Deutsch

FUJITSU

# openFT V12.0 für Unix- und Windows-Systeme

C-Programmschnittstelle

Programmierhandbuch

Ausgabe September 2012

## Kritik… Anregungen… Korrekturen…

Die Redaktion ist interessiert an Ihren Kommentaren zu diesem Handbuch. Ihre Rückmeldungen helfen uns, die Dokumentation zu optimieren und auf Ihre Wünsche und Bedürfnisse abzustimmen.

Sie können uns Ihre Kommentare per E-Mail an manuals@ts.fujitsu.com senden.

## Zertifizierte Dokumentation nach DIN EN ISO 9001:2008

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001:2008 erfüllt.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

## Copyright und Handelsmarken

# Inhalt

# 1 Einführung in die C-Programmschnittstelle

Mit der C-Programmschnittstelle können Sie Funktionen von openFT in selbst erstellten C-Programmen nutzen:

- synchrone Dateiübertragung
- asynchrone Dateiübertragung
- asynchrone Dateiübertragungsaufträge verwalten und löschen
- Dateiattribute im fernen System ermitteln
- Dateien oder Dateiverzeichnisse im fernen System löschen
- Dateiverzeichnisse im fernen System erzeugen
- Kommandos im fernen System ausführen

Diese Funktionen, die dem openFT-Benutzer zur Verfügung stehen, können in C-Programmen verwendet werden, um Abläufe zu automatisieren. Selbstverständlich stellt die Programmschnittstelle auch Mechanismen zur Überwachung und zur Fehlerbehandlung zur Verfügung.
Außerdem besitzt die Programmschnittstelle einen Funktionsaufruf, mit dem Sie Eigenschaften der Programmschnittstelle ermitteln. Sie können Ihre Programme unempfindlich gegenüber Änderungen in späteren Versionen machen, wenn Sie diese Eigenschaften abprüfen.

Die Programmschnittstelle unterstützt unter Windows das **Multithreading**, d.h. alle Aufrufe der Programmschnittstelle sind threadsafe.

## 1.1 Änderungen gegenüber der Vorgängerversion

Die C-Programmschnittstelle zu openFT V12 bietet folgende neue Funktionen:

- Funktionsgruppe *ft_sd*()* zur Ermittlung der Attribute aller Dateien eines Dateiverzeichnisses im fernen System.
  *ft_sd*()* umfasst die Einzelfunktionen *ft_sdopen()*, *ft_sdinfo()* und *ft_sdclose()*.
- Funktionsgruppe *ft_xc*()* zum Ausführen von Kommandos im fernen System.
  *ft_xc*()* umfasst die Einzelfunktionen *ft_xcopen()*, *ft_xcinfo()* und *ft_xcclose()*.

Die Struktur *ft_prop* in der Funktion *ft_properties()* wurde erweitert:

- neues Feld *maxcmdlen*
- zusätzlicher Wert `FT_PROPV2` für das Feld *ftpropvers*

Neue Beispielprogramme:

- Ferne Kommandoausführung
- Fernes Dateiverzeichnis speicherschonend auflisten

## 1.2 Übersicht

Mit der folgenden Übersicht können Sie sich schnell orientieren, welche C-Funktionsaufrufe für welche Aufgaben zur Verfügung stehen. In Klammern sind die entsprechenden FT-Kommandos angeführt, mit denen FT-Benutzer auf Shell-Ebene arbeiten (siehe openFT-Benutzerhandbuch).

**Funktion zur Dateiübertragung**

| | |
|---|---|
| *ft_transfer* | Datei übertragen (*ft* oder *ncopy*) |

**Funktionen zur Verwaltung asynchroner Dateiübertragungsaufträge**

| | |
|---|---|
| *ft_open* | Sitzung eröffnen |
| *ft_close* | Sitzung beenden |
| *ft_reqlist* | Nicht abgeschlossene Aufträge ermitteln |
| *ft_reqstat* | Status eines Auftrags ermitteln |
| *ft_reqterm* | Auftrag abschließen |
| *ft_cancel* | Auftrag abbrechen (*ftcanr*) |

**Funktionen zum Dateimanagement**

| | |
|---|---|
| *ft_show* | Attribute einer Datei oder eines Dateiverzeichnisses im fernen System ermitteln (*ftshw*) |
| *ft_showdir* | Attribute aller Dateien eines Dateiverzeichnisses im fernen System ermitteln (*ftshw -d*) |
| *ft_delete* | Datei oder Dateiverzeichnis im fernen System löschen<br>(Datei löschen: *ftdel*; Dateiverzeichnis löschen: *ftdeldir*) |
| *ft_credir* | Dateiverzeichnis im fernen System erzeugen (*ftcredir*) |
| *ft_sd** | Funktionsgruppe zur Ermittlung der Attribute aller Dateien eines Dateiverzeichnisses im fernen System.<br>Umfasst folgende, einzelne Funktionen:<br>*ft_sdopen* Ermitteln der Attribute aller Dateien eines Dateiverzeichnisses im fernen System starten<br>*ft_sdinfo* Dateiattribute auslesen<br>*ft_sdclose* Ermitteln der Dateiattribute beenden |

### Funktion zur Abfrage von Eigenschaften der Programmschnittstelle

| | |
|---|---|
| *ft_properties* | Eigenschaften der Programmschnittstelle ermitteln |

### Funktionen zur fernen Kommandoausführung

| | | |
|---|---|---|
| *ft_xc** | Funktionsgruppe zur synchronen Ausführung eines Kommandos im fernen System.<br>Umfasst folgende, einzelne Funktionen: | |
| | *ft_xcopen* | Kommando im fernen System ausführen |
| | *ft_xcinfo* | Vom Kommando erzeugte Daten auslesen |
| | *ft_xcclose* | Kommandoausführung beenden |

## 1.3 Programmierregeln

In diesem Abschnitt ist beschrieben, was Sie unbedingt beachten müssen, wenn Sie Programme für die Programmschnittstelle von openFT erstellen.

### 1.3.1 Dateiübertragung

#### Synchrone Übertragung

Um Dateien synchron zu übertragen, benutzen Sie die Funktion *ft_transfer()*. In der Parameterliste **muss** der Parameter *synchron* den Wert `FT_SYNC` enthalten. Erst nach Beendigung der Dateiübertragung erhält das Programm wieder die Kontrolle. Ob die Dateiübertragung erfolgreich war, können Sie anhand des Rückgabewerts feststellen.

#### Asynchrone Übertragung

Um Dateien asynchron zu übertragen, sind mehrere Funktionsaufrufe notwendig. Sie ergeben sich daraus, dass bei asynchroner Dateiübertragung Aufträge erteilt, im Auftragsbuch gespeichert und eventuell erst zu einem späteren Zeitpunkt ausgeführt werden. Die Aufträge müssen verwaltet und die erfolgreiche Übertragung überwacht werden. Deshalb können Dateien nur innerhalb von „Sitzungen“ asynchron übertragen werden.

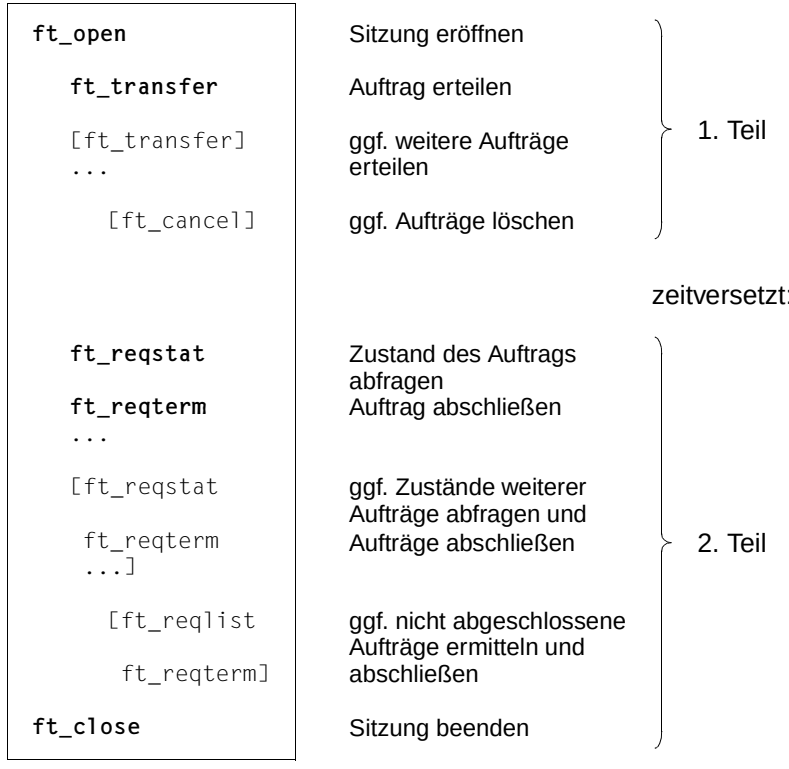
Ein Programm zur asynchronen Dateiübertragung besteht aus zwei Teilen:

– Im ersten Teil eröffnen Sie eine Sitzung. Außerdem erteilen Sie einen oder mehrere Aufträge zur Dateiübertragung. Ggf. löschen Sie Übertragungsaufträge.

  Den Auftrag selbst führt openFT zum nächstmöglichen Zeitpunkt aus.

– Im zweiten Teil erfragen Sie zeitversetzt den Status des Auftrags und schließen den Auftrag bei erfolgreicher Übertragung ab. Ggf. ermitteln Sie nicht abgeschlossene Übertragungsaufträge und schließen diese ebenfalls ab. Sie beenden die Sitzung.

Schema des Programmaufbaus:




Folgende Funktionsaufrufe sind unbedingt notwendig, um Dateien asynchron zu übertragen:

1. *ft_open()*

   Sie eröffnen eine Sitzung mit der Funktion *ft_open()*. Als Ergebnis von *ft_open()* erhalten Sie eine Sitzungsnummer (session identification), die die Sitzung eindeutig kennzeichnet. Diese Sitzungsnummer muss bei Funktionsaufrufen innerhalb derselben Sitzung als Parameter angegeben werden.

   Beim Eröffnen einer Sitzung müssen Sie ein existierendes Dateiverzeichnis als Arbeitsverzeichnis zuordnen. In diesem Arbeitsverzeichnis werden Dateien mit Verwaltungsinformationen über die bestehenden Übertragungsaufträge gespeichert.

   Sie können unterschiedlichen, **nacheinander folgenden** Sitzungen dasselbe Arbeitsverzeichnis zuordnen. Dies hat den Vorteil, dass Aufträge aus verschiedenen Sitzungen zusammen verwaltet werden.

In einem Programm können Sie mehrere Sitzungen **parallel** führen. Mit *ft_open()* können Sie allerdings mehrere Sitzungen nur dann gleichzeitig eröffnen, wenn jeder der parallelen Sitzungen ein anderes Arbeitsverzeichnis zugeordnet wird.

2. *ft_transfer()*

   Mit der Funktion *ft_transfer()* erteilen Sie einen Auftrag zur asynchronen Dateiübertragung. In der Parameterliste **muss** der Parameter *synchron* den Wert `FT_ASYNC` enthalten. In einer Sitzung können Sie nacheinander mehrere asynchrone Aufträge erteilen.

   Wenn der Auftrag erfolgreich in das Auftragsbuch eingetragen wurde, erhalten Sie als Ergebnis von *ft_transfer()* eine Request-Id. Die Request-Id kennzeichnet den Auftrag eindeutig. Sie ist so lange gültig, auch über das Sitzungsende hinaus, bis Sie den Auftrag mit der Funktion *ft_reqterm()* abschließen.

   Wenn der Auftrag im Auftragsbuch steht, müssen Sie sich nicht um die Auftragsausführung kümmern. openFT führt den asynchronen Auftrag zum frühestmöglichen Zeitpunkt aus. Wenn z.B. ein Partner im Moment nicht verfügbar ist, versucht openFT weiterhin Ihren Auftrag auszuführen. Der Auftrag bleibt so lange im Auftragsbuch, bis er erledigt ist oder bis ein evtl. angegebenes Löschdatum erreicht ist.

3. *ft_reqstat()*

   Ob die Dateiübertragung erfolgreich beendet wurde, stellen Sie mit der Funktion *ft_reqstat()* fest. Weil die asynchrone Übertragung nicht sofort erfolgt, sollten Sie den Status der Übertragung mit der Funktion *ft_reqstat()* zeitverzögert abfragen und diese Abfrage wiederholen. Wenn der Auftrag beendet ist, enthält der Parameter *status* den Wert `FT_STATT`, und wenn er abgebrochen wurde, den Wert `FT_STATA`.

4. *ft_reqterm()*

   Sie **müssen** den Auftrag mit der Funktion *ft_reqterm()* abschließen. Diese Funktion löscht die Request-Id des Auftrags und außerdem diejenige Datei, in der Informationen zum entsprechenden Dateiübertragungsauftrag gespeichert sind. Nicht mehr benötigte Ressourcen werden freigegeben.

   Die Verwaltungsdatei hat den Namen `mf.`*Request-ID* und befindet sich in dem Dateiverzeichnis, das als Parameter *workdir* mit dem Funktionsaufruf *ft_open()* bekanntgegeben wurde.

   Die Request-Ids nicht abgeschlossener Aufträge bleiben erhalten, auch wenn die Sitzungen, in denen sie erteilt wurden, bereits beendet sind. Diese Aufträge können zu einem späteren Zeitpunkt über die zugehörige Request-Id abgeschlossen werden, wenn der aktuellen Sitzung dasselbe Arbeitsverzeichnis zugeordnet ist wie der Sitzung, in der der Auftrag erteilt wurde.

5. *ft_close()*

   Mit der Funktion *ft_close()* beenden Sie die Sitzung.

### HOME-Verzeichnis

Wenn Datei- oder Verzeichnisnamen in fernen Unix- oder Windows-Systemen nicht absolut angegeben werden, dann ist das HOME-Verzeichnis des Benutzers im fernen System von Bedeutung. Relative Pfadnamen beziehen sich immer auf das HOME-Verzeichnis des jeweiligen Benutzers, sofern nicht über ein FTAC-Profil etwas anderes definiert wurde.

Für das HOME-Verzeichnis gilt:

- In Unix-Systemen ist das HOME-Verzeichnis das Verzeichnis, in dem sich der Benutzer nach dem login befindet.
- In Windows-Systemen ist das HOME-Verzeichnis eines Benutzers bei openFT-Aufträgen das Verzeichnis, das in der Benutzerverwaltung für diesen Benutzer eingetragen ist.

