Note that the data contained in the structure was allocated via *malloc*(3), and should be released with the FTRAFREE macro when no longer needed. The FTRAFREE macro behaves as if it was defined as:

```
void    FTRAFREE (ftra)
struct FTAMreadattr *ftra;
```

The macro frees only the data allocated by FWaitRequest, and not the FTAMreadattr structure itself. Note that the macro FTGFREE when applied to the group, will call FTRAFREE as appropriate.

If the flag FTG_CHNGATTR is present, then ftg_chngattr element contains an F-CHANGE-ATTRIB event encoded in an FTAMchngattr structure:

```
struct FTAMchngattr {
    int ftca_action;

    struct FTAMattributes ftca_attrs;

    int     ftca_ndiag;
    struct FTAMdiagnostic ftca_diags[NFDIAG];
};
```

The elements of this structure are:

ftca_action: the action result (.RESPONSE only);

ftca_attrs: the attribute values to change (in a .REQUEST), and possibly the new values (in a .RESPONSE); and,

ftca_diags/ftca_ndiag: any diagnostic information (and the number of diagnostics present, .RESPONSE only).

Note that the data contained in the structure was allocated via *malloc*(3), and should be released with the FTCAFREE macro when no longer needed. The FTCAFREE macro behaves as if it was defined as:

```
void    FTCAFREE (ftca)
struct FTAMchngattr *ftca;
```

The macro frees only the data allocated by `FWaitRequest`, and not the `FTAMchngattr` structure itself. Note that the macro `FTGFREE` when applied to the group, will call `FTCAFREE` as appropriate.

If the flag `FTG_OPEN` is present, then `ftg_open` element contains an F-OPEN event encoded in an `FTAMopen` structure:

```
struct FTAMopen {
    int     ftop_state;
    int     ftop_action;

    int     ftop_mode;

    OID     ftop_contents;
    PE      ftop_parameter;
    struct FTAMconcurrency ftop_conctl;
    PE      ftop_sharedASE;

    int     ftop_locking;

    int     ftop_ndiag;
    struct FTAMdiagnostic ftop_diags[NFDIAG];
};
```

The elements of this structure are:

`ftop_state`: the state result (.RESPONSE only);

`ftop_action`: the action result (.RESPONSE only);

`ftop_mode`: the processing mode, containing the inclusive-or of any of the values shown in Table 2.1 on page 23 (.REQUEST only);

`ftop_contents/ftop_parameter`: the contents type;

`ftop_conctl`: concurrency control;

`ftop_sharedASE`: shared ASE information;

`ftop_locking`: enable FADU locking (.REQUEST only); and,

`ftop_diags/ftop_ndiag`: any diagnostic information (and the number of diagnostics present, .RESPONSE only).

Note that the data contained in the structure was allocated via *malloc*(3), and should be released with the `FTOPFREE` macro when no longer needed. The `FTOPFREE` macro behaves as if it was defined as:

```
void    FTOPFREE (ftop)
struct FTAMopen *ftop;
```

The macro frees only the data allocated by `FWaitRequest`, and not the `FTAMopen` structure itself. Note that the macro `FTGFREE` when applied to the group, will call `FTOPFREE` as appropriate.

If the flag `FTG_CLOSE` is present, then `ftg_close` element contains an F-CLOSE event encoded in an `FTAMclose` structure:

```
struct FTAMclose {
    int     ftcl_action;

    PE      ftcl_sharedASE;

    int     ftcl_ndiag;
    struct FTAMdiagnostic   ftcl_diags[NFDIAG];
};
```

The elements of this structure are:

`ftcl_action`: the action result;

`ftcl_sharedASE`: shared ASE information; and,

`ftcl_diags/ftcl_ndiag`: any diagnostic information (and the number of diagnostics present).

Note that the data contained in the structure was allocated via *malloc*(3), and should be released with the `FTCLFREE` macro when no longer needed. The `FTCLFREE` macro behaves as if it was defined as:

```
void    FTCLFREE (ftcl)
struct FTAMclose *ftcl;
```

The macro frees only the data allocated by `FWaitRequest`, and not the `FTAMclose` structure itself. Note that the macro `FTGFREE` when applied to the group, will call `FTCLFREE` as appropriate.

