ERRNODTY       23 /* No document type in F-Open-cnf*/
#define ERRFTYPE       24 /* Unsupported Special local file type*/
#define ERRSTRLG       25 /* Invalid FTAM_1 or FTAM_2*
                          *                string length    */
#define ERRCHECK       26 /* Recovery error - Bad checkpoint*/
#define ERRDOCK        27 /* Recovery error - Corrupted docket*/
#define ERRNODOCK      28 /* Recovery error - No docket*/
#define ERRRECPT       29 /* Recovery error - No recovery point */
#define ERRCONTYP      30 /* Recovery error - Content type*
                          *                          inconsistent*/
#define ERRUNSPEC      31 /* Recovery error - Unspecific*/

/*----------------------------------------------------------------**
**                            Warning Codes                       **
**----------------------------------------------------------------*/

#define FTIWRINI         1 /* WaRning on INItialisation       */
#define FTIWRTRL        30 /* WaRning on Transfer List        */

/* local warnings */
/* Initialisation warnings */
#define WARINID 0x00000001 /* Init directory*/
```

```
#ident "@(#)ftiuser.h 1.14 94/01/11 SMI"
#define WARLOGD 0x00000002 /* Log  directory                    */
#define WARSHFD 0x00000004 /* Shf  directory                    */
#define WARRDBD 0x00000008 /* Rdb  directory                    */
#define WARTRFD 0x00000010 /* Trf  directory                    */
#define WARINIF 0x00000020 /* Init file                         */
#define WARTRFF 0x00000040 /* Transfer identifiers file         */
/* Tranfer List warnings */
#define WARNODA 0x00000001 /* NO Docket Available               */

/*-----------------------------------------------------------------**
**                           Administrative Variables Structure**
**-----------------------------------------------------------------*/

typedef struct
{
  unsigned charftiinid[FTIMXDIR];/* INIT Directory*/
  unsigned charftilogd[FTIMXDIR];/* LOG Files Directory*/
  unsigned charftishfd[FTIMXDIR];/* SHADOW Files Database*/
  unsigned charftirdbd[FTIMXDIR];/* REMOTE HOSTS Database*/
  unsigned charftitrfd[FTIMXDIR];/* Transfers Database*/
  long              logid;              /* Log File Identifier*/
} ftiadmT, *ftiadmTp;

/*-----------------------------------------------------------------**
**                           Local Association Identifier Type **
**-----------------------------------------------------------------*/

typedef long ftilaiT;
typedef ftilaiT *ftilaiTp;

/*-----------------------------------------------------------------**
**                           Execution Status Report Structure**
**-----------------------------------------------------------------*/

typedef struct
{
  int               status;                       /* Execution Status*/
  int               errcode;            /* Error Code       */
  int               errtype;            /* Error Type       */
  int               errcnxs;            /* Err Connexion Status */
  int               diag;                         /* Diagnostic Code*/
  unsigned charftamdfd[FTIMXDFD];/* FTAM Diagnostic:*
                                                   *        Further Details
*/
```

```
#ident "@(#)ftiuser.h 1.14 94/01/11 SMI"
} ftiesrT, *ftiesrTp;

/*-----------------------------------------------------------------**
**                          Document Type Parameters Structure**
**-----------------------------------------------------------------*/

typedef struct
{
  unsigned charchoice;                  /* Qualifier             */
  short             class;                      /* Universal Class Number*/
  long              length1;            /* Maximum String Length */
  long              length2;            /* String Length Signific*/
} ftidtpT, *ftidtpTp;

typedef ftidtpT dtparmT;/* for upward compatib.*/
typedef ftidtpTp dtparmTp;/* for upward compatib.*/

/*-----------------------------------------------------------------**
**                          Shadow Files Structure        **
**-----------------------------------------------------------------*/

typedef struct
{
  int               doctyp ;            /* document type number*/
  unsigned charperact[2] ;    /* permitted actions*/
  long              keypos ;            /* key position (NBS6/7)*/
  ftidtpT parm ;                        /* document type param.*/
} ftishfT, *ftishfTp ;

typedef ftishfT shfT;      /* for upward compatib.*/
typedef ftishfTp shfTp;    /* for upward compatib.*/