  Falls für den Benutzer in der Benutzerverwaltung kein Verzeichnis eingetragen ist, dann wird als HOME-Verzeichnis der Profil-Pfad des Benutzers verwendet. Der Profil-Pfad ist *\Users\Benutzer*, wobei *Benutzer* nicht identisch mit dem Namen des Benutzers sein muss.

  Falls auch der Profil-Pfad des Benutzers nicht ermittelt werden kann, wird das Home-Verzeichnis von openFT im Verzeichnis *\Users* angelegt und die Zugriffsrechte werden vollständig für SYSTEM, Administratoren und den jeweiligen Benutzer erlaubt.
  Der Name des durch openFT angelegten Home-Verzeichnisses wird dabei folgendermaßen bestimmt:
  – Lokale Benutzerkennung: *Benutzerkennung.Computername*
  – Globale Benutzerkennung (Domäne\Benutzerkennung): *Benutzerkennung.Domäne*

### Asynchrone Aufträge verwalten

Zur Verwaltung asynchroner Aufträge gibt es noch weitere Funktionen:

– *ft_cancel()*

Mit der Funktion *ft_cancel()* brechen Sie asynchrone Aufträge ab, die bereits bearbeitet werden oder die noch im Auftragsbuch auf ihre Bearbeitung warten.

Der Einsatz dieser Funktion ist z.B. sinnvoll, wenn das zu erstellende Programm eine Benutzeroberfläche hat und dem Benutzer Möglichkeiten gibt, einzugreifen. Man kann sich z.B. ein Programm vorstellen, das dem Benutzer wartende Übertragungsaufträge (*ft_reqstat* (status=FT_STATW)) anzeigt und das es ihm erlaubt, diese Aufträge abzubrechen.

Ein anderer Einsatzfall ist, wenn Übertragungsaufträge irrtümlich erteilt wurden und nun gelöscht werden sollen.

Mit der Funktion *ft_cancel()* können Sie nur Aufträge abbrechen, die im Auftragsbuch stehen und eine Request-Id haben. Wenn der Auftrag in einer anderen Sitzung erteilt wurde, muss der aktuellen Sitzung dasselbe Arbeitsverzeichnis zugeordnet sein, wie der Sitzung, in der der Auftrag erteilt wurde. Wenn die Funktion *ft_transfer()* den Rückgabewert 0 liefert, konnte der Auftrag nicht ins Auftragsbuch eingetragen werden. Diese erfolglosen Versuche, Aufträge einzutragen, beenden sich selbst mit einer Fehlermeldung.

Aufträge, die Sie mit *ft_cancel()* abbrechen, **müssen** Sie mit *ft_reqterm()* abschließen, damit die zugehörige Request-Id gelöscht wird.

– *ft_reqlist()*

Alle Übertragungsaufträge müssen abgeschlossen werden, damit zugehörige Request-Ids und Verwaltungsdateien gelöscht und nicht benötigte Ressourcen freigegeben werden können.

Mit der Funktion *ft_reqlist()* ermitteln Sie die nicht abgeschlossenen Aufträge aus allen Sitzungen, denen dasselbe Arbeitsverzeichnis zugeordnet wurde wie der aktuellen Sitzung. Beachten Sie, dass dabei nicht alle nicht-abgeschlossenen Aufträge ermittelt werden, sondern nur die aus Sitzungen mit demselben Arbeitsverzeichnis.

### 1.3.2 Dateimanagement

Um Dateiattribute im fernen System zu ermitteln, stehen Ihnen folgende Funktionen zur Verfügung:

- Mit der Funktion *ft_show()* lassen Sie sich die Attribute **einer** Datei oder eines Dateiverzeichnisses auflisten.
- Mit der Funktion *ft_showdir()* lassen Sie sich die Attribute **mehrerer** Dateien eines Dateiverzeichnisses auflisten. Die Anzahl der Dateien muss vor dem Aufruf festgelegt werden.
- Die Funktionsgruppe *ft_sd*()* ermittelt die Attribute **aller** Dateien eines Dateiverzeichnisses im fernen System. Im Gegensatz zu *ft_showdir()* muss die Anzahl der Dateien vor dem Aufruf nicht bekannt sein. Die Funktionsgruppe umfasst folgende, einzelne Funktionen:
  - Mit der Funktion *ft_sdopen()* wird das Ermitteln der Attribute aller Dateien eines Dateiverzeichnisses im fernen System initiiert.
  - Mit der Funktion *ft_sdinfo()* werden die Dateiattribute ausgelesen.
  - Mit der Funktion *ft_sdclose()* wird das Ermitteln der Dateiattribute beendet.
- Mit der Funktion *ft_delete() löschen Sie* eine Datei oder ein Dateiverzeichnis im fernen System.

### 1.3.3 Ferne Kommandoausführung

Ein Kommando wird synchron im fernen System ausgeführt. Die vom ausgeführten Kommando auf *stdout* und *stderr* ausgegebenen Daten können getrennt abgerufen werden.

Dazu steht Ihnen die Funktionsgruppe *ft_xc*()* zur Verfügung. Sie umfasst folgende, einzelne Funktionen:

ft_xcopen()
: Mit der Funktion *ft_xcopen()* wird das Kommando im fernen System synchron ausgeführt.

ft_xcinfo()
: Mit der Funktion *ft_xcinfo()* werden die vom Kommando erzeugten Daten ausgelesen.

ft_xcclose()
: Mit der Funktion *ft_xcclose()* wird die Kommandoausführung beendet.

### 1.3.4 Angaben zum fernen System

Alle Funktionen, die auf ein fernes System zugreifen, müssen das ferne System und die Berechtigung für den Zugang zum fernen System bekanntgeben. Dazu dient die Struktur *ft_admission*.

In Windows ist die Struktur *ft_admission* in der Header-Datei *...\openFT\include\ftapi.h* definiert.

In Unix-Systemen ist die Struktur *ft_admission* in der Header-Datei */usr/include/ftapi.h* definiert.

#### ft_admission

Die Struktur *ft_admission* ist folgendermaßen aufgebaut:

```
struct ft_admission
{
   char  *remsys;         /* Eingabe */
   char  *remadmis;       /* Eingabe */
   char  *remaccount;     /* Eingabe */
   char  *rempasswd;      /* Eingabe */
};
```

Die Felder der Struktur *ft_admission* haben folgende Bedeutung.

remsys
: Name des Partnersystems in der Partnerliste oder Adresse des Partnersystems.

  Die Adresse des Partnersystems wird in folgender Form angegeben:

  [protocol://]host[:[port].[tsel].[ssel].[psel]]

  protocol
  : Protokollstack, über den der Partner angesprochen wird.
    Mögliche Werte:

    **openft** (openFT-Protokoll), Standardwert

    **ftam** (FTAM-Protokoll)

    **ftp** (ftp-Protokoll)

  host
  : Internet-Hostname, IP-Adresse oder GLOBALER NAME aus dem TNS, Pflichtparameter. Format der IP-Adressen (Beispiel):
    `%ip111.222.123.234` (IPv4) bzw.
    `%ip6[FEDC:BA98:7654:3210:FEDC:BA98:7654:3210]` (IPv6)

    Die eckigen Klammern [..] müssen bei IPv6 angegeben werden.

port Portnummer bei TCP/IP-Kopplung, optional.

tsel Transport-Selektor (nur openFT- und FTAM-Protokoll), optional.

ssel Session-Selektor bei FTAM-Kopplung, optional.

psel Presentation-Selektor bei FTAM-Kopplung, optional.

Weitere Einzelheiten zur Adressierung von Partnersystemen finden Sie in der Online-Hilfe oder im openFT-Benutzerhandbuch.

remadmis
: Entweder Benutzerkennung oder eine FTAC-Zugangsberechtigung im fernen System.

remaccount
: Abrechnungsnummer im fernen System

rempasswd
: Kennwort im fernen System

Je nach openFT-Partnersystem muss für *remadmis*, *remaccount* und *rempasswd* folgendes angegeben werden:

Bei BS2000:
: *remadmis*, wenn für das ferne System eine FTAC-Zugangsberechtigung vorhanden ist, sonst: *remadmis*, *remaccount* und, falls ein Kennwort vergeben ist, *rempasswd*.

Bei Unix-Systemen:
: *remadmis*, wenn für das ferne System eine FTAC-Zugangsberechtigung vorhanden ist, sonst: *remadmis* und, falls ein Kennwort vergeben ist, *rempasswd*.

Bei Windows-Systemen:
: *remadmis*, wenn für das ferne System eine FTAC-Zugangsberechtigung vorhanden ist, sonst: *remadmis*, falls eine Benutzerkennung vergeben ist, und *rempasswd*, falls ein Kennwort vergeben ist.

Bei OS/390 und z/OS
: *remadmis*, *remaccount* und, falls ein Kennwort vergeben ist, *rempasswd*.

Bei FTAM-Partnersystemen, bei denen kein Produkt der openFT-Produktfamilie im Einsatz ist:
: *remadmis*, und falls eine Abrechnungnummer vergeben ist, *remaccount* und falls ein Kennwort vorgegeben ist *rempasswd*.

Bei anderen Partnersystemen:
: entsprechend den Konventionen des jeweiligen Partnersystems

Alle nicht anzugebenden Felder müssen den Wert NULL enthalten.

### 1.3.5 Fehlerbehandlung

Alle Funktionsaufrufe enden mit einer Rückmeldung. Der Rückgabewert zeigt den erfolgreichen Abschluss an oder informiert pauschal über einen aufgetretenen Fehler. Detailliertere Informationen erhalten Sie, wenn Sie eine Funktion mit dem optionalen Parameter *errorinfo* aufrufen. Sofort nach Auftreten eines Fehlers enthält die Struktur *ft_err* Fehlermeldungen, mit denen Sie entsprechende Fehlerbehandlungen programmieren können.

In Windows ist die Struktur *ft_err* in der Header-Datei *...\openFT\include\ftapi.h* definiert.

In Unix-Systemen ist die Struktur *ft_err* in der Header-Datei */usr/include/ftapi.h* definiert.

**ft_err**

Die Struktur *ft_err* ist folgendermaßen aufgebaut:

```
struct ft_err
{
   long  main;           /* Ausgabe */
   long  detail;         /* Ausgabe */
   long  additional;     /* Ausgabe */
};
```

Die Felder der Struktur *ft_err* haben folgende Bedeutung:

main
: enthält die Fehlerklasse, z.B. Parameterfehler, interne Fehler

detail
: beschreibt den Fehler, z.B. ungültiger Parameterwert

additional
: enthält zusätzliche Fehlerinformation, z.B. welcher Parameter fehlerhaft ist

Die Fehlercodes sind im Kapitel „Fehlercodes“ auf Seite 79 beschrieben.

### 1.3.6 Version der Programmschnittstelle

Mit dem Funktionsaufruf *ft_properties()* ermitteln Sie die Version der openFT-Programmschnittstelle sowie wichtige versionsspezifische Systemwerte. Mit dieser Funktion sichern Sie - auch ohne erneutes Compilieren - die Ablauffähigkeit Ihrer Programme mit zukünftigen Versionen von openFT. Dieser Funktionsaufruf hat vor allem Bedeutung, wenn Sie Programme einsetzen, die mit verschiedenen Versionen der Programmschnittstelle ablaufen sollen.

#### ft_options

Die in der Version 2 der openFT-Programmschnittstelle eingeführte Funktion *ft_credir()* und die erweiterten Datenstrukturen können nur verwendet werden, wenn der Parameter *options* bei den jeweilien Funktionen angegeben wird.

Die in Version 3 der openFT-Programmschnittstelle eingeführten Funktionen *ft_sdopen()* und *ft_xcopen()* können nur verwendet werden, wenn der Parameter *options* bei den jeweiligen Funktionen angegeben wird.

Die Struktur *ft_options* ist folgendermaßen aufgebaut:

```
struct ft_options
{
   int ftoptsvers;   /* Eingabe */
   int ftapivers;    /* Eingabe */
};
```

Die Felder der Struktur haben folgende Bedeutung:

ftoptsvers
: Version der Datenstruktur.
*ftoptsvers* muss mit dem Wert `FT_OPTSV1` versorgt werden.

ftapivers
: gibt die Version der Programmschnittstelle an:

  `FT_APIV2`
  : Die im Parameter *additional* angegebene openFT-Meldungsnummer (*ft_err.detail*=`FTED_FTMSG`) folgt dem neuen Meldungsnummern-Schema, das in openFT V10 eingeführt wurde.

  `FT_APIV3`
  : Wird für die Nutzung der Funktionen *ft_sdopen()* und *ft_xcopen()* benötigt. Die im Parameter *additional* angegebene openFT-Meldungsnummer (*ft_err.detail*=`FTED_FTMSG`) folgt dem neuen Meldungsnummern-Schema, das in openFT V10 eingeführt wurde.

# 2 Programme erstellen und einsetzen

**Include-Datei**

Alle C-Programme, die die Programmschnittstelle von openFT nutzen, müssen folgende Zeile enthalten:

– in Windows: `#include <ftapi.h>`
– in Unix-Systemen: `#include "ftapi.h"`

In diesem Include sind die Datentypen und Funktionsprototypen definiert.

In Windows ist diese Include-Datei im Unterverzeichnis *openFT\include* des openFT-Installationsverzeichnisses zu finden.

## 2.1 Übersetzen und Binden in Windows-Systemen

Ein Programm, das die Programmschnittstelle von openFT nutzt, muss mit der Import-Bibliothek *ftapi.lib* gebunden werden. Diese Bibliothek steht im Verzeichnis *openFT\lib* des openFT-Installationsverzeichnisses.

Zur Laufzeit wird zusätzlich die Bibliothek *ftapi.dll* aus dem Verzeichnis *..\openFT\bin* nachgeladen.

*ftapi.lib* und *ftapi.dll* wurden mit Microsoft Visual Studio 2010 erstellt.

**64-Bit Unterstützung in Windows-Systemen**

Zusätzlich wird eine 64 Bit-DLL mit dem Namen *ftapi64.dll* für die Windows x64-Systeme und für Windows Itanium-Systeme zur Verfügung gestellt.

Bei der Installation von openFT wird die zum entsprechenden Betriebssystem gehörende Variante der *ftapi64.dll* automatisch in das Verzeichnis *..\openFT\bin* installiert.

Die zugehörige Import-Bibliothek *ftapi64.lib* ist für Windows x64-Systeme im Verzeichnis *..\openFT\lib\x64* und für Windows Itanium-Systeme im Verzeichnis *..\openFT\lib\ia64* zu finden.

