2.3.2 Diagnostics

An FTAM diagnostic is represented by the `FTAMdiagnostic` structure.

```
struct FTAMdiagnostic {
    int     ftd_type;

    int     ftd_identifier;

    int     ftd_observer;
    int     ftd_source;

    int     ftd_delay;

#define FTD_SIZE        512
    int     ftd_cc;
    char    ftd_data[FTD_SIZE];
};
```

This elements of this structure are:

`ftd_type`: the diagnostic type, one of:

Value	Meaning
DIAG_INFORM	informative
DIAG_TRANS	transient
DIAG_PERM	permanent

`ftd_identifier`: the error-identifier (consult `<isode/ftam.h>` for the list, there are far too many to list here);

`ftd_observer/ftd_source`: the observer and source of the error, one of:

Value	Meaning
EREF_NONE	no categorization possible
EREF_IFSU	initiating file service user
EREF_IFPM	initiating file service machine
EREF_SERV	service support the file protocol machine
EREF_RFPM	responding file protocol machine
EREF_RFSU	responding file service user

`ftd_delay`: the suggested delay that might be honored prior to retrying the operation (use `DIAG_NODELAY` if a delay is not appropriate); and,

`ftd_data/ftd_cc`: a diagnostic string (and the length of that string).

A limitation of this implementation is that the routines and data structures which support diagnostics have a fixed limit as to the number of diagnostics permitted. This is currently denoted by the manifest constant `NFDIAG`.

2.3.3 Charging

A list of FTAM charges is represented by the `FTAMcharging` structure.

```
struct FTAMcharging {
    int     fc_ncharge;

#define NFCHRG  5
    struct fc_charge {
        char   *fc_resource;
        char   *fc_unit;
        int     fc_value;
    }           fc_charges[NFCHRG];
};
```

This elements of this structure are:

`fc_charges/fc_ncharge`: the charging list (and the number of charges in the list, which may not exceed the manifest constant `NFCHRG`). Each charge consists of:

`fc_resource`: the resource identifier;

`fc_unit`: the charging unit; and,

`fc_value`: the charging value.

A limitation of this implementation is that a fixed number of charges may be supported, as denoted by the manifest constant `NFCHRG`.

2.3.4 Passwords

A list of FTAM passwords is represented by the `FTAMpasswords` structure.

```
struct FTAMpasswords {
    char   *fp_read;
    int     fp_readlen;

    char   *fp_insert;
    int     fp_insertlen;

    char   *fp_replace;
    int     fp_replacelen;

    char   *fp_extend;
    int     fp_extendlen;

    char   *fp_erase;
    int     fp_eraselen;

    char   *fp_readattr;
    int     fp_readattrlen;

    char   *fp_chngattr;
    int     fp_chngattrlen;

    char   *fp_delete;
    int     fp_deletelen;
};
```

This elements of this structure are:

`fp_read/fp_readlen`: the read password (and the length of the password);

`fp_insert/fp_insertlen`: the insert password (and the length of the password);

`fp_replace/fp_replacelen`: the replace password (and the length of the password);

`fp_extend/fp_extendlen:` the extend password (and the length of the password);

`fp_erase/fp_eraselen:` the erase password (and the length of the password);

`fp_readattr/fp_readattrlen:` the read attribute password (and the length of the password);

`fp_chngattr/fp_chngattrlen:` the change attribute password (and the length of the password); and,

`fp_delete/fp_deletelen:` the delete password (and the length of the password).

The `FPFREE` macro can be used to free the storage associated with an `FTAMpasswords` structure (without freeing the structure itself). It behaves as if it was defined as:

```
void    FPFREE (fp)
struct FTAMpasswords *fp;
```

2.3.5 Access Control

An FTAM access control element is represented by the `FTAMacelement` structure.

```
struct FTAMacelement {
    int     fe_actions;

    struct FTAMconcurrency fe_concurrency;

    char   *fe_aet;

    struct FTAMpasswords fe_passwords;

    AEI    fe_aet;

    struct FTAMacelement *fe_next;
};
```

Requested Access	
FA_PERM_READ	read fadu
FA_PERM_INSERT	insert fadu
FA_PERM_REPLACE	replace fadu
FA_PERM_EXTEND	extend fadu
FA_PERM_ERASE	erase fadu
FA_PERM_READATTR	read attribute
FA_PERM_CHNGATTR	change attribute
FA_PERM_DELETE	delete file

Table 2.1: FTAM Requested Access Values

The elements of this structure are:

`fe_actions:` the action list, containing the inclusive-or of any of the values shown in Table 2.1;

`fe_concurrency:` any concurrency-control constraints;

`fe_aet:` the user identity;

`fe_passwords` the passwords;

`fe_aet:` the application-entity title (defined in Section 2.2.1 on page 17 in *Volume One*); and,

`fe_next:` the next access element in the linked list.

