`password/passlen:` the password (and its length), if any;

`qos:` the quality of service on the connection (see Section 4.6.2 in *Volume Two*);

`tracing:` the address of a tracing routine to be invoked when an FTAM event occurs (consult Section 2.5.3);

`ftc:` a pointer to an `FTAMconnect` structure, which is updated only if the call succeeds; and,

`fti:` a pointer to an `FTAMindication` structure, which is updated only if the call fails.

If the call to `FInitializeRequest` is successful (the F-INITIALIZE.CONFIRMATION event occurs), it returns information in the `ftc` parameter which is a pointer to an `FTAMconnect` structure.

```
struct FTAMconnect {
    int     ftc_sd;

    int     ftc_state;
    int     ftc_action;

    AEInfo  ftc_respondtitle;

    struct PSAPaddr ftc_respondaddr;

    int     ftc_manage;

    int     ftc_units;
    int     ftc_attrs;

    int     ftc_rollback;

    struct FTAMcontentlist ftc_contents;

    int     ftc_ndiag;
    struct FTAMdiagnostic  ftc_diags[NFDIAG];
```

```
        int     ftc_ssdusize;

        struct QOStype ftc_qos;
    };
```

The elements of an `FTAMconnect` structure are:

`ftc_sd`: the association-descriptor to be used to reference this association;

`ftc_state`: a state result

`ftc_action`: an action result

`ftc_respondtitle`: the responding application-entity title, if any;

`ftc_respondaddr`: the responding presentation address, if any;

`ftc_manage`: the negotiated use of presentation context management;

`ftc_units`: the negotiated functional units;

`ftc_attrs`: the negotiated attribute groups;

`ftc_rollback`: the negotiated rollback availability;

`ftc_contents`: the negotiated contents type list;

`ftc_diags/ftc_ndiag`: any diagnostic information (and the number of diagnostics present)

`ftc_ssdusize`: the largest atomic SSDU size that can be used on the underlying connection; and,

`ftc_qos`: the quality of service on the connection (see Section 4.6.2 in *Volume Two*).

If the call to `FInitializeRequest` is successful, and the `ftc_state` element is set to `FSTATE_SUCCESS`, then association establishment has completed. If the call is successful, but the `acc_result` element is not `FSTATE_SUCCESS`, the the association attempt has been rejected and the diagnostics present indicate the reason for the rejection. Otherwise, if the call fails then the

association is not established and the FTAMabort structure contained in the FTAMindication discriminated union has been updated.

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

```
void    FTCFREE (ftc)
struct FTAMconnect *ftc;
```

The macro frees only the data allocated by FInitializeRequest, and not the FTAMconnect structure itself. Further, FTCFREE should be called only if the call to the FInitializeRequest routine returned OK.

2.5 Event Handling

Once the association has been established, an association-descriptor is used to reference the connection. This is usually the first parameter given to any of the remaining routines in the *libftam*(3n) library. Further, the last parameter is usually a pointer to an FTAMindication structure (as described on page 34). If a call to one of these routines fails, then the structure is updated. Consult the FTAMabort element of the FTAMindication structure and look at the diagnostics (typically only one) present. When a diagnostic is reported by the provider, the ftd_type element indicates if the error is "fatal" by having the value DIAG_PERM. Other values, such as DIAG_TRANS indicate non-fatal errors (the association still exists). The most common non-fatal error to occur is FS_GEN_WAITING which indicates that an indication is waiting to be read.

The FWaitRequest routine is used to wait for an event to occur.

```
int     FWaitRequest (sd, secs, fti)
int     sd;
int     secs;
struct FTAMindication *fti;
```

The parameters to this procedure are:

sd: the association-descriptor;

`secs`: the maximum number of seconds to wait for the event (a value of `NOTOK` indicates that the call should block indefinitely, whereas a value of `OK` indicates that the call should not block at all, e.g., a polling action); and,

`fti`: a pointer to an `FTAMindication` structure, which is always updated.