/*-----------------------------------------------------------------**
**                          Remote Host DataBase Structure**
**-----------------------------------------------------------------*/

typedef struct
{
  unsigned charhostname[FTIMXRHN];/* Remote host name*/
  int               pstlen;                     /* Pres. sel. length    */
  unsigned char pstsel[FTIMXPSL];/* Pres. sel. value     */
  int               seslen;                     /* Session sel. length  */
  unsigned charsessel[FTIMXSSL];/* Session sel. value*/
  int               trslen;                     /* Transport sel. length*/
```

```
#ident "@(#)ftiuser.h 1.14 94/01/11 SMI"
  unsigned chartrssel[FTIMXTSL];/* Transport sel. value*/
  int               netlen;                     /* Network addr. length*/
  unsigned char netadd[FTIMXNAL];/* Network addr. value*/
  int               aetlen;                     /* Appl.Ent.Title length*/
  unsigned char aetitle[FTIMXATL];/* Appl.Ent.Title value */
  unsigned char subnet;       /* Subnet number*/
  unsigned char state;        /* Remote host state*/
} ftirdbT, *ftirdbTp;

typedef ftirdbT rdbT;      /* for upward compatib.*/
typedef ftirdbTp rdbTp;    /* for upward compatib.*/

/*-----------------------------------------------------------------**
**                          Bind (Connection) Structure**
**-----------------------------------------------------------------*/

typedef struct
{
  unsigned char hostname[FTIMXRHN];/* Remote Host Name*/
  unsigned char username[FTIMXSTR];/* User Name*
                                                  * (on remote host)*/
  unsigned char password[FTIMXSTR];/* User Password*
                                                  * (on remote host)*/
  unsigned char account[FTIMXSTR];/* Account to be charged*
                                                  * (on remote host)*/
} ftibndT, *ftibndTp;

typedef ftibndT bndT;      /* for upward compatib.*/
typedef ftibndTp bndTp;    /* for upward compatib.*/

/*-----------------------------------------------------------------**
**                          File Transfer Structure       **
**-----------------------------------------------------------------*/

typedef struct
{
  int               doctyp;                     /* Document Type        */
  int               override;           /* Creation Override    */
  int               mode;                       /* Transfer Mode        */
  unsigned characcnt[FTIMXSTR];/* Account Name         */
  unsigned charlocname[FTIMXDIR];/* Local File Name      */
  unsigned charremname[FTIMXDIR];/* Remote File Name     */
  unsigned charfilpssd[FTIMXSTR];/* File Password        */
  unsigned charreqattri[3];   /* Requested Attributes*/
```

```
#ident "@(#)ftiuser.h 1.14 94/01/11 SMI"
                        /* docparm of NBS_9     */
  unsigned chardform;         /* display format*
                                          * (temporary for NBS-9)*/
  int               frfadu;             /* Start from FADU no.*
                                                  (FTAM-2) */
  int               tofadu;             /* Ending at FADU no.*
                                          *         (FTAM-2)*/
  unsigned charfadui;         /* FADU Id.         */
  int               fadun;              /* FADU No.         */
  unsigned characsctx;        /* Access Context*/
  long              result;             /* No. of bytes *
                                          * ( sent/received )*/
  double  elapse;             /* Elapsed Time     */
  double  speed;              /* Speed            */
  int               recmode;            /* Recovery Mode*/
  int               trfid;              /* Transfer Identifier*/
  long              filsize;            /* Future file size*/
} ftitrfT, *ftitrfTp;

/*----------------------------------------------------------------**
**                           Create Structure              **
**----------------------------------------------------------------*/

typedef struct
{
 long               filsize;            /* Future file size*/
 int                override;           /* Override parameter*/
 int                doctyp;                      /* Document type*/
 int                avail;                       /* File availability*/
 unsigned charperact[2];      /* Permitted actions (2)*/
 unsigned characcnt[FTIMXSTR];/* Account Name*/
 unsigned charfilname[FTIMXDIR];/* Remote File Name*/
 unsigned charfilpssd[FTIMXSTR];/* Create password*/
 unsigned characctrl[FTIMXSTR];/* Access ctrl elt. attr*/
 unsigned charshfname[FTIMXDIR];/* Create Shadow File*/
} fticreT, *fticreTp;

typedef fticreT creT;      /* for upward compatib.*/
typedef fticreTp creTp;    /* for upward compatib.*/