The `FEFREE` macro can be used to free the storage associated with a list of FTAM access elements, including the head element on the list. It behaves as if it was defined as:

```
void    FEFREE (fe)
struct FTAMacelement *fe;
```

2.3.6 Attributes

An FTAM attribute list is represented by the `FTAMattributes` structure.

```
struct FTAMattributes {
    long    fa_present;
    long    fa_novalue;

#define NFFILE  5
    int     fa_nfile;
    char   *fa_files[NFFILE];

    int     fa_permitted;

    OID     fa_contents;
    PE      fa_parameter;

    char   *fa_account;

    struct UTCtime fa_date_create;
    struct UTCtime fa_date_modify;
    struct UTCtime fa_date_read;
    struct UTCtime fa_date_attribute;

    char   *fa_id_create;
    char   *fa_id_modify;
    char   *fa_id_read;
    char   *fa_id_attribute;

    int     fa_availability;

    int     fa_filesize;
    int     fa_futuresize;

    struct FTAMacelement fa_control;
    char   *fa_legal;
};
```

The elements of this structure are:

`fa_present`: the values present, containing the inclusive-or of any of the values shown in Table 2.2;

`fa_novalue`: the values which are not available are not available (using the same values listed in Table 2.2, the value of this element is a subset of the value of the `fa_present` element);

`fa_files/fa_nfile`: the list of filename components (and the number of components, which may not exceed the manifest constant `NFFILE`);

`fa_permitted`: permitted actions, containing the inclusive-or of any of the values shown in Table 2.1 on page 23, in addition to any of:

Value	Meaning
FA_PERM_TRAV	traversal
FA_PERM_RVTRAV	reverse-traversal
FA_PERM_RANDOM	random-order

`fa_contents/fa_parameter`: the contents type;

`fa_account`: the account;

`fa_date_create`: the date and time of creation;

`fa_date_modify`: the date and time of last modification;

`fa_date_read`: the date and time of last read access;

`fa_date_attribute`: the date and time of last attribute modification;

`fa_id_create`: the identity of creator;

`fa_id_modify`: the identity of last modifier;

`fa_id_read`: the identity of last reader;

`fa_id_attribute`: the identity of last attribute modifier;

`fa_availability`: file availability, one of:

Value	Meaning
`FA_AVAIL_IMMED`	immediate
`FA_AVAIL_DEFER`	deferred

`fa_filesize`: filesize;

`fa_futuresize`: future filesize;

`fa_control`: access control;

`fa_encrypt`: encryption name; and,

`fa_legal`: legal qualification.

A limitation of this implementation is that a fixed number of filename components may be supported, as denoted by the manifest constant `NFFILE`.

The routine `FAFREE` can be used to free the storage associated with an `FTAMattributes` structure (without freeing the structure itself). It is defined as:

```
void    FAFREE (fa)
struct FTAMattributes *fa;
```

2.3.7 Concurrency

An FTAM concurrency list is represented by the `FTAMconcurrency` structure.

```
struct FTAMconcurrency {
#define FLOCK_SHARED    00      /* shared */
#define FLOCK_EXCLUSIVE 01      /* exclusive */
#define FLOCK_NOTREQD   02      /* not-required */
#define FLOCK_NOACCESS  03      /* no-access */

    char    fc_readlock;
    char    fc_insertlock;
    char    fc_replacelock;
    char    fc_extendlock;
```

Attribute	
FA_FILENAME	filename
FA_ACTIONS	permitted actions
FA_CONTENTS	contents type
FA_ACCOUNT	account
FA_DATE_CREATE	date and time of creation
FA_DATE_MODIFY	date and time of last modification
FA_DATE_READ	date and time of last read access
FA_DATE_ATTR	date and time of last attribute modification
FA_ID_CREATE	identity of creator
FA_ID_MODIFY	identity of last modifier
FA_ID_READ	identity of last reader
FA_ID_ATTR	identity of last attribute modifier
FA_AVAILABILITY	file availability
FA_FILESIZE	filesize
FA_FUTURESIZE	future filesize
FA_CONTROL	access control
FA_LEGAL	legal qualifications
FA_PRIVATE	private use

Table 2.2: FTAM Attribute Values

```
    char    fc_eraselock;
    char    fc_readattrlock;
    char    fc_chngattrlock;
    char    fc_deletelock;
};
```

This elements of this structure are:

`fc_readlock:` the concurrency constraints for the read operation;

`fc_insertlock:` the concurrency constraints for the insert operation;

`fc_replacelock:` the concurrency constraints for the replace operation;

`fc_extendlock:` the concurrency constraints for the extend operation;

`fc_eraselock:` the concurrency constraints for the erase operation;

`fc_readattrlock:` the concurrency constraints for the read attribute operation;

`fc_chngattrlock:` the concurrency constraints for the change attribute operation; and,

`fc_deletelock:` the concurrency constraints for the delete operation.

The `FCINIT` macro can be used to initialize an `FTAMconcurrency` structure. It behaves as if it was defined as:

```
void    FCINIT (fc)
struct FTAMconcurrency *fc;
```

2.3.8 FADU Identity

An FADU identity is represented by the `FADUidentity` structure.[1]

[1]FADU stands for File Access Data Unit; if you didn't already know this, you shouldn't be reading this chapter. Read Section 2.1 right now.