## 2.2 Übersetzen und Binden in Unix-Systemen

In ein Programm, das die Programmschnittstelle von openFT nutzt, müssen die openFT-Funktionen aus der openFT-Bibliothek eingebunden werden. Rufen Sie den C-Compiler mit der Option *-lftapi* auf.

Zusätzlich müssen auf einigen Anlagen noch folgende Schalter angegeben werden:

| | |
|---|---|
| AIX | -WI,-brtl |
| HP (Itanium 64bit) | +DD64 |
| Solaris (SPARC)[1] | -xarch=v9 |

[1] Auf der Plattform Solais (SPARC) wird zusätzlich eine 64bit-Bibliothek ausgeliefert.

Unter HP-UX muss der C-Compiler außerdem unbedingt im ANSI-Modus aufgerufen werden.

## 2.3 Hinweis für den Programmeinsatz

Informationen über asynchrone Dateiübertragungsaufträge werden in Dateien gespeichert mit dem Namen `mf`.*Request-ID* in dem Dateiverzeichnis, das als Parameter *workdir* mit dem Funktionsaufruf *ft_open()* bekanntgegeben wurde. Diese Dateien werden gelöscht, wenn Sie mit dem Funktionsaufruf *ft_reqterm()* Aufträge beenden.

# 3 Beschreibung der C-Funktionen

**Darstellungsmittel**

Bei der Darstellung der Funktionen wird folgende Auszeichnung verwendet:

`dicktengleiche Schrift`
: für Shell-Kommandos, Funktionsaufrufe, Programme und Programmteile sowie für konstante Werte im Fließtext.

*kursive Schrift*
: für Funktionsnamen und Parameter

In der Syntaxdarstellung kennzeichnet der Kommentar „Eingabe“ Eingabeparameter und der Kommentar „Ausgabe“ Ausgabeparameter. Diese Kommentare stehen nicht bei Strukturen, sondern jeweils bei den Parametern der untersten Ebene.

## 3.1 ft_cancel - Asynchronen Auftrag abbrechen

*ft_cancel()* löscht asynchrone Aufträge, die gerade bearbeitet werden oder die noch auf die Bearbeitung warten.

### Syntax

```
#include <ftapi.h>

int ft_cancel(const void *session,          /* Eingabe */
              long rid,                     /* Eingabe */
              struct ft_err *errorinfo,
              void *options);               /* Eingabe */
```

### Parameter

session
: Sitzungsnummer der Sitzung, in der der Auftrag abgebrochen werden soll.

rid
: Request-Id des Auftrags, der abgebrochen werden soll.

  Wenn der abzubrechende Auftrag in einer anderen Sitzung erteilt wurde, muss der aktuellen Sitzung dasselbe Arbeitsverzeichnis zugeordnet sein wie der Sitzung, in der der Auftrag erteilt wurde.
  Außerdem muss das Programm, in dem der asynchrone Auftrag abgebrochen wird, unter derselben Kennung laufen wie das, in dem der Auftrag erteilt wurde.

errorinfo
: Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17).
  Die Angabe des Parameters ist optional. Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert NULL angeben.

options
: Die Angabe des Parameters *options* ist optional. Wird der Wert NULL angegeben, dann ist das Meldungsverhalten der Programmschnittstelle kompatibel zur Programmschnittstelle von openFT < V10.
  Alternativ können durch die Angabe der Struktur *ft_options* (siehe Abschnitt „ft_options“ auf Seite 18) das openFT-Meldungsnummernschema ab openFT V10 und die Funktionserweiterungen aktiviert werden.

### Rückgabewert

0 Kein Fehler

-1 Fehler. Die Fehlerart wird in *errorinfo* hinterlegt.

## 3.2 ft_close - Sitzung beenden

*ft_close()* beendet eine Sitzung, die mit *ft_open()* eröffnet wurde. Diese Funktion muss innerhalb einer Sitzung als letztes aufgerufen werden. *ft_close()* gibt nicht mehr benötigte Ressourcen frei. Die Sitzungsnummer wird gelöscht, anschließend können Sie sich nicht mehr auf diese Sitzung beziehen.

### Syntax

```
#include <ftapi.h>

int ft_close(const void *session,          /* Eingabe */
             struct ft_err *errorinfo,
             void *options);               /* Eingabe */
```

### Parameter

session
: Sitzungsnummer der Sitzung, die beendet werden soll.

errorinfo
: Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17).
Die Angabe des Parameters ist optional. Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert NULL angeben.

options
: Die Angabe des Parameters *options* ist optional. Wird der Wert NULL angegeben, dann ist das Meldungsverhalten der Programmschnittstelle kompatibel zur Programmschnittstelle von openFT < V10.
Alternativ können durch die Angabe der Struktur *ft_options* (siehe Abschnitt „ft_options“ auf Seite 18) das openFT-Meldungsnummernschema ab openFT V10 und die Funktionserweiterungen aktiviert werden.

### Rückgabewert

0 Kein Fehler

-1 Fehler. Die Fehlerart wird in *errorinfo* hinterlegt.

## 3.3 ft_credir - Dateiverzeichnis im fernen System erstellen

*ft_credir()* erstellt ein Dateiverzeichnis im fernen System.

Dateiverzeichnisnamen dürfen die in der Struktur *ft_prop* im Feld *maxrfnsize* angegebene Länge nicht überschreiten, siehe Abschnitt „ft_properties - Eigenschaften der Programmschnittstelle ermitteln“ auf Seite 31.

### Syntax

```
#include <ftapi.h>

int ft_credir(const struct ft_admission *admis,  /* Eingabe */
              const struct ft_crepar *par,       /* Eingabe */
              struct ft_err *errorinfo,
              void *options);                    /* Eingabe */
```

### Parameter

admis

Angaben für das ferne System (siehe Abschnitt „ft_admission“ auf Seite 15).

par Angaben für den Auftrag, die Sie mit der Struktur *ft_crepar* bekanntgeben:

```
struct ft_crepar
{
   int   creparvers;            /* Eingabe */
   char  *dn;                   /* Eingabe */
   char  *mgmtpasswd;           /* Eingabe */
   char  *fud;                  /* Eingabe */
   int   fudlen;                /* Eingabe */
};
```

Die Felder der Struktur *ft_crepar* haben folgende Bedeutung:

creparvers

Version der Datenstruktur.
*creparvers* muss mit dem Wert `FT_CPARV1` versorgt werden.

dn Name des Dateiverzeichnisses im fernen System, das erstellt werden soll.

Absolute und relative Pfadangaben sind erlaubt. Relative Pfadangaben beziehen sich auf die im Berechtigungsprofil festgelegte Benutzerkennung, wenn die FTAC-Funktion eingesetzt wird, sonst auf das HOME-Verzeichnis.

mgmtpasswd
Kennwort des Dateiverzeichnisses, falls es mit einem Kennwort geschützt ist.

fud Adresse eines Datenbereichs für die sogenannten „Further Details“, die im Fehlerfall eine genauere Fehlerursache angeben können.
Bei Angabe von NULL wird keine weiterführende Fehlerursache ausgegeben. Der Parameter *fud* steht nur dann zur Verfügung, wenn *creparvers* auf den Wert `FT_CPARV1` gesetzt wird und beim Aufruf von *ft_credir* der Parameter *options* angegeben ist.

fudlen
Länge des Datenbereichs von *fud*.
Der Parameter *fudlen* steht nur dann zur Verfügung, wenn *creparvers* auf den Wert `FT_CPARV1` gesetzt wird und beim Aufruf von *ft_credir* der Parameter *options* angegeben ist.

errorinfo
Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17).
Die Angabe des Parameters ist optional. Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert NULL angeben.

options
Die Angabe des Parameters *options* ist obligatorisch. Der Aufbau der Struktur *ft_options* ist im Abschnitt „Version der Programmschnittstelle“ auf Seite 18 beschrieben.

**Rückgabewert**

0 Kein Fehler. Das Dateiverzeichnis wurde erstellt.

-1 Fehler. Das Dateiverzeichnis wurde nicht erstellt.
Die Fehlerart wird in *errorinfo* hinterlegt.

## 3.4 ft_delete - Datei oder Dateiverzeichnis im fernen System löschen

*ft_delete()* löscht eine Datei oder ein Dateiverzeichnis im fernen System. Dateiverzeichnisse, die gelöscht werden sollen, müssen leer sein.

Um eine **Datei** zu löschen, muss der Parameter *filetype* in der Struktur *par* den Wert `FT_FILE` enthalten.

Um ein **Dateiverzeichnis** zu löschen, muss der Parameter *filetype* in der Struktur *par* den Wert `FT_DIRECTORY` enthalten.

Dateinamen bzw. Dateiverzeichnisnamen dürfen die in der Struktur *ft_prop* im Feld *maxrfnsize* angegebene Länge nicht überschreiten (siehe Abschnitt „ft_properties - Eigenschaften der Programmschnittstelle ermitteln“ auf Seite 31).

### Syntax

```
#include <ftapi.h>

int ft_delete(const struct ft_admission *admis,  /* Eingabe */
              const struct ft_delpar *par,       /* Eingabe */
              struct ft_err *errorinfo,
              void *options);                    /* Eingabe */
```

### Parameter

admis
: Angaben für das ferne System (siehe Abschnitt „ft_admission“ auf Seite 15).

par
: Angaben für den Löschauftrag, die Sie mit der Struktur *ft_delpar* bekanntgeben:

```
struct ft_delpar
{
   int   delparvers;             /* Eingabe */
   char  *fn;                    /* Eingabe */
   char  *mgmtpasswd;            /* Eingabe */
   enum  ft_filedir filetype;    /* Eingabe */
   char  *fud;                   /* Eingabe */
   int   fudlen;                 /* Eingabe */
};
```

Die Felder der Struktur *ft_delpar* haben folgende Bedeutung:

delparvers
: Version der Datenstruktur.
*delparvers* muss mit dem Wert `FT_DPARV1` oder `FT_DPARV2` versorgt werden.

fn
: Name der Datei oder des Dateiverzeichnisses im fernen System, die/das gelöscht werden soll.

  Absolute und relative Pfadangaben sind erlaubt. Relative Pfadangaben beziehen sich auf die im Berechtigungsprofil festgelegte Benutzerkennung, wenn die FTAC-Funktion eingesetzt wird, sonst auf das HOME-Verzeichnis, siehe Seite 12.

mgmtpasswd
: Kennwort der Datei/des Dateiverzeichnisses, falls sie/es mit einem Kennwort geschützt ist.

filetype
: gibt an, was gelöscht werden soll:

  `FT_FILE`
  : Datei (Defaultwert nach Initialisierung der Parameterliste *ft_delpar* mit binär 0)

  `FT_DIRECTORY`
  : Dateiverzeichnis (nicht für FTAM-Partner)

fud
: Adresse eines Datenbereichs für die sogenannten „Further Details“, die im Fehlerfall eine genauere Fehlerursache angeben können.
Bei Angabe von NULL wird keine weiterführende Fehlerursache ausgegeben. Der Parameter *fud* steht nur dann zur Verfügung, wenn *delparvers* auf den Wert `FT_DPARV2` gesetzt wird und beim Aufruf von *ft_delete* der Parameter *options* angegeben ist.

fudlen
: Länge des Datenbereichs von *fud*.
Der Parameter *fudlen* steht nur dann zur Verfügung, wenn *delparvers* auf den Wert `FT_DPARV2` gesetzt wird und beim Aufruf von *ft_delete* der Parameter *options* angegeben ist.

errorinfo

Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17).
Die Angabe des Parameters ist optional. Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert NULL angeben.

options

Die Angabe des Parameters *options* ist optional. Wird der Wert NULL angegeben, dann ist das Meldungsverhalten der Programmschnittstelle kompatibel zur Programmschnittstelle von openFT < V10.
Alternativ können durch die Angabe der Struktur *ft_options* (siehe Abschnitt „ft_options“ auf Seite 18) das openFT-Meldungsnummernschema ab openFT V10 und die Funktionserweiterungen aktiviert werden.

**Rückgabewert**

0 Kein Fehler. Die Datei oder das Dateiverzeichnis wurde gelöscht.

-1 Fehler. Die Datei oder das Dateiverzeichnis wurde nicht gelöscht.
Die Fehlerart wird in *errorinfo* hinterlegt.

## 3.5 ft_open - Sitzung eröffnen

*ft_open()* eröffnet eine Sitzung. Nur innerhalb einer Sitzung können Sie Dateien asynchron übertragen (Funktion *ft_transfer()*) und asynchrone Dateiübertragungsaufträge verwalten (Funktionen *ft_reqlist()*, *ft_reqstat()*, *ft_cancel()* und *ft_reqterm()*).

*ft_open()* liefert als Resultat eine Sitzungsnummer, die die Sitzung eindeutig kennzeichnet. Diese Sitzungsnummer muss bei Funktionsaufrufen innerhalb derselben Sitzung als Parameter angegeben werden.

In einem Programm können Sie mehrere Sitzungen gleichzeitig eröffnen, wenn die zugeordneten Arbeitsverzeichnisse unterschiedlich sind.

### Syntax

```
#include <ftapi.h>

void *ft_open(const char *workdir,          /* Eingabe */
              struct ft_err *errorinfo,
              void *options);               /* Eingabe */
```

### Parameter

workdir
: Name des Arbeitsverzeichnisses, das der Sitzung zugeordnet wird.

  In diesem Arbeitsverzeichnis werden Dateien mit Verwaltungsinformationen abgelegt.

  Beachten Sie, dass die Kennung, unter der die Programmschnittstelle aufgerufen wird, das Recht haben muss, Dateien in diesem Dateiverzeichnis anzulegen.

errorinfo
: Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17).
  Die Angabe des Parameters ist optional. Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert NULL angeben.

options

Die Angabe des Parameters *options* ist optional. Wird der Wert NULL angegeben, dann ist das Meldungsverhalten der Programmschnittstelle kompatibel zur Programmschnittstelle von openFT < V10.
Alternativ können durch die Angabe der Struktur *ft_options* (siehe Abschnitt „ft_options“ auf Seite 18) das openFT-Meldungsnummernschema ab openFT V10 und die Funktionserweiterungen aktiviert werden.

**Rückgabewert**

*n* Sitzungs-Id ($n \neq 0$).
Dieser Wert muss bei Funktionsaufrufen innerhalb derselben Sitzung angegeben werden.

NULL Fehler. Die Fehlerart wird in *errorinfo* hinterlegt.