/*----------------------------------------------------------------**
**                           Delete Structure              **
**----------------------------------------------------------------*/
```

```
#ident "@(#)ftiuser.h 1.14 94/01/11 SMI"
typedef struct
{
 unsigned characcnt[FTIMXSTR];/* Account*/
 unsigned charfilname[FTIMXDIR];/* File name*/
 unsigned charfilpssd[FTIMXSTR];/* Delete password*/
} ftidelT, *ftidelTp ;

typedef ftidelT delT;      /* for upward compatib.*/
typedef ftidelTp delTp;    /* for upward compatib.*/

/*-----------------------------------------------------------------**
**                            Rename Structure                     **
**-----------------------------------------------------------------*/

typedef struct
{
 unsigned characcnt[FTIMXSTR];/* Account              */
 unsigned charfilname[FTIMXDIR];/* File name            */
 unsigned charfilpssd[FTIMXSTR];/* Change attr. password*/
 unsigned charnewname[FTIMXDIR];/* New File name */
} ftirenT, *ftirenTp ;

typedef ftirenT renT;      /* for upward compatib.*/
typedef ftirenTp renTp;    /* for upward compatib.*/

/*-----------------------------------------------------------------**
**                            File Read Attribute Structure**
**-----------------------------------------------------------------*/

typedef struct
{
 unsigned characcnt[FTIMXSTR];        /* Account*/
 unsigned charfilname[FTIMXDIR];      /* File name*/
 unsigned charfilpssd[FTIMXSTR];      /* Read attr. password*/
 unsigned charreqattri[3];            /* Requested attributes*/
} ftiratT, *ftiratTp ;

typedef ftiratT ratT;      /* for upward compatib.*/
typedef ftiratTp ratTp;    /* for upward compatib.*/

/*-----------------------------------------------------------------**
**                            File Read Attribute Response Structure**
**-----------------------------------------------------------------*/
```

```
#ident "@(#)ftiuser.h 1.14 94/01/11 SMI"
typedef struct
{
 unsigned charfilname[FTIMXDIR];/* File name*/
 unsigned charperact[2];      /* Permitted actions (2)*/
 int                doctyp;                      /* Document type*/
 ftidtpT  parmdoc;            /* Document type param.*/
 unsigned characcnt[FTIMXSTR];/* Storage account*/
 unsigned chardtcreat[FTIMXDAT];/* Date and Time of *
                                                 *        CREATION*/
 unsigned chardtlmodi[FTIMXDAT];/* of LAST MODIFICATION */
 unsigned chardtlracc[FTIMXDAT];/* of LAST READ ACCESS*/
 unsigned chardtlatmo[FTIMXDAT];/* of LAST ATTR. MODIF.*/
 unsigned charidcreat[FTIMXSTR];/* Identity of CREATOR*/
 unsigned charidlmodi[FTIMXSTR];/* of LAST MODIFIER*/
 unsigned charidlread[FTIMXSTR];/* of LAST READER*/
 unsigned charidlatmo[FTIMXSTR];/* of LAST ATTR.MODIFIER*/
 int                avail;                       /* File availability    */
 long               filsize;           /* File size            */
 long               futfsize;          /* Future file size     */
 unsigned characctrl[FTIMXSTR];/* Access ctrl elt.attr.*/
 unsigned charlegqua[FTIMXSTR];/* Legal Qualification*/
 unsigned charpriuse[FTIMXSTR];/* Private use */
} ftirasT, *ftirasTp ;

/*--------------------------------------------------------------**
**                           Change Local Directory Name Structure**
**--------------------------------------------------------------*/

typedef struct
{
 unsigned chardirname[FTIMXDIR];/* directory to change to
                                                          (input)*/
} ftilcdT, *ftilcdTp ;