If the call to `FWaitRequest` returns the manifest constant `NOTOK`, then the `FTAMabort` structure contained in the `FTAMindication` parameter `fti` contains the reason for the failure. Similarly, if the manifest constant `OK` is returned, an event is encoded in the `fti` parameter, depending on the value of the `fti_type` element:

Value	Event
FTI_FINISH	F-TERMINATE.INDICATION
FTI_MANAGEMENT	grouped management operation
FTI_BULKBEGIN	begin of grouped file transfer operation
FTI_BULKEND	end of grouped file transfer operation
FTI_ACCESS	F-LOCATE.INDICATION F-LOCATE.CONFIRMATION F-ERASE.INDICATION F-ERASE.CONFIRMATION
FTI_READWRITE	F-READ.INDICATION F-WRITE.INDICATION
FTI_DATA	F-DATA.INDICATION
FTI_DATAEND	F-DATA-END.INDICATION
FTI_CANCEL	F-CANCEL.INDICATION F-CANCEL.CONFIRMATION
FTI_TRANSEND	F-TRANSFER-END.INDICATION F-TRANSFER-END.CONFIRMATION

These are now discussed in turn.

Termination Indication

When an event associated with association release occurs, the `fti_type` field of the `fti` parameter contains the value `FTI_FINISH`. Further, an `FTAMfinish`

structure is contained inside the `fti` parameter, indicating an F-TERMINATE.INDICATION event.

```
struct FTAMfinish {
    int     ftf_sharedASE;
};
```

The elements of this structure are:

`ftf_sharedASE`: shared ASE information.

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

```
void    FTFFREE (ftf)
struct FTAMfinish *ftf;
```

Group Indications

Use of the grouping function unit is required in the current implementation, hence when a transfer or management indication occurs, the `fti_type` field of the `fti` parameter contains one of three values:

Value	Event
`FTI_MANAGEMENT`	grouped management operation
`FTI_BULKBEGIN`	begin of grouped file transfer operation
`FTI_BULKEND`	end of grouped file transfer operation

and an `FTAMgroup` structure is contained inside the `fti` parameter.

```
struct FTAMgroup {
    int     ftg_threshold;

    int     ftg_flags;

    union {
        struct FTAMselect   ftg_un1_select;
        struct FTAMcreate   ftg_un1_create;
        struct FTAMclose    ftg_un1_close;
    }                   ftg_un1;
```

```
#define ftg_select      ftg_un1.ftg_un1_select
#define ftg_create      ftg_un1.ftg_un1_create
#define ftg_close       ftg_un1.ftg_un1_close

    struct FTAMreadattr ftg_readattr;

    struct FTAMchngattr ftg_chngattr;

    union {
        struct FTAMdeselect ftg_un2_deselect;
        struct FTAMdelete   ftg_un2_delete;
        struct FTAMopen     ftg_un2_open;
    } ftg_un2;
#define ftg_deselect    ftg_un2.ftg_un2_deselect
#define ftg_delete      ftg_un2.ftg_un2_delete
#define ftg_open        ftg_un2.ftg_un2_open
};
```

The elements of this structure are:

`ftg_threshold`: the grouping threshold;

`ftg_flags`: the indications which are present, containing the inclusive-or of any of the values:

Value	Meaning
`FTG_BEGIN`	F-BEGIN-GROUP
`FTG_SELECT`	F-SELECT
`FTG_CREATE`	F-CREATE
`FTG_RDATTR`	F-READ-ATTRIB
`FTG_CHATTR`	F-CHANGE-ATTRIB
`FTG_OPEN`	F-OPEN
`FTG_CLOSE`	F-CLOSE
`FTG_DESELECT`	F-DESELECT
`FTG_DELETE`	F-DELETE
`FTG_END`	F-END-GROUP

`ftg_select`: an `FTAMselect` structure;

`ftg_create`: an `FTAMcreate` structure;

`ftg_close`: an `FTAMclose` structure;

`ftg_readattr`: an `FTAMreadattr` structure;

`ftg_chngattr`: an `FTAMchngattr` structure;

`ftg_deselect`: an `FTAMdeselect` structure;

`ftg_delete`: an `FTAMdelete` structure; and,

`ftg_open`: an `FTAMopen` structure.