If the flag `FTG_DESELECT` is present, then `ftg_deselect` element contains an F-DESELECT event encoded in an `FTAMdeselect` structure:

```
struct FTAMdeselect {
    int     ftde_action;

    PE      ftde_sharedASE;

    struct FTAMcharging ftde_charges;
    int     ftde_ndiag;
    struct FTAMdiagnostic   ftde_diags[NFDIAG];
};
```

The elements of this structure are:

`ftde_action`: the action result (.RESPONSE only);

`ftde_sharedASE`: shared ASE information;

`ftde_charges`: any charges (.RESPONSE only); and,

`ftde_diags/ftde_ndiag`: any diagnostic information (and the number of diagnostics present, .RESPONSE only).

Note that the data contained in the structure was allocated via *malloc*(3), and should be released with the `FTDEFREE` macro when no longer needed. The `FTDEFREE` macro behaves as if it was defined as:

```
void    FTDEFREE (ftde)
struct FTAMdeselect *ftde;
```

The macro frees only the data allocated by `FWaitRequest`, and not the `FTAMdeselect` structure itself. Note that the macro `FTGFREE` when applied to the group, will call `FTDEFREE` as appropriate.

If the flag `FTG_DELETE` is present, then `ftg_delete` element contains an F-DELETE event encoded in an `FTAMdelete` structure:

```
struct FTAMdelete {
    int     ftxe_action;

    PE      ftxe_sharedASE;

    struct FTAMcharging ftxe_charges;
    int     ftxe_ndiag;
    struct FTAMdiagnostic   ftxe_diags[NFDIAG];
};
```

The elements of this structure are:

`ftxe_action`: the action result (.RESPONSE only);

`ftxe_sharedASE`: shared ASE information;

`ftxe_charges`: any charges (.RESPONSE only); and,

`ftxe_diags/ftxe_ndiag`: any diagnostic information (and the number of diagnostics present, .RESPONSE only).

Note that the data contained in the structure was allocated via *malloc*(3), and should be released with the `FTXEFREE` macro when no longer needed. The `FTXEFREE` macro behaves as if it was defined as:

```
void    FTXEFREE (ftxe)
struct FTAMdeltee *ftxe;
```

The macro frees only the data allocated by `FWaitRequest`, and not the `FTAMdelete` structure itself. Note that the macro `FTGFREE` when applied to the group, will call `FTXEFREE` as appropriate.

Access Indications

When an event associated with file access occurs, the `fti_type` field of the `fti` parameter contains the value `FTI_ACCESS`. Further, an `FTAMaccess` structure is contained inside the `fti` parameter, indicating an F-LOCATE or F-ERASE event.

```
struct FTAMaccess {
    int     ftac_operation;

    int     ftac_action;

    struct FADUidentity ftac_identity;

    int     ftac_locking;

    int     ftac_ndiag;
    struct FTAMdiagnostic   ftac_diags[NFDIAG];
};
```

Access Operations	
FA_OPS_LOCATE	F-LOCATE
FA_OPS_ERASE	F-ERASE

Table 2.3: FTAM Access Operations

The elements of this structure are:

`ftac_operation`: the operation, one of the values shown in Table 2.3;

`ftac_action`: the action result (.CONFIRMATION only);

`ftac_identity`: the FADU identity (either .INDICATION and optionally on F-LOCATE.CONFIRMATION);

`ftac_locking`: FADU locked (F-LOCATE.INDICATION only); and,

`ftac_diags/ftac_ndiag`: any diagnostic information (and the number of diagnostics present, .CONFIRMATION only).

Note that the data contained in the structure was allocated via *malloc*(3), and should be released with the `FTACFREE` macro when no longer needed. The `FTACFREE` macro behaves as if it was defined as:

```
void    FTACFREE (ftac)
struct FTAMaccess *ftac;
```

The macro frees only the data allocated by `FWaitRequest`, and not the `FTAMaccess` structure itself.

Read/Write Indications

When an event associated with the beginning of bulk data transfer occurs, the `fti_type` field of the `fti` parameter contains the value `FTI_READWRITE`. Further, an `FTAMreadwrite` structure is contained inside the `fti` parameter, indicating an F-READ.INDICATION or F-WRITE.INDICATION event.

```
struct FTAMreadwrite {
    int     ftrw_operation;

    struct FADUidentity ftrw_identity;

    int     ftrw_context;
    int     ftrw_level;

    int     ftrw_locking;
};
```

The elements of this structure are:

`ftrw_operation`: the operation, one of the values shown in Table 2.4;

`ftrw_identity`: the FADU identity;

`ftrw_context/ftrw_level`: the access context, one of the values shown in Table 2.5 (F-READ.INDICATION only); and,

`ftrw_locking`: FADU locked.