/*--------------------------------------------------------------**
**                           Change Remote Directory Name Structure**
**--------------------------------------------------------------*/

typedef struct
{
 unsigned chardirname[FTIMXDIR];/* directory to change to*/
} ftircdT, *ftircdTp ;

/*--------------------------------------------------------------**
```

```
#ident "@(#)ftiuser.h 1.14 94/01/11 SMI"
**                          Get Remote Directory Name Structure**
**-----------------------------------------------------------------*/

typedef struct
{
 unsigned chardirname[FTIMXDIR];/* directory name (OUT)*/
} ftipwdT, *ftipwdTp ;

/*-----------------------------------------------------------------**
**                          Docket Structure (Recovery)**
**-----------------------------------------------------------------*/

typedef struct
{                                                /* Structure docket*/
 unsigned charir ;                    /* 'i' / 'r'        */
 int                id ;                         /* Unique id     */
 unsigned charusername[FTIMXSTR];/* User name *
                                                  * ( on remote host )*/
 unsigned charlocname[FTIMXDIR] ;/* Local File Name*/
 unsigned charhostname[FTIMXRHN];/* Remote hostname*/
 long               cpi ;              /* Last NOT ack checkpt*/
 long               acpi ;                       /* Last ack checkpt*/
 unsigned charact_state ;     /* Activity state :*
                                                  * starting, in progress*
                               * data transfer end,*
                                                  *        finished       */
 unsigned chardoctyp ;        /* Document type*/
 long               result;                      /* No. of bytes *
                                                  * ( sent / received )*/
 double             elapse;                      /* Elapsed Time  */
 double             speed;                       /* Speed         */
} ftidockT, *ftidockTp ;

#ifdef  __cplusplus
}
#endif

#endif    /* _FTIUSER_H */
```

A

FTAM Responder Header (`ftruser.h`*)*

The content of the C header file `ftruser.h`, which must be included in all applications developed using the FTAM Responder interface is listed on the following pages:

```
#ident "@(#)ftruser.h 1.8 93/10/22 SMI"
/*
          Copyright 1993 Sun Microsystems, Inc. All Rights Reserved

          (c) Copyright 1991 Marben Produit. All rights reserved
*/

#ifndef   _FTRUSER_H
#define   _FTRUSER_H

#ifdef  __cplusplus
extern "C" {
#endif

/*-----------------------------------------------------------------**
**                            General Constants                    **
**-----------------------------------------------------------------*/

#include  <sys/param.h>/* used for MAXPATHLEN*/
#define FTRMXDIR  MAXPATHLEN/* MaXimum DIRectory Length    */
#define FTRMXSTR 128/* MaX General String Length    */
#define FTRMXDFD 128/* MaX FTAM Diagnostic*
                                        * Further Details  */

/*-----------------------------------------------------------------**
**                            FTR Execution Status                 **
**-----------------------------------------------------------------*/

#define FTRSTNER   0/* No ERror                      */
#define FTRSTFTM   3/* FTaM service or protocol err */

/*-----------------------------------------------------------------**
**        Association defines  - For Connect and Disconnect**
**-----------------------------------------------------------------*/

#define FTRASNER   1/* Association closed success.  */
#define FTRASABT   2/* Association ABorTed          */

/*-----------------------------------------------------------------**
```

```
#ident "@(#)ftruser.h 1.8 93/10/22 SMI"
**        Transfer defines     - For Transfer End function**
**-------------------------------------------------------------*/

#define FTRTRNER   1/* TRansfer ended successfully, *
                                        * No ERrors               */
#define FTRTRCAN   2/* TRansfer CANcelled           */

/*-------------------------------------------------------------**
**                            Passwords  types              **
**-------------------------------------------------------------*/

#define FTRGraStr 0x00/* GraphicString  */
#define FTROctStr 0x01/* OctetString              */


/*-------------------------------------------------------------**
**                            Document types                  **
**-------------------------------------------------------------*/