## 3.6 ft_properties - Eigenschaften der Programmschnittstelle ermitteln

*ft_properties()* ermittelt die Version der Programmschnittstelle von openFT und versionsspezifische Systemwerte. Mit diesen von der Funktion *ft_properties()* gelieferten Werten überprüfen Sie, ob Ihr Programm mit der gleichen oder einer anderen Version der Programmschnittstelle erstellt wurde.

### Syntax

```
#include <ftapi.h>

int ft_properties(struct ft_prop *prop,
                  struct ft_err *errorinfo);
```

### Parameter

prop Bereich, in dem die Version der verwendeten openFT-Programmschnittstelle sowie die gültigen Systemwerte hinterlegt sind. Dazu dient die Struktur *ft_prop*:

```
struct ft_prop
{
   int   ftpropvers;        /* Eingabe */
   int   ftvers;            /* Ausgabe */
   long  optfunct;          /* Ausgabe */
   int   maxlfnsize;        /* Ausgabe */
   int   maxrfnsize;        /* Ausgabe */
   int   maxsyssize;        /* Ausgabe */
   int   maxadmissize;      /* Ausgabe */
   int   maxaccsize;        /* Ausgabe */
   int   maxpwdsize;        /* Ausgabe */
   int   maxfpwdsize;       /* Ausgabe */
   int   maxrecord;         /* Ausgabe */
   int   maxacntsize;       /* Ausgabe */
   int   maxlegalqsize;     /* Ausgabe */
   int   maxcpwdsize;       /* Ausgabe */
   int   maxlprocsize;      /* Ausgabe */
   int   maxrprocsize;      /* Ausgabe */
   int   maxcmdlen;         /* Ausgabe */
};
```

Die Felder der Struktur *ft_prop* haben folgende Bedeutung:

ftpropvers
: Version der Datenstruktur.
*ftpropvers* muss mit dem Wert `FT_PROPV1` oder `FT_PROPV2` versorgt werden.

ftvers
: Version von openFT
(z.B. für Version 8.1: `810`, für Version 10.0: `1000`, für Version 12.0: `1200`)

optfunct
: (Für späteren Gebrauch reserviert.)

maxlfnsize
: maximale Länge für den lokalen Dateinamen

maxrfnsize
: maximale Länge für den Dateinamen im fernen System

maxsyssize
: maximale Länge für den Namen des fernen Systems

maxadmissize
: maximale Länge für die Benutzerkennung bzw. die Zugangsberechtigung im fernen System

maxaccsize
: maximale Länge für die Abrechnungsnummer im fernen System

maxpwdsize
: maximale Länge für das Kennwort im fernen System

maxfpwdsize
: maximale Länge für das Dateikennwort im fernen System

maxrecord
: maximale Satzlänge

maxacntsize
: maximale Länge für das Abrechnungskonto beim FTAM-Partner

maxlegalqsize
: maximale Länge der rechtlichen Bestimmung (Copyright)

maxcpwdsize
: maximale Länge für das Kennwort zum Erzeugen einer Datei im fernen System

maxlprocsize
: maximale Gesamtlänge der lokalen Folgeverarbeitungen

maxrprocsize
: maximale Gesamtlänge der fernen Folgeverarbeitungen

maxcmdlen
: maximale Länge des im fernen System mit *ft_xcopen()* auszuführenden Kommandos.
Der Parameter *maxcmdlen* steht nur dann zur Verfügung, wenn *ftpropvers* auf den Wert `FT_PROPV2` gesetzt wird.

errorinfo
: Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17).
Die Angabe des Parameters ist optional. Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert NULL angeben.

**Rückgabewert**

| | |
|---|---|
| 0 | Kein Fehler |
| -1 | Fehler. Die Fehlerart wird in *errorinfo* hinterlegt. |

## 3.7 ft_reqlist - Nicht abgeschlossene Aufträge ermitteln

*ft_reqlist()* ermittelt die Request-Ids der Aufträge für asynchrone Dateiübertragung, die noch nicht mit der Funktion *ft_reqterm()* abgeschlossen sind.
Wenn der Parameter *list* den Wert NULL oder wenn der Parameter *listlen* den Wert 0 hat, erhalten Sie nur die Anzahl der nicht mit *ft_reqterm()* abgeschlossenen Aufträge.

Es werden die Aufträge aus allen Sitzungen erfasst, denen mit der Funktion *ft_open* dasselbe Arbeitsverzeichnis zugeordnet wurde wie der aktuellen Sitzung.

### Syntax

```
#include <ftapi.h>

int ft_reqlist(const void *session,          /* Eingabe */
               long *list,
               int listlen,                  /* Eingabe */
               struct ft_err *errorinfo,
               void *options);               /* Eingabe */
```

### Parameter

session
: Sitzungs-Id der Sitzung, für die die nicht abgeschlossenen asynchronen Dateiübertragungsaufträge ermittelt werden sollen.

list
: Bereich, in dem die Request-Ids der nicht abgeschlossenen asynchronen Dateiübertragungsaufträge gespeichert werden. Die Länge dieses Bereichs (Anzahl der Einträge) muss in *listlen* angegeben werden.

  Wenn *list* NULL ist, wird nur die Anzahl (und nicht die Request-Ids) der noch nicht abgeschlossenen Aufträge ermittelt.

listlen
: Anzahl der Einträge in *list*.

  Wenn *listlen* 0 ist, wird nur die Anzahl (und nicht die Request-Ids) der noch nicht abgeschlossenen Aufträge ermittelt.

errorinfo

Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17).
Die Angabe des Parameters ist optional. Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert NULL angeben.

options

Die Angabe des Parameters *options* ist optional. Wird der Wert NULL angegeben, dann ist das Meldungsverhalten der Programmschnittstelle kompatibel zur Programmschnittstelle von openFT < V10.
Alternativ können durch die Angabe der Struktur *ft_options* (siehe Abschnitt „ft_options“ auf Seite 18) das openFT-Meldungsnummernschema ab openFT V10 und die Funktionserweiterungen aktiviert werden.

**Rückgabewert**

*n* Anzahl der gefundenen Einträge ($n \geq 0$).
Wenn *n* größer als *listlen* ist, werden die ersten *listlen* Einträge in *list* hinterlegt.

-1 Fehler. Die Fehlerart wird in *errorinfo* hinterlegt.

## 3.8 ft_reqstat - Status eines Auftrags ermitteln

*ft_reqstat()* ermittelt den Status eines asynchronen Dateiübertragungsauftrags.

### Syntax

```
#include <ftapi.h>

int ft_reqstat(const void *session,          /* Eingabe */
               long rid,                     /* Eingabe */
               struct ft_status *stat,
               struct ft_err *errorinfo,
               void *options);               /* Eingabe */
```

### Parameter

session
: Sitzungs-Id der Sitzung, in der der Status des Übertragungsauftrags ermittelt werden soll.

rid
: Request-Id des Auftrags, dessen Status ermittelt werden soll.

  Wenn der Auftrag in einer anderen Sitzung erteilt wurde, muss der aktuellen Sitzung dasselbe Arbeitsverzeichnis zugeordnet sein wie der Sitzung, in der der Auftrag erteilt wurde.

stat
: Bereich, in den die Statusinformation geschrieben wird. Dazu dient die Struktur *ft_status*:

```
#define STAT_FUD_LEN    65
#define STAT_FN_LEN    128

struct ft_status
{
   int   ftstatvers;          /* Eingabe */
   enum  ft_stat status;      /* Ausgabe */
   char  fn[STAT_FN_LEN];     /* Ausgabe */
   long  tid;                 /* Ausgabe */
   int   msg;                 /* Ausgabe */
   char  fud[STAT_FUD_LEN];   /* Ausgabe */
};
```

  ftstatvers
  : Version der Datenstruktur.
    *ftstatvers* muss mit dem Wert `FT_STATV1` oder `FT_STATV2` versorgt werden.

status

Status des Auftrags:

`FT_STATW`
: Der Auftrag wartet auf Ausführung.

`FT_STATR`
: Der Auftrag wird ausgeführt.

`FT_STATA`
: Der Auftrag wurde abgebrochen.

`FT_STATT`
: Der Auftrag ist beendet.

fn
: lokaler mit '\0' terminierter Dateiname. Wenn der Dateiname länger als 127 Zeichen ist, wird er gekürzt.

tid
: Transfer-Id

msg
: Meldungsnummer bei abgebrochenen oder beendeten Aufträgen (siehe Online-Hilfe).

  Mit Hilfe des Feldes *ft_apivers* in der Struktur *ft_options* kann das zu verwendende Meldungsnummern-Schema festgelegt werden.

fud
: mit '\0' terminierte „Further Details“, die im Fehlerfall eine genauere Fehlerursache angeben können.
  Der Parameter *fud* steht nur dann zur Verfügung, wenn *ftstatvers* auf den Wert `FT_STATV2` gesetzt wird und beim Aufruf von *ft_reqstat* der Parameter *options* angegeben ist.

errorinfo

Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17).
Die Angabe des Parameters ist optional. Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert NULL angeben.

options

Die Angabe des Parameters *options* ist optional. Wird der Wert NULL angegeben, dann ist das Meldungsverhalten der Programmschnittstelle kompatibel zur Programmschnittstelle von openFT < V10.
Alternativ können durch die Angabe der Struktur *ft_options* (siehe Abschnitt „ft_options“ auf Seite 18) das openFT-Meldungsnummernschema ab openFT V10 und die Funktionserweiterungen aktiviert werden.

**Rückgabewert**

0
: Kein Fehler

-1
: Fehler. Die Fehlerart wird in *errorinfo* hinterlegt.

## 3.9 ft_reqterm - Auftrag abschließen

*ft_reqterm()* schließt einen asynchronen Dateiübertragungsauftrag ab. Dies ist nur möglich, wenn der Auftrag den Status „abgebrochen“ oder „beendet“ hat. *ft_reqterm()* löscht die zugehörige Datei mit Verwaltungsinformationen. Danach ist die Request-Id gelöscht und der Auftrag kann nicht mehr angesprochen werden.

### Syntax

```
#include <ftapi.h>

int ft_reqterm(const void *session,          /* Eingabe */
               long rid,                     /* Eingabe */
               struct ft_err *errorinfo,
               void *options);               /* Eingabe */
```

### Parameter

session
: Sitzungs-Id der Sitzung, in der der Übertragungsauftrag abgeschlossen werden soll.

rid
: Request-Id des Auftrags, der abgeschlossen werden soll.

  Wenn der Auftrag in einer anderen Sitzung erteilt wurde, muss der aktuellen Sitzung dasselbe Arbeitsverzeichnis zugeordnet sein wie der Sitzung, in der der Auftrag erteilt wurde.

errorinfo
: Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17).

  Die Angabe des Parameters ist optional. Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert NULL angeben.

options
: Die Angabe des Parameters *options* ist optional. Wird der Wert NULL angegeben, dann ist das Meldungsverhalten der Programmschnittstelle kompatibel zur Programmschnittstelle von openFT < V10.
  Alternativ können durch die Angabe der Struktur *ft_options* (siehe Abschnitt „ft_options“ auf Seite 18) das openFT-Meldungsnummernschema ab openFT V10 und die Funktionserweiterungen aktiviert werden.

**Rückgabewert**

| | |
|---|---|
| 0 | Kein Fehler |
| -1 | Fehler. Die Fehlerart wird in *errorinfo* hinterlegt |

## 3.10 ft_sdopen - Ermitteln der Attribute aller Dateien eines Dateiverzeichnisses starten

*ft_sdopen()* startet die Ermittlung der Attribute aller Dateien eines Dateiverzeichnisses im fernen System.

Dateiverzeichnisnamen dürfen die in der Struktur *ft_prop* im Feld *maxrfnsize* angegebene Länge nicht überschreiten (siehe Abschnitt „ft_properties - Eigenschaften der Programmschnittstelle ermitteln“ auf Seite 31).

### Syntax

```
#include <ftapi.h>

void *ft_sdopen(const struct ft_admission *admis,       /* Eingabe */
                struct ft_shwpar *par,
                struct ft_err *errorinfo
                void *options);                        /* Eingabe */
```

### Parameter

admis
: Angaben für das ferne System (siehe Abschnitt „ft_admission“ auf Seite 15).

par
: Angaben für den Auftrag, die Sie mit der Struktur *ft_shwpar* bekanntgeben:

```
struct ft_shwpar
{
   int  shwparvers;      /* Eingabe */
   char *fn;             /* Eingabe */
   char *mgmtpasswd;     /* Eingabe */
   char *fud;            /* Eingabe */
   int  fudlen;          /* Eingabe */
};
```

Die Felder der Struktur *ft_shwpar* haben folgende Bedeutung:

shwparvers
: Version der Datenstruktur.
*shwparvers* muss mit dem Wert `FT_SPARV1` oder `FT_SPARV2` versorgt werden.

fn Name des Dateiverzeichnisses, für dessen Dateien die Attribute ermittelt werden sollen.

Absolute und relative Pfadangaben sind erlaubt. Relative Pfadangaben beziehen sich auf die im Berechtigungsprofil festgelegte Benutzerkennung, wenn die FTAC-Funktion eingesetzt wird, sonst auf das HOME-Verzeichnis, siehe Seite 12.

mgmtpasswd
Kennwort des Dateiverzeichnisses, falls es mit einem Kennwort geschützt ist.

fud Adresse eines Datenbereichs für die sogenannten „Further Details“, die im Fehlerfall eine genauere Fehlerursache angeben können.
Bei Angabe von NULL wird keine weiterführende Fehlerursache ausgegeben. Der Parameter *fud* steht nur dann zur Verfügung, wenn *shwparvers* auf den Wert `FT_SPARV2` gesetzt wird und beim Aufruf von *ft_showdir* der Parameter *options* angegeben ist.

fudlen
Länge des Datenbereichs von *fud*.
Der Parameter *fudlen* steht nur dann zur Verfügung, wenn *shwparvers* auf den Wert `FT_SPARV2` gesetzt wird und beim Aufruf von *ft_showdir* der Parameter *options* angegeben ist.

errorinfo
Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17). Die Angabe des Parameters ist optional.

Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert `NULL` angeben.

options
Die Angabe des Parameters *options* ist obligatorisch. Der Aufbau der Struktur *ft_options* ist im Abschnitt „Version der Programmschnittstelle“ auf Seite 18 beschrieben.

**Rückgabewert**

id Id des Aufrufs. Diese muss bei *ft_sdinfo()* und *ft_sdclose()* angegeben werden.

NULL Fehler. Die Fehlerart wird in *errorinfo* hinterlegt.
Der Aufruf von *ft_sdclose()* ist im Fehlerfall nicht notwendig.