Note that the data contained in the structure was allocated via *malloc*(3), and should be released with the `FTRWFREE` macro when no longer needed. The `FTRWFREE` macro behaves as if it was defined as:

```
void    FTRWFREE (ftrw)
struct FTAMreadwrite *ftrw;
```

The macro frees only the data allocated by `FWaitRequest`, and not the `FTAMreadwrite` structure itself.

Data Indications

When an event associated with user data occurs, the `fti_type` field of the `fti` parameter contains the value `FTI_DATA`, and an `PSAPdata` structure (described in Chapter 2 of *Volume Two*) is contained inside the `fti` parameter, indicating an F-DATA.INDICATION event. To distinguish between structure and contents, FTAM uses different presentation context identifiers. If the `pe_context` field of any presentation element contains the value

Data Operations	
FA_OPS_READ	F-READ
FA_OPS_INSERT	F-WRITE insert
FA_OPS_REPLACE	F-WRITE replace
FA_OPS_EXTEND	F-WRITE extend

Table 2.4: FTAM Data Operations

Access Contexts	
FA_ACC_HA	hierarchical all data units
FA_ACC_HN	hierarchical no data units
FA_ACC_FA	flat all data units
FA_ACC_FL	flat one level data units
FA_ACC_FS	flat single data unit
FA_ACC_UA	unstructured all data units
FA_ACC_US	unstructured single data unit

Table 2.5: FTAM Access Contexts

PE_DFLT_CONTEXT, then the element contains structuring information in the FTAM PCI, and the pe_id element should be consulted to determine the particular structuring information being conveyed:

Value	Meaning
FADU_NODESCR	node descriptor data element
FADU_ENTERTREE	enter subtree data element
FADU_EXITREE	exit subtree data element

Otherwise, the element contains file contents as defined in the PCI for the document type being transferred.

Data End Indication

When an event associated with the end of bulk data transfer occurs, the fti_type field of the fti parameter contains the value FTI_DATAEND. Further, an FTAMdataend structure is contained inside the fti parameter, indicating an F-DATA-END.INDICATION event.

```
struct FTAMdataend {
    int     ftda_action;

    int     ftda_ndiag;
    struct FTAMdiagnostic   ftda_diags[NFDIAG];
};
```

The elements of this structure are:

ftda_action: the action result; and,

ftda_diags/ftda_ndiag: any diagnostic information (and the number of diagnostics present).

Cancel Indications

When an event associated with the cancellation of bulk data transfer occurs, the fti_type field of the fti parameter contains the value FTI_CANCEL. Further, an FTAMcancel structure is contained inside the fti parameter, indicating an F-CANCEL event.

```
struct FTAMcancel {
    int     ftcn_action;

    PE      ftcn_sharedASE;

    int     ftcn_ndiag;
    struct FTAMdiagnostic   ftcn_diags[NFDIAG];
};
```

The elements of this structure are:

`ftcn_action`: the action result;

`ftcn_sharedASE`: shared ASE information; and,

`ftcn_diags/ftcn_ndiag`: any diagnostic information (and the number of diagnostics present).

Note that the data contained in the structure was allocated via *malloc*(3), and should be released with the `FTCNFREE` macro when no longer needed. The `FTCNFREE` macro behaves as if it was defined as:

```
void    FTCNFREE (ftcn)
struct FTAMcancel *ftcn;
```

The macro frees only the data allocated by `FWaitRequest`, and not the `FTAMcancel` structure itself.

Transfer End Indications

When an event associated with the termination of bulk data transfer occurs, the `fti_type` field of the `fti` parameter contains the value `FTI_TRANSEND`. Further, an `FTAMtransend` structure is contained inside the `fti` parameter, indicating an F-TRANSFER-END event.

```
struct FTAMtransend {
    int     ftre_action;

    PE      ftre_sharedASE;

    int     ftre_ndiag;
    struct FTAMdiagnostic   ftre_diags[NFDIAG];
};
```