#define FTMDTFT1  1 /* FTAM_1  Document Type*/
#define FTMDTFT2  2 /* FTAM_1  Document Type*/
#define FTMDTFT3  3 /* FTAM_1  Document Type*/
#define FTMDTNB9  9 /* NBS_9   Document Type*/
#define FTMDTIN1 10 /* INTAP_1 Document Type*/

/*-------------------------------------------------------------**
**                            Override Parameters           **
**-------------------------------------------------------------*/

#define FTMOVNOC 0  /* NO Creation, Create Failure*/
#define FTMOVSOF 1  /* Select Old File  */
#define FTMOVDOA 2  /* Deleted and created OLD attr.*/
#define FTMOVDNA 3  /* Deleted and created NEW attr.*/

/*-------------------------------------------------------------**
**        FTAM Requested Access - For Select and Create**
**-------------------------------------------------------------*/

#define FTMACREA  0x80/* Required Access: READ*/
#define FTMACCINS 0x40/*    "        "  : INSERT*/
#define FTMACREP  0x20/*    "        "  : REPLACE*/
#define FTMACEXT  0x10/*    "        "  : EXTEND*/
#define FTMACERA  0x08/*    "        "  : ERASE*/
#define FTMACRAT  0x04/*    "        "  : READ ATTRIB.*/
```

```
#ident "@(#)ftruser.h 1.8 93/10/22 SMI"
#define FTMACCAT  0x02/*    "        "  : CHANGE ATTR.*/
#define FTMACDEL  0x01/*    "        "  : DELETE*/

/*---------------------------------------------------------------**
**                          Permitted Actions                    **
**---------------------------------------------------------------*/

#define FTMPAREA  0x80/* READ                    */
#define FTMPAINS  0x40/* INSERT                  */
#define FTMPAREP  0x20/* REPLACE                 */
#define FTMPAEXT  0x10/* EXTEND                  */
#define FTMPAERA  0x08/* ERASE                   */
#define FTMPARAT  0x04/* READ-ATTRIBUTE */
#define FTMPACAT  0x02/* CHANGE ATTRIBUTE*/
#define FTMPADEL  0x01/* DELETE file             */

/*---------------------------------------------------------------**
**                  Execution Status Report Structure**
**                  - for NEGATIVE Reponder's API answer -**
**---------------------------------------------------------------*/

typedef struct
{
  int               status;                     /* Execution status*/
  int               diag;                       /* FTAM diagnostic code*/
  unsigned charftamdfd[FTRMXDFD];/* FTAM Diagnostic *
                                                 * - Further Details*/
} ftresrT, *ftresrTp ;

/*---------------------------------------------------------------**
**                          Bind (Connection) Structure**
**---------------------------------------------------------------*/

typedef struct
{
  unsigned char username[FTRMXSTR];/* User Name*
                                                 * (on remote host)*/
  unsigned char password[FTRMXSTR];/* User Password*
                                                 * (on remote host)*/
  unsigned char account[FTRMXSTR];/* Account to be charged*
                                                 * (on remote host)*/
} ftrbndT, *ftrbndTp;
```

```
#ident "@(#)ftruser.h 1.8 93/10/22 SMI"
/*--------------------------------------------------------------**
**                          Select Structure                    **
**--------------------------------------------------------------*/

typedef struct
{
  int               doctyp;                       /* Document Type*/
  int               reqaccess;          /* Requested Access*/
  unsigned char filname[FTRMXDIR];/* File name*/
  unsigned char account[FTRMXSTR];/* Account to be charged*
                                                    * ( on remote host )*/
  /* The following passwords are reserved for future extensions*/
  unsigned char reafilpssd[FTRMXSTR];/* Passw for READ Access*/
  unsigned char insfilpssd[FTRMXSTR];/* Passw for INSERT*/
  unsigned char refilpssd[FTRMXSTR];/* Passw for REPLACE*/
  unsigned char exfilpssd[FTRMXSTR];/* Passw for EXTEND*/
  unsigned char erfilpssd[FTRMXSTR];/* Passw for ERASE*/
  unsigned char rafilpssd[FTRMXSTR];/* Passw for READ ATTR  */
  unsigned char cafilpssd[FTRMXSTR];/* Passw for CHANGE ATTR*/
  unsigned char defilpssd[FTRMXSTR];/* Passw for DELETE*/
} ftrselT, *ftrselTp ;