## 3.11 ft_sdinfo - Dateiattribute auslesen

*ft_sdinfo*() liest die mit *ft_sdopen()* ermittelten Dateiattribute eines Dateiverzeichnisses im fernen System aus. Sie können *ft_sdinfo* mehrfach aufrufen. Es werden dann jeweils die nächsten, noch nicht gelesenen Daten in den Puffer *buf* geschrieben. Wenn alle Daten gelesen wurden, ist der Rückgabewert 0.

### Syntax

```
#include <ftapi.h>

int ft_sdinfo(void *id,                              /* Eingabe */
              struct ft_fileinfo *buf,
              int bufsize,                           /* Eingabe */
              struct ft_err *errorinfo);
```

### Parameter

id Id des Aufrufs (Rückgabewert von *ft_sdopen*)

buf Bereich, in den die Dateiattribute geschrieben werden. Dieser Bereich besteht aus Elementen mit der die Struktur *ft_fileinfo*:

```
#define ACC_LEN       65
#define INFO_FN_LEN  257
#define LQ_LEN        81
#define USER_LEN      68

struct ft_fileinfo
{
   int   ftshowivers;                /* Eingabe */
   char  fn[INFO_FN_LEN];            /* Ausgabe */
   enum  ft_ftype filetype;          /* Ausgabe */
   enum  ft_charset charset;         /* Ausgabe */
   enum  ft_rform recordform;        /* Ausgabe */
   long  recsize;                    /* Ausgabe */
   enum  ft_available availability;  /* Ausgabe */
   int   acccess;                    /* Ausgabe */
   char  accout[ACC_LEN];            /* Ausgabe */
   long  size;                       /* Ausgabe */
   long  maxsize;                    /* Ausgabe */
   char  legalqual[LQ_LEN];          /* Ausgabe */
   char  cre_user[USER_LEN];         /* Ausgabe */
   long  cre_date;                   /* Ausgabe */
   char  mod_user[USER_LEN];         /* Ausgabe */
   long  mod_date;                   /* Ausgabe */
   char  rea_user[USER_LEN];         /* Ausgabe */
   long  rea_date;                   /* Ausgabe */
```

```
   char  atm_user[USER_LEN];          /* Ausgabe */
   long  atm_date;                    /* Ausgabe */
   long  long fsize;                  /* Ausgabe */
   long  long fmaxsize;               /* Ausgabe */
};
```

Die Felder der Struktur *ft_fileinfo* haben folgende Bedeutung:

ftshowivers
: Version der Datenstruktur.
*ftshowivers* muss mit dem Wert `FT_SHOWIV2` versorgt werden. Es ist ausreichend, wenn *ftshowivers* in der ersten übergebenen Datenstruktur gesetzt ist.

fn
: Dateiname oder Dateiverzeichnisname

filetype
: Dateityp:

  `FT_TYPEUNKN`
  : Dateityp unbekannt

  `FT_BIN`
  : Binärdatei

  `FT_DIR`
  : Dateiverzeichnis

  `FT_TXT`
  : Textdatei

charset
: Zeichensatz (nur bei Textdateien):

  `FT_NOSET`
  : Zeichensatz unbekannt

  `FT_VISIBLE`
  : Die Datei kann Zeichen aus dem G0-Set von ISO646 enthalten.

  `FT_IA5`
  : Die Datei kann Zeichen aus dem dem C0-Set und dem G0-Set von ISO646 enthalten.

  `FT_GRAPHIC`
  : Die Datei kann Zeichen aus dem G0-Set von ISO646 oder aus dem G0-Set von ISO8859-1 und dem G1-Set von ISO8859-1 enthalten.

  `FT_GENERAL`
  : Die Datei kann Zeichen aus dem C0-Set von ISO646, aus dem G0-Set von ISO646 oder ISO8859-1 und aus dem G1-Set von ISO8859-1 enthalten.

recordform
: Satzformat:

  FT_NOFORM
  : Satzformat unbekannt

  FT_VARIABLE
  : variabel lange Sätze

  FT_FIXED
  : einheitlich lange Sätze

  FT_UNDEF
  : undefinierte Satzlänge

recsize
: maximale Satzlänge oder 0, wenn die maximale Satzlänge unbekannt ist

availability
: Verfügbarkeit der Datei:

  FT_NOAVAIL
  : Die Verfügbarkeit ist nicht festgelegt.

  FT_AVAILIMM
  : Die Datei ist sofort verfügbar.

  FT_AVAILNIMM
  : Die Datei ist nicht sofort verfügbar.

access
: Zugriffsrechte. Das Recht ist vorhanden, wenn das Bit gesetzt ist.
  Folgende Bits sind definiert:

  FT_ACCR
  : Die Datei darf gelesen werden.

  FT_ACCI
  : Dateneinheiten dürfen in die Datei eingefügt werden.

  FT_ACCP
  : Die Datei darf überschrieben werden.

  FT_ACCX
  : Die Datei darf erweitert werden, d.h. Daten können an die Datei angefügt werden.

  FT_ACCE
  : Dateneinheiten dürfen aus der Datei gelöscht werden.

  FT_ACCA
  : Dateiattribute dürfen gelesen werden.

  FT_ACCC
  : Dateiattribute dürfen geändert werden.

  FT_ACCD
  : Die Datei darf gelöscht werden.

account
: Abrechnungsnummer, über die die Kosten im fernen System verrechnet werden

size
: aktuelle Dateigröße in Bytes oder -1, wenn die Dateigröße unbekannt ist. Auf Systemen, bei denen die Größe einer Variablen des Typs `long` 32 Bit beträgt, wird der Wert für die Dateigröße abgeschnitten, wenn er nicht in das Feld passt. Der vollständige Wert für die Dateigröße ist im Feld *fsize* zu finden.

maxsize
: erlaubte Dateigröße in Bytes oder -1, wenn die erlaubte Dateigröße unbekannt ist. Auf Systemen, bei denen die Größe einer Variablen des Typs `long` 32 Bit beträgt, wird der Wert für die Dateigröße abgeschnitten, wenn er nicht in das Feld passt. Der vollständige Wert für die Dateigröße ist im Feld *fmaxsize* zu finden.

legalqual
: rechtliche Bestimmung

cre_user
: Dateibenutzer, der die Datei erstellt hat

cre_date
: Zeitpunkt, zu dem die Datei erstellt wurde, oder 0, wenn der Zeitpunkt unbekannt ist.
Die Angabe erfolgt im internen Zeitformat (Sekunden seit dem 1.1.1970 00:00:00).

mod_user
: Dateibenutzer, der den Dateiinhalt zuletzt geändert hat

mod_date
: Zeitpunkt, zu dem der Dateiinhalt zuletzt geändert wurde, oder 0, wenn der Zeitpunkt unbekannt ist.
Die Angabe erfolgt im internen Zeitformat (Sekunden seit dem 1.1.1970 00:00:00).

rea_user
: Dateibenutzer, der die Datei zuletzt gelesen hat

rea_date
: Zeitpunkt, zu dem die Datei zuletzt gelesen wurde, oder 0, wenn der Zeitpunkt unbekannt ist.
Die Angabe erfolgt im internen Zeitformat (Sekunden seit dem 1.1.1970 00:00:00).

atm_user
: Dateibenutzer, der die Dateiattribute zuletzt geändert hat

atm_date
: Zeitpunkt, zu dem die Dateiattribute zuletzt geändert wurden, oder 0, wenn der Zeitpunkt unbekannt ist.
Die Angabe erfolgt im internen Zeitformat (Sekunden seit dem 1.1.1970 00:00:00).

fsize
: aktuelle Dateigröße in Bytes oder -1, wenn die Dateigröße unbekannt ist. Der Parameter *fsize* steht nur dann zur Verfügung, wenn *ftshowivers* auf den Wert `FT_SHOWIV2` gesetzt wird und beim Aufruf von *ft_showdir* der Parameter *options* angegeben ist.

fmaxsize
: erlaubte Dateigröße in Bytes oder -1, wenn die erlaubte Dateigröße unbekannt ist. Der Parameter *fmaxsize* steht nur dann zur Verfügung, wenn *ftshowivers* auf den Wert `FT_SHOWIV2` gesetzt wird und beim Aufruf von *ft_showdir* der Parameter *options* angegeben ist.

bufsize
: Größe von *buf*, d.h. maximale Anzahl der Elemente mit der Struktur *ft_fileinfo*, die in *buf* passen.

errorinfo
: Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17). Die Angabe des Parameters ist optional.
Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert `NULL` angeben.

**Rückgabewert**

| | |
|---|---|
| n | Anzahl der in den Puffer *buf* geschriebenen Elemente. |
| 0 | Es stehen keine weiteren Daten zur Verfügung. |
| -1 | Fehler. Die Fehlerart wird in *errorinfo* hinterlegt. |

## 3.12 ft_sdclose - Ermitteln der Dateiattribute beenden

*ft_sdclo*se() beendet das Auslesen der Dateiattribute, deren Ermittlung mit *ft_sdopen()* gestartet wurde. Diese Funktion muss nach erfolgreichem Aufruf von f*t_sdopen()* als letzter Schritt aufgerufen werden. *ft_sdclose()* gibt nicht mehr benötigte Ressourcen frei. Anschließend können Sie sich nicht mehr auf diese Id beziehen.

### Syntax

```
#include <ftapi.h>

int ft_sdclose(void *id,                          /* Eingabe/
               struct ft_err *errorinfo);
```

### Parameter

id Id des Aufrufs (Rückgabewert von *ft_sdopen*)

errorinfo
Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17). Die Angabe des Parameters ist optional.
Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert `NULL` angeben.

### Rückgabewert

0 Kein Fehler.

-1 Fehler. Die Fehlerart wird in *errorinfo* hinterlegt.

## 3.13 ft_show - Attribute einer Datei oder eines Dateiverzeichnisses ermitteln

*ft_show()* ermittelt die Attribute einer einzelnen Datei oder eines Dateiverzeichnisses im fernen System. Die Attribute mehrerer Dateien ermittelt die Funktion *ft_showdir()*.

Dateinamen bzw. Dateiverzeichnisnamen dürfen die in der Struktur *ft_prop* im Feld *maxrfnsize* angegebene Länge nicht überschreiten (siehe Abschnitt „ft_properties - Eigenschaften der Programmschnittstelle ermitteln“ auf Seite 31).

### Syntax

```
#include <ftapi.h>

int ft_show(const struct ft_admission *admis,    /* Eingabe */
            const struct ft_shwpar *par,         /* Eingabe */
            struct ft_fileinfo *info,
            struct ft_err *errorinfo,
            void *options);                      /* Eingabe */
```

### Parameter

admis

Angaben für das ferne System (siehe Abschnitt „ft_admission“ auf Seite 15).

par Angaben für den Auftrag, die Sie mit der Struktur *ft_shwpar* bekanntgeben:

```
struct ft_shwpar
{
   int  shwparvers;      /* Eingabe */
   char *fn;             /* Eingabe */
   char *mgmtpasswd;     /* Eingabe */
   char *fud;            /* Eingabe */
   int  fudlen;          /* Eingabe */
};
```

Die Felder der Struktur *ft_shwpar* haben folgende Bedeutung:

shwparvers

Version der Datenstruktur.
*shwparvers* muss mit dem Wert `FT_SPARV1` oder `FT_SPARV2` versorgt werden.

fn Name der Datei oder des Dateiverzeichnisses, deren Attribute ermittelt werden sollen.

Absolute und relative Pfadangaben sind erlaubt. Relative Pfadangaben beziehen sich auf die im Berechtigungsprofil festgelegte Benutzerkennung, wenn die FTAC-Funktion eingesetzt wird, sonst auf das HOME-Verzeichnis, siehe Seite 12.

mgmtpasswd
Kennwort der Datei oder des Dateiverzeichnisses, falls sie/es mit einem Kennwort geschützt ist.

fud Adresse eines Datenbereichs für die sogenannten „Further Details“, die im Fehlerfall eine genauere Fehlerursache angeben können.
Bei Angabe von NULL wird keine weiterführende Fehlerursache ausgegeben. Der Parameter *fud* steht nur dann zur Verfügung, wenn *shwparvers* auf den Wert `FT_SPARV2` gesetzt wird und beim Aufruf von *ft_show* der Parameter *options* angegeben ist.

fudlen
Länge des Datenbereichs von *fud*.
Der Parameter *fudlen* steht nur dann zur Verfügung, wenn *shwparvers* auf den Wert `FT_SPARV2` gesetzt wird und beim Aufruf von *ft_show* der Parameter *options* angegeben ist.

info

Bereich, in den die Dateiattribute geschrieben werden. Dazu dient die Struktur *ft_fileinfo*:

```
#define ACC_LEN       65
#define INFO_FN_LEN  257
#define LQ_LEN        81
#define USER_LEN      68

struct ft_fileinfo
{
   int   ftshowivers;                /* Eingabe */
   char  fn[INFO_FN_LEN];            /* Ausgabe */
   enum  ft_ftype filetype;          /* Ausgabe */
   enum  ft_charset charset;         /* Ausgabe */
   enum  ft_rform recordform;        /* Ausgabe */
   long  recsize;                    /* Ausgabe */
   enum  ft_available availability;  /* Ausgabe */
   int   acccess;                    /* Ausgabe */
   char  accout[ACC_LEN];            /* Ausgabe */
   long  size;                       /* Ausgabe */
   long  maxsize;                    /* Ausgabe */
   char  legalqual[LQ_LEN];          /* Ausgabe */
   char  cre_user[USER_LEN];         /* Ausgabe */
```

```
   long  cre_date;                    /* Ausgabe */
   char  mod_user[USER_LEN];          /* Ausgabe */
   long  mod_date;                    /* Ausgabe */
   char  rea_user[USER_LEN];          /* Ausgabe */
   long  rea_date;                    /* Ausgabe */
   char  atm_user[USER_LEN];          /* Ausgabe */
   long  atm_date;                    /* Ausgabe */
   long  long fsize;                  /* Ausgabe */
   long  long fmaxsize;               /* Ausgabe */
};
```

Die Felder der Struktur *ft_fileinfo* haben folgende Bedeutung:

ftshowivers
: Version der Datenstruktur.
*ftshowivers* muss mit dem Wert `FT_SHOWIV1` oder `FT_SHOWIV2` versorgt werden.

fn
: Dateiname oder Dateiverzeichnisname

filetype
: Dateityp:

  `FT_TYPEUNKN`
  : Dateityp unbekannt

  `FT_BIN`
  : Binärdatei

  `FT_DIR`
  : Dateiverzeichnis

  `FT_TXT`
  : Textdatei

charset
: Zeichensatz (nur bei Textdateien):

  `FT_NOSET`
  : Zeichensatz unbekannt

  `FT_VISIBLE`
  : Die Datei kann Zeichen aus dem G0-Set von ISO646 enthalten.