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

```
void    FTGFREE (ftg)
struct FTAMgroup *ftg;
```

If the flag `FTG_SELECT` is present, then `ftg_select` element contains an F-SELECT event encoded in an `FTAMselect` structure:

```
struct FTAMselect {
    int     ftse_state;
    int     ftse_action;

    struct FTAMattributes ftse_attrs;

    int     ftse_access;
    struct FTAMpasswords ftse_pwds;
    struct FTAMconcurrency ftse_conctl;

    PE      ftse_sharedASE;

    char   *ftse_account;

    int     ftse_ndiag;
    struct FTAMdiagnostic ftse_diags[NFDIAG];
};
```

The elements of this structure are:

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

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

`ftse_attrs`: file attributes;

`ftse_access`: requested access, containing the inclusive-or of any of the values found in Table 2.1 (.REQUEST only);

`ftse_pwds`: access passwords (.REQUEST only);

`ftse_conctl`: concurrency control (.REQUEST only);

`ftse_sharedASE`: shared ASE information (.REQUEST only);

`ftse_account`: account (.REQUEST only); and,

`ftse_diags/ftse_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 `FTSEFREE` macro when no longer needed. The `FTSEFREE` macro behaves as if it was defined as:

```
void    FTSEFREE (ftse)
struct FTAMselect *ftse;
```

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

If the flag `FTG_CREATE` is present, then `ftg_create` element contains an F-CREATE event encoded in an `FTAMcreate` structure:

```
struct FTAMcreate {
    int     ftce_state;
    int     ftce_action;

    int     ftce_override;
    char   *ftce_create;
    int     ftce_crelen;
```

```
    struct FTAMattributes ftce_attrs;

    int     ftce_access;
    struct FTAMpasswords ftce_pwds;
    struct FTAMconcurrency ftce_conctl;

    PE      ftce_sharedASE;

    char   *ftce_account;

    int     ftce_ndiag;
    struct FTAMdiagnostic ftce_diags[NFDIAG];
};
```

The elements of this structure are:

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

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

`ftce_override`: the override setting, one of:

Value	Meaning
FOVER_FAIL	fail, if already exists
FOVER_SELECT	select, if already exists
FOVER_WRITE	zero-truncate, if already exists
FOVER_DELETE	delete, if already exists

(.REQUEST only);

`ftce_attrs`: file attributes;

`ftce_create/ftce_crelen`: the create password and its length, if any (.REQUEST only);

`ftce_access`: requested access, containing the inclusive-or of any of the values found in Table 2.1 (.REQUEST only);

`ftce_pwds`: access passwords (.REQUEST only);

`ftce_conctl`: concurrency control (.REQUEST only);

`ftce_sharedASE`: shared ASE information (.REQUEST only);

`ftce_account`: account (.REQUEST only); and,

`ftce_diags/ftce_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 `FTCEFREE` macro when no longer needed. The `FTCEFREE` macro behaves as if it was defined as:

```
void    FTCEFREE (ftce)
struct FTAMcreate *ftce;
```

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

If the flag `FTG_RDATTR` is present, then `ftg_readattr` element contains an F-READ-ATTRIB event encoded in an `FTAMreadattr` structure:

```
struct FTAMreadattr {
    int     ftra_action;

    int     ftra_attrnames;

    struct FTAMattributes ftra_attrs;
    int     ftra_ndiag;
    struct FTAMdiagnostic ftra_diags[NFDIAG];
};
```

The elements of this structure are:

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

`ftra_attrnames`: the attributes to read, containing the inclusive-or of any of the values shown in Table 2.2 on page 27 (.REQUEST only);

`ftra_attrs`: the attribute values (.RESPONSE only); and,

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