/*--------------------------------------------------------------**
**                          Create Structure                    **
**--------------------------------------------------------------*/

typedef struct
{
  int               doctyp;                       /* Document Type*/
  int               reqaccess;          /* Requested Access*/
  long              filsize;            /* Future File Size*/
  int               override;           /* Override Parameter*/
  unsigned char peract[2];/* Permitted Actions*/
  unsigned char filname[FTRMXDIR];/* File Name*/
  unsigned char account[FTRMXSTR];/* Account to be charged*
                                                    * ( on remote host )*/
  /* The following passwords are reserved for future extensions*/
  unsigned char cfilpssd[FTRMXSTR];/* Passw for CREATE Acc*/
  unsigned char reafilpssd[FTRMXSTR];/* Passw for READ Access*/
  unsigned char insfilpssd[FTRMXSTR];/* Passw for INSERT     */
  unsigned char refilpssd[FTRMXSTR];/* Passw for REPLACE    */
  unsigned char exfilpssd[FTRMXSTR];/* Passw for EXTEND     */
  unsigned char erfilpssd[FTRMXSTR];/* Passw for ERASE      */
  unsigned char rafilpssd[FTRMXSTR];/* Passw for READ ATTR  */
```

```
#ident "@(#)ftruser.h 1.8 93/10/22 SMI"
  unsigned char cafilpssd[FTRMXSTR];/* Passw for CHANGE ATTR*/
  unsigned char defilpssd[FTRMXSTR];/* Passw for DELETE     */
} ftrcreT, *ftrcreTp ;

/*---------------------------------------------------------------**
**                            Recover Structure                  **
**---------------------------------------------------------------*/

typedef struct
{
  int               trfid ;                      /* Transfer Identifier*/
  int               doctyp ;            /* Document Type*/
  int               reqaccess ;        /* Requested Access*/
  unsigned char filname[FTRMXDIR];/* File Name*/
  unsigned char account[FTRMXSTR];/* Account to be charged*
                                                  * ( on remote host )*/
  /* The following passwords are reserved for future extensions*/
  unsigned char reafilpssd[FTRMXSTR];/* Passw for READ access*/
  unsigned char insfilpssd[FTRMXSTR];/* Passw for INSERT*/
  unsigned char refilpssd[FTRMXSTR];/* Passw for REPLACE*/
  unsigned char exfilpssd[FTRMXSTR];/* Passw for EXTEND*/
  unsigned char erfilpssd[FTRMXSTR];/* Passw for ERASE*/
  unsigned char rafilpssd[FTRMXSTR];/* Passw for READ ATTR*/
  unsigned char cafilpssd[FTRMXSTR];/* Passw for CHANGE ATTR*/
  unsigned char defilpssd[FTRMXSTR];/* Passw for DELETE*/
} ftrrcoT, *ftrrcoTp ;

/*---------------------------------------------------------------**
**                            Read - Write  Structure            **
**---------------------------------------------------------------*/

typedef struct
{
  int trfid ;                           /* Transfer identifier*/
} ftrtrfT, *ftrtrfTp ;

/*---------------------------------------------------------------**
**                            Responder API Global Structure**
**---------------------------------------------------------------*/
/* Union of all structures */

typedef union
{
  ftrbndT  open;              /* Bind Structure*/
```

```
#ident "@(#)ftruser.h 1.8 93/10/22 SMI"
  ftrselT  sel ;              /* Select Structure*/
  ftrcreT  cre ;              /* Create Structure*/
  ftrrcoT  reco ;             /* Recovery Structure*/
  ftrtrfT  trf ;              /* Transfer Structure*/
} ftrapiT, *ftrapiTp ;

#ifdef  __cplusplus
}
#endif

#endif    /* _FTRUSER_H */
```

A

Index

R

S

T

U

V

W