  `FT_IA5`
  : Die Datei kann Zeichen aus dem dem C0-Set und dem G0-Set von ISO646 enthalten.

  `FT_GRAPHIC`
  : Die Datei kann Zeichen aus dem G0-Set von ISO646 oder aus dem G0-Set von ISO8859-1 und dem G1-Set von ISO8859-1 enthalten.

FT_GENERAL
: Die Datei kann Zeichen aus dem C0-Set von ISO646, aus dem G0-Set von ISO646 oder ISO8859-1 und aus dem G1-Set von ISO8859-1 enthalten.

recordform
: Satzformat:

FT_NOFORM
: Satzformat unbekannt

FT_VARIABLE
: variabel lange Sätze

FT_FIXED
: einheitlich lange Sätze

FT_UNDEF
: undefinierte Satzlänge

recsize
: maximale Satzlänge oder 0, wenn die maximale Satzlänge unbekannt ist

availability
: Verfügbarkeit der Datei:

FT_NOAVAIL
: Die Verfügbarkeit ist nicht festgelegt.

FT_AVAILIMM
: Die Datei ist sofort verfügbar.

FT_AVAILNIMM
: Die Datei ist nicht sofort verfügbar.

access
: Zugriffsrechte. Das Recht ist vorhanden, wenn das Bit gesetzt ist. Folgende Bits sind definiert:

FT_ACCR
: Die Datei darf gelesen werden.

FT_ACCI
: Dateneinheiten dürfen in die Datei eingefügt werden.

FT_ACCP
: Die Datei darf überschrieben werden.

FT_ACCX
: Die Datei darf erweitert werden, d.h. Daten können an die Datei angefügt werden.

FT_ACCE
: Dateneinheiten dürfen aus der Datei gelöscht werden.

FT_ACCA
: Dateiattribute dürfen gelesen werden.

FT_ACCC
: Dateiattribute dürfen geändert werden.

FT_ACCD
: Die Datei darf gelöscht werden.

account
: Abrechnungsnummer, über die die Kosten im fernen System verrechnet werden

size
: aktuelle Dateigröße in Bytes oder -1, wenn die Dateigröße unbekannt ist. Auf Systemen, bei denen die Größe einer Variablen des Typs `long` 32 Bit beträgt, wird der Wert für die Dateigröße abgeschnitten, wenn er nicht in das Feld passt. Der vollständige Wert für die Dateigröße ist im Feld *fsize* zu finden.

maxsize
: erlaubte Dateigröße in Bytes oder -1, wenn die erlaubte Dateigröße unbekannt ist. Auf Systemen, bei denen die Größe einer Variablen des Typs `long` 32 Bit beträgt, wird der Wert für die Dateigröße abgeschnitten, wenn er nicht in das Feld passt. Der vollständige Wert für die Dateigröße ist im Feld *fmaxsize* zu finden.

legalqual
: rechtliche Bestimmung

cre_user
: Dateibenutzer, der die Datei erstellt hat

cre_date
: Zeitpunkt, zu dem die Datei erstellt wurde, oder 0, wenn der Zeitpunkt unbekannt ist.
Die Angabe erfolgt im internen Zeitformat (Sekunden seit dem 1.1.1970 00:00:00).

mod_user
: Dateibenutzer, der den Dateiinhalt zuletzt geändert hat

mod_date
: Zeitpunkt, zu dem der Dateiinhalt zuletzt geändert wurde, oder 0, wenn der Zeitpunkt unbekannt ist.
Die Angabe erfolgt im internen Zeitformat (Sekunden seit dem 1.1.1970 00:00:00).

rea_user
: Dateibenutzer, der die Datei zuletzt gelesen hat

rea_date
: Zeitpunkt, zu dem die Datei zuletzt gelesen wurde, oder 0, wenn der Zeitpunkt unbekannt ist.
Die Angabe erfolgt im internen Zeitformat (Sekunden seit dem 1.1.1970 00:00:00).

atm_user
: Dateibenutzer, der die Dateiattribute zuletzt geändert hat

atm_date
: Zeitpunkt, zu dem die Dateiattribute zuletzt geändert wurden, oder 0, wenn der Zeitpunkt unbekannt ist.
Die Angabe erfolgt im internen Zeitformat (Sekunden seit dem 1.1.1970 00:00:00).

fsize
: aktuelle Dateigröße in Bytes oder -1, wenn die Dateigröße unbekannt ist.
Der Parameter *fsize* steht nur dann zur Verfügung, wenn *ftshowivers* auf den Wert `FT_SHOWIV2` gesetzt wird und beim Aufruf von *ft_show* der Parameter *options* angegeben ist.

fmaxsize
: erlaubte Dateigröße in Bytes oder -1, wenn die erlaubte Dateigröße unbekannt ist. Der Parameter *fmaxsize* steht nur dann zur Verfügung, wenn *ftshowivers* auf den Wert `FT_SHOWIV2` gesetzt wird und beim Aufruf von *ft_show* der Parameter *options* angegeben ist.

errorinfo
: Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17).
Die Angabe des Parameters ist optional. Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert NULL angeben.

options
: Die Angabe des Parameters *options* ist optional. Wird der Wert NULL angegeben, dann ist das Meldungsverhalten der Programmschnittstelle kompatibel zur Programmschnittstelle von openFT < V10.
Alternativ können durch die Angabe der Struktur *ft_options* (siehe Abschnitt „ft_options“ auf Seite 18) das openFT-Meldungsnummernschema ab openFT V10 und die Funktionserweiterungen aktiviert werden.

**Rückgabewert**

0 Kein Fehler

-1 Fehler. Über die Datei wurden keine Informationen geliefert.
Die Fehlerart wird in *errorinfo* hinterlegt.

## 3.14 ft_showdir - Attribute aller Dateien eines Verzeichnisses ermitteln

*ft_showdir()* ermittelt die Attribute der Dateien eines Dateiverzeichnisses im fernen System. Dabei werden jeweils soviele Attributdatensätze ermittelt, wie Sie im Parameter *bufsize* angegeben haben. Sind im Dateiverzeichnis auf dem fernen System mehr Daten vorhanden, müssen Sie *ft_showdir()* mehrfach aufrufen. Beachten Sie, dass Sie keine Auswahl innerhalb des Verzeichnisses treffen können.

Die Attribute einer einzelnen Datei/Dateiverzeichnisses werden mit der Funktion *ft_show()* ermittelt.

Dateiverzeichnisnamen dürfen die in der Struktur *ft_prop* im Feld *maxrfnsize* angegebene Länge nicht überschreiten (siehe Abschnitt „ft_properties - Eigenschaften der Programmschnittstelle ermitteln“ auf Seite 31).

### Syntax

```
#include <ftapi.h>

long ft_showdir(const struct ft_admission *admis,/* Eingabe */
                const struct ft_shwpar *par,     /* Eingabe */
                struct ft_fileinfo *buf,
                int bufsize,                     /* Eingabe */
                struct ft_err *errorinfo,
                void *options);                  /* Eingabe */
```

### Parameter

admis

Angaben für das ferne System (siehe Abschnitt „ft_admission“ auf Seite 15).

par Angaben für den Auftrag, die Sie mit der Struktur *ft_shwpar* bekanntgeben:

```
struct ft_shwpar
{
   int  shwparvers;      /* Eingabe */
   char *fn;             /* Eingabe */
   char *mgmtpasswd;     /* Eingabe */
   char *fud;            /* Eingabe */
   int  fudlen;          /* Eingabe */
};
```

Die Felder der Struktur *ft_shwpar* haben folgende Bedeutung:

shwparvers
: Version der Datenstruktur.
*shwparvers* muss mit dem Wert `FT_SPARV1` oder `FT_SPARV2` versorgt werden.

fn
: Name des Dateiverzeichnisses, für dessen Dateien die Attribute ermittelt werden sollen.

  Absolute und relative Pfadangaben sind erlaubt. Relative Pfadangaben beziehen sich auf die im Berechtigungsprofil festgelegte Benutzerkennung, wenn die FTAC-Funktion eingesetzt wird, sonst auf das HOME-Verzeichnis, siehe Seite 12.

mgmtpasswd
: Kennwort des Dateiverzeichnisses, falls es mit einem Kennwort geschützt ist.

fud
: Adresse eines Datenbereichs für die sogenannten „Further Details“, die im Fehlerfall eine genauere Fehlerursache angeben können.
Bei Angabe von NULL wird keine weiterführende Fehlerursache ausgegeben. Der Parameter *fud* steht nur dann zur Verfügung, wenn *shwparvers* auf den Wert `FT_SPARV2` gesetzt wird und beim Aufruf von *ft_showdir* der Parameter *options* angegeben ist.

fudlen
: Länge des Datenbereichs von *fud*.
Der Parameter *fudlen* steht nur dann zur Verfügung, wenn *shwparvers* auf den Wert `FT_SPARV2` gesetzt wird und beim Aufruf von *ft_showdir* der Parameter *options* angegeben ist.

buf
: Bereich, in den die Dateiattribute geschrieben werden. Dieser Bereich besteht aus Elementen mit der die Struktur *ft_fileinfo*:

```
#define ACC_LEN       65
#define INFO_FN_LEN  257
#define LQ_LEN        81
#define USER_LEN      68

struct ft_fileinfo
{
   int   ftshowivers;                /* Eingabe */
   char  fn[INFO_FN_LEN];            /* Ausgabe */
   enum  ft_ftype filetype;          /* Ausgabe */
   enum  ft_charset charset;         /* Ausgabe */
   enum  ft_rform recordform;        /* Ausgabe */
   long  recsize;                    /* Ausgabe */
   enum  ft_available availability;  /* Ausgabe */
```

```
   int   acccess;                    /* Ausgabe */
   char  accout[ACC_LEN];            /* Ausgabe */
   long  size;                       /* Ausgabe */
   long  maxsize;                    /* Ausgabe */
   char  legalqual[LQ_LEN];          /* Ausgabe */
   char  cre_user[USER_LEN];         /* Ausgabe */
   long  cre_date;                   /* Ausgabe */
   char  mod_user[USER_LEN];         /* Ausgabe */
   long  mod_date;                   /* Ausgabe */
   char  rea_user[USER_LEN];         /* Ausgabe */
   long  rea_date;                   /* Ausgabe */
   char  atm_user[USER_LEN];         /* Ausgabe */
   long  atm_date;                   /* Ausgabe */
   long  long fsize;                 /* Ausgabe */
   long  long fmaxsize;              /* Ausgabe */
};
```

Die Felder der Struktur *ft_fileinfo* haben folgende Bedeutung:

ftshowivers
: Version der Datenstruktur.
*ftshowivers* muss mit dem Wert `FT_SHOWIV1` oder `FT_SHOWIV2` versorgt werden. Es ist ausreichend, wenn *ftshowivers* in der ersten übergebenen Datenstruktur gesetzt ist.

fn
: Dateiname oder Dateiverzeichnisname

filetype
: Dateityp:

  `FT_TYPEUNKN`
  : Dateityp unbekannt

  `FT_BIN`
  : Binärdatei

  `FT_DIR`
  : Dateiverzeichnis

  `FT_TXT`
  : Textdatei

charset
: Zeichensatz (nur bei Textdateien):

  `FT_NOSET`
  : Zeichensatz unbekannt

  `FT_VISIBLE`
  : Die Datei kann Zeichen aus dem G0-Set von ISO646 enthalten.

FT_IA5
: Die Datei kann Zeichen aus dem dem C0-Set und dem G0-Set von ISO646 enthalten.

FT_GRAPHIC
: Die Datei kann Zeichen aus dem G0-Set von ISO646 oder aus dem G0-Set von ISO8859-1 und dem G1-Set von ISO8859-1 enthalten.

FT_GENERAL
: Die Datei kann Zeichen aus dem C0-Set von ISO646, aus dem G0-Set von ISO646 oder ISO8859-1 und aus dem G1-Set von ISO8859-1 enthalten.

recordform
: Satzformat:

FT_NOFORM
: Satzformat unbekannt

FT_VARIABLE
: variabel lange Sätze

FT_FIXED
: einheitlich lange Sätze

FT_UNDEF
: undefinierte Satzlänge

recsize
: maximale Satzlänge oder 0, wenn die maximale Satzlänge unbekannt ist

availability
: Verfügbarkeit der Datei:

FT_NOAVAIL
: Die Verfügbarkeit ist nicht festgelegt.

FT_AVAILIMM
: Die Datei ist sofort verfügbar.

FT_AVAILNIMM
: Die Datei ist nicht sofort verfügbar.

access
: Zugriffsrechte. Das Recht ist vorhanden, wenn das Bit gesetzt ist. Folgende Bits sind definiert:

  `FT_ACCR`
  : Die Datei darf gelesen werden.

  `FT_ACCI`
  : Dateneinheiten dürfen in die Datei eingefügt werden.

  `FT_ACCP`
  : Die Datei darf überschrieben werden.

  `FT_ACCX`
  : Die Datei darf erweitert werden, d.h. Daten können an die Datei angefügt werden.

  `FT_ACCE`
  : Dateneinheiten dürfen aus der Datei gelöscht werden.

  `FT_ACCA`
  : Dateiattribute dürfen gelesen werden.

  `FT_ACCC`
  : Dateiattribute dürfen geändert werden.

  `FT_ACCD`
  : Die Datei darf gelöscht werden.

account
: Abrechnungsnummer, über die die Kosten im fernen System verrechnet werden

size
: aktuelle Dateigröße in Bytes oder -1, wenn die Dateigröße unbekannt ist. Auf Systemen, bei denen die Größe einer Variablen des Typs `long` 32 Bit beträgt, wird der Wert für die Dateigröße abgeschnitten, wenn er nicht in das Feld passt. Der vollständige Wert für die Dateigröße ist im Feld *fsize* zu finden.

maxsize
: erlaubte Dateigröße in Bytes oder -1, wenn die erlaubte Dateigröße unbekannt ist. Auf Systemen, bei denen die Größe einer Variablen des Typs `long` 32 Bit beträgt, wird der Wert für die Dateigröße abgeschnitten, wenn er nicht in das Feld passt. Der vollständige Wert für die Dateigröße ist im Feld *fmaxsize* zu finden.

legalqual
: rechtliche Bestimmung

cre_user
: Dateibenutzer, der die Datei erstellt hat

cre_date
: Zeitpunkt, zu dem die Datei erstellt wurde, oder 0, wenn der Zeitpunkt unbekannt ist.
Die Angabe erfolgt im internen Zeitformat (Sekunden seit dem 1.1.1970 00:00:00).

mod_user
: Dateibenutzer, der den Dateiinhalt zuletzt geändert hat

mod_date
: Zeitpunkt, zu dem der Dateiinhalt zuletzt geändert wurde, oder 0, wenn der Zeitpunkt unbekannt ist.
Die Angabe erfolgt im internen Zeitformat (Sekunden seit dem 1.1.1970 00:00:00).

rea_user
: Dateibenutzer, der die Datei zuletzt gelesen hat

rea_date
: Zeitpunkt, zu dem die Datei zuletzt gelesen wurde, oder 0, wenn der Zeitpunkt unbekannt ist.
Die Angabe erfolgt im internen Zeitformat (Sekunden seit dem 1.1.1970 00:00:00).

atm_user
: Dateibenutzer, der die Dateiattribute zuletzt geändert hat

atm_date
: Zeitpunkt, zu dem die Dateiattribute zuletzt geändert wurden, oder 0, wenn der Zeitpunkt unbekannt ist.
Die Angabe erfolgt im internen Zeitformat (Sekunden seit dem 1.1.1970 00:00:00).

fsize
: aktuelle Dateigröße in Bytes oder -1, wenn die Dateigröße unbekannt ist.
Der Parameter *fsize* steht nur dann zur Verfügung, wenn *ftshowivers* auf den Wert `FT_SHOWIV2` gesetzt wird und beim Aufruf von *ft_showdir* der Parameter *options* angegeben ist.

fmaxsize
: erlaubte Dateigröße in Bytes oder -1, wenn die erlaubte Dateigröße unbekannt ist. Der Parameter *fmaxsize* steht nur dann zur Verfügung, wenn *ftshowivers* auf den Wert `FT_SHOWIV2` gesetzt wird und beim Aufruf von *ft_showdir* der Parameter *options* angegeben ist.

bufsize
: Größe von *buf*, d.h. maximale Anzahl der Elemente mit der Struktur *ft_fileinfo*, die in *buf* passen.

errorinfo

Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17).
Die Angabe des Parameters ist optional. Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert NULL angeben.

options

Die Angabe des Parameters *options* ist optional. Wird der Wert NULL angegeben, dann ist das Meldungsverhalten der Programmschnittstelle kompatibel zur Programmschnittstelle von openFT < V10.
Alternativ können durch die Angabe der Struktur *ft_options* (siehe Abschnitt „ft_options“ auf Seite 18) das openFT-Meldungsnummernschema ab openFT V10 und die Funktionserweiterungen aktiviert werden.

**Rückgabewert**

*n* Anzahl der gefundenen Dateien im fernen Dateiverzeichnis ($n \geq 0$).
Wenn *n* größer als *bufsize* ist, werden die ersten *bufsize* Einträge in *buf* hinterlegt.
Wenn *buf* den Wert NULL hat, verhält sich der Funktionsaufruf, als ob *bufsize* den Wert 0 hätte.

-1 Fehler. Über das Dateiverzeichnis wurden keine Informationen geliefert.
Die Fehlerart wird in *errorinfo* hinterlegt.

## 3.15 ft_transfer - Datei übertragen

*ft_transfer()* sendet eine Datei ins ferne System oder holt eine Datei aus dem fernen System.

Um Dateien **synchron** zu übertragen, muss der Parameter *synchron* den Wert `FT_SYNC` enthalten.

Um Dateien **asynchron** zu übertragen, muss der Parameter *synchron* den Wert `FT_ASYNC` enthalten.
Bei asynchronen Dateiübertragungen liefert die Funktion *ft_transfer()* eine Request-Id zurück, die Sie angeben müssen, wenn Sie sich auf diesen Auftrag beziehen.

Dateinamen dürfen die in der Struktur *ft_prop* im Feld *maxlfnsize* bzw. *maxrfnsize* angegebene Länge nicht überschreiten (siehe Abschnitt „ft_properties - Eigenschaften der Programmschnittstelle ermitteln“ auf Seite 31).

### Syntax

```
#include <ftapi.h>

long ft_transfer(const void *session,                    /* Eingabe */
                 const struct ft_admission *admis,       /* Eingabe */
                 const struct ft_transpar *par,          /* Eingabe */
                 struct ft_err *errorinfo,
                 void *options);                         /* Eingabe */
```

### Parameter

session
: Bei asynchroner Übertragung:
Sitzungsnummer der Sitzung, in der der Übertragungsauftrag ausgeführt werden soll.

  Bei synchroner Übertragung:
*session* muss den Wert NULL haben.

admis
: Angaben für das ferne System (siehe Abschnitt „ft_admission“ auf Seite 15).

par

Angaben für den Auftrag, die Sie mit der Struktur *ft_transpar* bekanntgeben:

```
struct ft_transpar
{
   int    ftparvers;                  /* Eingabe */
   enum   ft_direction direction;     /* Eingabe */
   enum   ft_sync synchron;           /* Eingabe */
   char   *locfn;                     /* Eingabe */
   char   *remfn;                     /* Eingabe */
   enum   ft_filetype filetype;       /* Eingabe */
   enum   ft_writemode writemode;     /* Eingabe */
   enum   ft_compress compress;       /* Eingabe */
   char   *filepasswd;                /* Eingabe */
   char   *locsuccproc;               /* Eingabe */
   char   *locfailproc;               /* Eingabe */
   char   *remsuccproc;               /* Eingabe */
   char   *remfailproc;               /* Eingabe */
   long   maxrecsize;                 /* Eingabe */
   long   cantime;                    /* Eingabe */
   long   starttime;                  /* Eingabe */
   enum   ft_prio priority;           /* Eingabe */
   enum   ft_transpar transparent;    /* Eingabe */
   enum   ft_encrypt encryption;      /* Eingabe */
   struct ft_transftam *ftamext;      /* Eingabe */
   char   *locccsn;                   /* Eingabe */
   char   *remccsn;                   /* Eingabe */
   enum   ft_tabexp tabexp;           /* Eingabe */
   char   *fud;                       /* Eingabe */
   int    fudlen;                     /* Eingabe */
   enum   ft_rform rform;             /* Eingabe */
};
```

*Hinweis*

Für bestimmte Parameter gelten Defaultwerte (s.u.), wenn Sie diese Parameterliste mit den Angaben für den Übertragungsauftrag mit binär 0 initialisieren.

Sie initialisieren die Parameterliste mit binär 0 z.B. mit dem Befehl:

```
memset (transpar, '\0', sizeof(struct ft_transpar));
```

Die Felder der Struktur *ft_transpar* haben folgende Bedeutung:

ftparvers
: Version der Datenstruktur.
*ftparvers* muss mit dem Wert `FT_TPARV1` oder `FT_TPARV2` versorgt werden.

direction
: Richtung der Dateiübertragung:

  `FT_SEND`
  : Datei ins ferne System senden

  `FT_RECEIVE`
  : Datei aus dem fernen System holen. Beim Holen einer Datei können Sie keine Wildcards verwenden.

synchron
: gibt an, wie die Datei übertragen werden soll:

  `FT_ASYNC`
  : asynchrone Übertragung (Defaultwert nach Initialisierung mit binär 0)

  `FT_SYNC`
  : synchrone Übertragung

locfn
: Dateiname im lokalen System oder Vor-/Nachverarbeitungskommdo.

  Beim lokalen Dateinamen kann jetzt auch ein Vorverarbeitungskommando (beim Senden von Daten) oder ein Nachverarbeitungskommando (beim Empfangen von Daten) angegeben werden.

  Absolute und relative Pfadangaben sind erlaubt. Relative Pfadangaben beziehen sich auf das Dateiverzeichnis, in dem das Programm gestartet wird.

remfn
: Dateiname im fernen System oder Vor-/Nachverarbeitungskommando.

  Beim fernen Dateinamen kann auch ein Vorverarbeitungskommando (beim Empfangen von Daten) oder ein Nachverarbeitungskommando (beim Senden von Daten) angegeben werden.

  Absolute und relative Pfadangaben sind erlaubt. Relative Pfadangaben beziehen sich auf die im Berechtigungsprofil festgelegte Benutzerkennung, wenn die FTAC-Funktion eingesetzt wird, sonst auf das HOME-Verzeichnis, siehe Seite 12. Daneben kann, wie beim Kommando *ncopy*, anstelle eines Dateinamens ein Vorverarbeitungskommando mit vorangestelltem Pipe-Zeichen (|) verwendet werden (siehe auch Kommandobeschreibung zu *ncopy*).

filetype

Dateityp im lokalen System:

FT_NOTYPE
: keine Festlegung des Dateityps. Es gelten die Standardwerte (siehe openFT-Benutzerhandbuch, Kommando *ft*). (Defaultwert nach Initialisierung mit binär 0)

FT_TEXT
: Die Datei enthält Text mit variablen Satzlängen. Sätze sind in Windows durch CRLF (X'0D0A') und in Unix-Systemen durch das Zeichen Zeilenvorschub \n abgeschlossen.

FT_USER
: Die Datei enthält vom Benutzer strukturierte Binärdaten mit variabler Satzlänge. Jeder Satz beginnt mit zwei Bytes, die die Längenangabe des Satzes enthalten.

FT_BINARY
: Die Datei enthält eine unstrukturierte Folge von Binärdaten.

writemode

gibt an, ob die Zieldatei neu erzeugt, überschrieben oder erweitert werden soll:

FT_NOMODE
: keine Festlegung der Schreibregel. Es gelten die Standardwerte (siehe openFT-Benutzerhandbuch, Kommando *ft*). (Defaultwert nach Initialisierung mit binär 0)

FT_OVERWR
: Eine bereits vorhandene Zieldatei wird überschrieben. War die Zieldatei noch nicht vorhanden, wird sie neu eingerichtet.

FT_EXTEND
: Die übertragene Datei wird an das Ende einer bereits vorhandenen Zieldatei angehängt. War die Zieldatei noch nicht vorhanden, wird sie neu eingerichtet.

FT_NEW
: Die Zieldatei wird neu erzeugt und beschrieben. Ist die Zieldatei bereits vorhanden, wird der Auftrag abgelehnt.

compress

gibt an, ob komprimiert übertragen werden soll:

FT_NOCOMPR
: keine Komprimierung (Defaultwert nach Initialisierung mit binär 0)

FT_COMPRESS
: Mehrere gleiche, aufeinanderfolgende Zeichen werden in komprimierter Form übertragen (Byte-Komprimierung).

FT_COMPRESSZIP
: Zip-Komprimierung. Bei Kopplung zu Partnern, die diese Komprimierung nicht unterstützen wird automatisch auf Byte-Komprimierung oder keine Komprimierung umgeschaltet. Die Zip-Kompression steht nur zur Verfügung, wenn *ftparvers* auf den Wert FT_TPARV2 gesetzt wird und beim Aufruf von *ft_transfer* der Parameter *options* angegeben ist.

filepasswd
: Kennwort der Datei im fernen System, falls sie mit einem Kennwort geschützt ist.

locsuccproc
: Kommando, das im lokalen System im Anschluss an eine erfolgreiche asynchrone Dateiübertragung ausgeführt wird.

  Bei synchronen Übertragungsaufträgen darf *locsuccproc* nicht angegeben werden.

  Innerhalb des Kommandos bzw. der Kommandofolge für die Folgeverarbeitung können Variablen angegeben werden. Weitere Informationen siehe Abschnitt „Kommandos für Folgeverarbeitung“ auf Seite 66.

locfailproc
: Kommando, das im lokalen System ausgeführt wird, wenn eine asynchrone Dateiübertragung durch einen Fehler abgebrochen wurde.

  Bei synchronen Übertragungsaufträgen darf *locfailproc* nicht angegeben werden.

  Innerhalb des Kommandos bzw. der Kommandofolge für die Folgeverarbeitung können Variablen angegeben werden.Weitere Informationen siehe Abschnitt „Kommandos für Folgeverarbeitung“ auf Seite 66.

remsuccproc
: Kommando, das im fernen System im Anschluss an eine erfolgreiche Dateiübertragung ausgeführt wird.

  Einige Partnersysteme (z.B. openFT für BS2000/OSD) unterstützen sogar Folgen aus mehreren Kommandos. Im Anschluss an eine erfolgreiche Übertragung werden diese Kommandos im fernen System unter dem angegebenen login ausgeführt.

  Innerhalb des Kommandos bzw. der Kommandofolge für die Folgeverarbeitung können Variablen angegeben werden. Weitere Informationen siehe Abschnitt „Kommandos für Folgeverarbeitung“ auf Seite 66.

remfailproc

Kommando, das im fernen System ausgeführt wird, wenn eine Dateiübertragung durch einen Fehler abgebrochen wurde.

Einige Partnersysteme (z.B. openFT für BS2000/OSD) unterstützen sogar Folgen aus mehreren Kommandos. Diese Kommandos werden im fernen System unter dem angegebenen login ausgeführt, wenn eine bereits begonnene Dateiübertragung durch einen Fehler abgebrochen wurde.

Innerhalb des Kommandos bzw. der Kommandofolge für die Folgeverarbeitung können Variablen angegeben werden. Weitere Informationen siehe unten bei „Kommandos für Folgeverarbeitung“.

**Kommandos für Folgeverarbeitung**

– Die Angaben für die lokale Folgeverarbeitung, also für *locsuccproc* und *locfailproc* zusammen, dürfen nicht mehr als 1000 Zeichen betragen.

– Die Angaben für die ferne Folgeverarbeitung, also für *remsuccproc* und *remfailproc* zusammen, dürfen nicht mehr als 1000 Zeichen betragen.

– Variable werden bei Start der Folgeverarbeitung im lokalen bzw. im fernen System durch Werte ersetzt, die sich aus der Funktion *ft_transfer()* ergeben.

  Als Variablen stehen Ihnen %FILENAME für Dateiname, %PARTNER für den Partnernamen, %RESULT für das Ergebnis des Auftrags und %RID für die Request-Id zur Verfügung.

  %RID ist nur für lokale Folgeverarbeitung erlaubt.

  Nach dem Start der Folgeverarbeitung werden die Variablen im jeweiligen System ersetzt. Anschließend werden die Kommandos der Folgeverarbeitung ausgeführt.
  Folgende Variablenersetzungen sind möglich:

  – %FILENAME
    durch den Dateinamen, wie er im Auftrag für das entsprechende System angegeben wurde

  – %PARTNER
    bei lokaler Folgeverarbeitung durch den beim Aufruf angegebenen Partnernamen. Bei Folgeverarbeitung im fernen System wird %PARTNER durch den Namen des Auftraggeber-Systems ersetzt, d.h. durch den Namen, mit dem es im Partnersystem bekannt ist.

- %RESULT
  durch die Meldungsnummer des Auftrags bezogen auf das jeweilige System. So erhält %RESULT z.B. bei erfolgreicher Ausführung eines Auftrags die Meldungsnummer 0 (Wird für *options* der Wert NULL angegeben, dann ist das Meldungsverhalten der Programmschnittstelle kompatibel zur Programmschnittstelle von openFT < V10).
- %RID
  durch die Request-Id des Auftrags im lokalen System.

Ist das Partnersystem ein openFT für BS2000/OSD, dann können Sie auch die Variablen %ELEMNAME, %ELEMVERS und %ELEMTYP verwenden.

- In Windows stehen bei der Folgeverarbeitung im lokalen System nur die System-Umgebungsvariablen zur Verfügung.
- Bei der Folgeverarbeitung im lokalen Unix-System und bei Folgeverarbeitung in einem fernen Unix-System wird **nicht** die in der Datei *.profile* abgelegte Kommandofolge durchlaufen. Ihnen stehen nur die Standardwerte der Shell-Variablen HOME, LOGNAME, PATH und USER zur Verfügung sowie die von *root* gesetzten Werte der Shell-Variablen LANG und TZ.
- Denken Sie daran, bei der Angabe von BS2000-Kommandos am Anfang des Kommandos den Schrägstrich (/) mit anzugeben.
- Bei Aufträgen mit FTAM- und FTP-Partnern steht nur die Funktion „lokale Folgeverarbeitung“ zur Verfügung. Wird im fernen System die FTAC-Funktion eingesetzt, dann kann diese Einschränkung umgangen werden. In dem Fall kann im fernen System ein Berechtigungsprofil erzeugt werden, in dem eine Folgeverarbeitung definiert ist.

maxrecsize
: maximal zulässige Satzlänge bei Dateien vom Typ „Textdatei“ und „strukturierte Binärdatei“. Damit können auch Sätze übertragen und abgespeichert werden, die größer als der Standardwert sind. Sie müssen jedoch berücksichtigen, dass nicht alle Satzlängen in jedem beliebigen Partnersystem bearbeitet werden können.

  Der Maximalwert darf die in der Struktur *ft_prop* im Feld *maxrecord* angegebene Länge nicht überschreiten (siehe Abschnitt „ft_properties - Eigenschaften der Programmschnittstelle ermitteln“ auf Seite 31).

  Wenn Sie den Dateityp „binär“ gewählt haben, ist *maxrecsize* der Wert für alle Sätze der Sendedatei.

Bei FTAM-Partnern wird die Angabe zur maximalen Satzlänge nur wirksam, wenn für *filetype* der Dateityp explizit mit FT_TEXT, FT_USER oder FT_BINARY angegeben wird.

cantime
: Zeitpunkt, zu dem ein Übertragungsauftrag abgebrochen werden soll. Dieser Zeitpunkt muss im internen Zeitformat (Sekunden seit dem 1.1.1970 00:00:00) angegeben werden.

  Der Wert 0 bedeutet, dass kein zeitgesteuerter Abbruch erfolgt.

  Bei synchronen Aufträgen wird *cantime* ignoriert.

starttime
: Zeitpunkt, zu dem ein Übertragungsauftrag frühestens gestartet werden soll. Dieser Zeitpunkt muss im internen Zeitformat (Sekunden seit dem 1.1.1970 00:00:00) angegeben werden.

  Der Wert 0 bedeutet, dass die Übertragung so bald wie möglich gestartet wird.

  Bei synchronen Aufträgen wird *starttime* ignoriert.

priority
: gibt die Priorität des Auftrags an:

  FT_PRIONORM
  : normale Priorität
    (Defaultwert nach Initialisierung mit binär 0)

  FT_PRIOLOW
  : niedrige Priorität (wird bei synchronen Aufträgen ignoriert)

transparent
: gibt an, ob die Dateiübertragung transparent sein soll:

  FT_NOTRANSPAR
  : normale Übertragung (Defaultwert nach Initialisierung mit binär 0)

  FT_TRANSPARENT
  : transparente Übertragung

encryption
: gibt an, ob die Benutzerdaten verschlüsselt werden sollen bzw. ob eine Datenintegritätsprüfung durchgeführt werden soll:

  FT_NOENCRYPT
  : Benutzerdaten werden nicht verschlüsselt und eine Datenintegritätsprüfung wird nicht durchgeführt (Defaultwert nach Initialisierung mit binär 0)

FT_ENCRYPT
: Benutzerdaten werden verschlüsselt und die Datenintegrität wird automatisch geprüft.
Dazu muss openFT-CR installiert sein.

FT_ONLYDICHECK
: Die Datenintegritätsprüfung des übertragenen Dateiinhalts wird durchgeführt. Die Datenintegritätsprüfung steht nur zur Verfügung, wenn *ftparvers* auf den Wert FT_TPARV2 gesetzt wird und beim Aufruf von *ft_transfer* der Parameter *options* angegeben ist.

ftamext
: FTAM-spezifische Parameter, die Sie mit der Struktur *ft_transftam* bekanntgeben (siehe auch bei den Kommandos *ft* und *ncopy*, Optionen *-av*, *-ac*, *-am*, *-lq* und *-cp*):

```
struct ft_transftam
{
   enum ft_available available;     /* Eingabe */
   char  *account;                  /* Eingabe */
   int   accessmode;                /* Eingabe */
   char  *legalq;                   /* Eingabe */
   char  *crpasswd;                 /* Eingabe */
};
```

Die Felder der Struktur *ft_transftam* haben folgende Bedeutung:

available
: legt die Verfügbarkeit der Zieldatei fest:

FT_NOAVAIL:
: keine Festlegung der Verfügbarkeit (Defaultwert nach Initialisierung mit binär 0)

FT_AVAILIMM:
: Die Zieldatei erhält das Attribut „sofort verfügbar“.

FT_AVAILNIMM:
: Die Zieldatei erhält das Attribut „nicht sofort verfügbar“.

account
: Abrechnungskonto beim FTAM-Partner

accessmode
: legt die Zugriffsrechte der Zieldatei fest. Die Zugriffsrechte entstehen durch logisches Odern folgender Einzelrechte:

  `FT_ACCR`:
  : Die Datei darf gelesen werden.

  `FT_ACCI`:
  : Dateneinheiten dürfen in die Datei eingefügt werden.

  `FT_ACCP`:
  : Die Datei darf überschrieben werden.

  `FT_ACCX`:
  : Die Datei darf erweitert werden, d.h. Daten können an die Datei angefügt werden.

  `FT_ACCE`:
  : Dateneinheiten dürfen aus der Datei gelöscht werden.

  `FT_ACCA`:
  : Dateiattribute dürfen gelesen werden.

  `FT_ACCC`:
  : Dateiattribute dürfen geändert werden.

  `FT_ACCD`:
  : Die Datei darf gelöscht werden.

legalq
: legt die rechtliche Bestimmung (Copyright) für die Zieldatei fest.

crpasswd
: Kennwort, das Sie im fernen System benötigen, um eine Datei zu erzeugen.

locccsn
: gibt den Namen der Codierung an (CCS-Name), mit der die lokale Datei gelesen oder geschrieben wird. *CCS-Name* muss im lokalen System bekannt sein.

  Wird keine Codierung angegeben, wird der bei openFT per Betriebsparameter eingestellte Standardwert für die Codierung verwendet.

  Die Unterstützung der „Coded Character Sets“ (CCS) steht nur dann zur Verfügung, wenn *ftparvers* auf den Wert `FT_TPARV2` gesetzt wird und beim Aufruf von *ft_transfer* der Parameter *options* angegeben ist.

remccsn

gibt den Namen der Codierung an (CCS-Name), mit der die ferne Datei gelesen oder geschrieben wird. *CCS-Name* muss im fernen System bekannt sein. Wird keine Codierung angegeben, wird der durch XHCS (BS2000/OSD) bzw. per openFT-Betriebsparameter (andere Plattformen) eingestellte Zeichensatz für die Codierung verwendet.

Die Unterstützung der „Coded Character Sets“ (CCS) wird nur für das openFT-Protokoll und für Partner mit openFT ab V10.0 unterstützt und steht nur dann zur Verfügung, wenn *ftparvers* auf den Wert `FT_TPARV2` gesetzt wird und beim Aufruf von *ft_transfer* der Parameter *options* angegeben ist.

tabexp

gibt an, ob für einen Outbound-Sendeauftrag die Tabulator-Expansion und die Umwandlung leerer Zeilen in Zeilen mit einem Zeichen für nicht-FTAM-Partner durchgeführt werden soll. Die Tabulator-Expansion steht nur zur Verfügung, wenn *ftparvers* auf den Wert `FT_TPARV2` gesetzt wird und beim Aufruf von *ft_transfer* der Parameter *options* angegeben ist.

`FT_TABAUTO`

Tabulator-Expansion und Umwandlung der Leerzeilen sind eingeschaltet, wenn eine Datei zu einem BS2000-, OS/390- oder z/OS-System gesendet wird (Standardwert nach Initialisierung mit binär 0).

`FT_TABON`

Tabulator-Expansion und Umwandlung der Leerzeilen sind eingeschaltet.

`FT_TABOFF`

Tabulator-Expansion und Umwandlung der Leerzeilen sind ausgeschaltet.

fud

Adresse eines Datenbereichs für die sogenannten „Further Details“, die im Fehlerfall eine genauere Fehlerursache angeben können.
Bei Angabe von NULL wird keine weiterführende Fehlerursache ausgegeben. Der Parameter *fud* steht nur dann zur Verfügung, wenn *ftparvers* auf den Wert `FT_TPARV2` gesetzt wird und beim Aufruf von *ft_transfer* der Parameter *options* angegeben ist.

fudlen

Länge des Datenbereichs von *fud*.
Der Parameter *fudlen* steht nur dann zur Verfügung, wenn *ftparvers* auf den Wert `FT_TPARV2` gesetzt wird und beim Aufruf von *ft_transfer* der Parameter *options* angegeben ist.

rform

gibt das Satzformat der zu übertragenden Datei an. Der Parameter *rform* steht nur dann zur Verfügung, wenn *ftparvers* auf den Wert `FT_TPARV2` gesetzt wird und beim Aufruf von *ft_transfer* der Parameter *options* angegeben ist.

`FT_NOFORM`
: Das Satzformat der zu übertragenden Datei ist unbekannt (Standardwert nach Initialisierung mit binär 0).

`FT_VARIABLE`
: Die zu übertragende Datei enthält Sätze mit variabler Satzlänge.

`FT_FIXED`
: Die zu übertragende Datei enthält Sätze mit einheitlich fester Satzlänge.

`FT_UNDEF`
: Die zu übertragende Datei enthält Sätze mit undefinierter Satzlänge.

errorinfo

Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17).
Die Angabe des Parameters ist optional. Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert NULL angeben.

options

Die Angabe des Parameters *options* ist optional. Wird der Wert NULL angegeben, dann ist das Meldungsverhalten der Programmschnittstelle kompatibel zur Programmschnittstelle von openFT < V10.
Alternativ können durch die Angabe der Struktur *ft_options* (siehe Abschnitt „ft_options“ auf Seite 18) das openFT-Meldungsnummernschema ab openFT V10 und die Funktionserweiterungen aktiviert werden.

**Rückgabewert**

| | |
|---|---|
| *n* | Bei erfolgreichen asynchronen Aufträgen: Request-Id ($n \neq 0$) |
| 1 | Bei erfolgreichen synchronen Aufträgen |
| 0 | Fehler. Die Dateiübertragung wurde nicht angestoßen.<br>Die Fehlerart wird in *errorinfo* hinterlegt. |

## 3.16 ft_xcopen - Kommando im fernen System ausführen

*ft_xcopen()* führt das Kommando im fernen System synchron aus.

### Syntax

```
#include <ftapi.h>

void *ft_xcopen(const struct ft_admission *admis,      /* Eingabe */
                struct ft_xcpar *par,
                struct ft_err *errorinfo,
                void *options);                        /* Eingabe */
```

### Parameter

admis Angaben für das ferne System (siehe Abschnitt „ft_admission“ auf Seite 15).

par Angaben für den Auftrag, die Sie mit der Struktur *ft_xcpar* bekanntgeben:

```
struct ft_xcpar
{
   int xcparvers;                    /* Eingabe */
   char *cmd;                        /* Eingabe */
   enum ft_filetype type;            /* Eingabe */
   enum ft_encrypt encryption;       /* Eingabe */
   char *locccsn;                    /* Eingabe */
   char *remccsn;                    /* Eingabe */
   int retcode;                      /* Ausgabe */
   long long outlen;                 /* Ausgabe */
   long long errlen;                 /* Ausgabe */
   char *fud;                        /* Eingabe */
   int fudlen;                       /* Eingabe */
};
```

Die Felder der Struktur *ft_xcpar* haben folgende Bedeutung:

xcparvers
: Version der Datenstruktur;
*xcparvers* muss mit dem Wert `FT_XCPARV1` versorgt werden.

cmd
: Das auf dem Partnersystem auszuführende Kommando. Die maximale Länge darf die in der Struktur *ft_prop* im Feld *maxcmdlen* angegebene Länge nicht überschreiten (siehe Abschnitt „ft_properties - Eigenschaften der Programmschnittstelle ermitteln“ auf Seite 31).

type Datentyp der übertragenen Nutzdaten (in *stdout*). Folgende Werte sind zulässig:

`FT_TEXT`
: gibt das Übertragungsformat als Text an. Die Tabulator-Expansion ist ausgeschaltet (Defaultwert, wenn bei *locccsn* und/oder *remccsn* ein CCS-Name angegeben wird).

`FT_BINARY`
: gibt das Übertragungsformat als binär ohne Konvertierungen an. (Defaultwert, wenn bei *locccsn* und *remccsn* kein CCS-Name angegeben wird).

encryption
: gibt an, ob die Benutzerdaten verschlüsselt werden sollen. Folgende Werte sind zulässig:

`FT_NOENCRYPT`
: Benutzerdaten werden nicht verschlüsselt (Defaultwert nach Initialisierung mit binär 0).

`FT_ENCRYPT`
: Benutzerdaten werden verschlüsselt. Dazu muss openFT-CR installiert sein. Kann das Partnersystem nicht mit Verschlüsselung arbeiten, wird der Auftrag abgelehnt.

locccsn
: gibt den Namen der Codierung an (CCS-Name), mit der die Daten der Standardausgabe geschrieben werden sollen. *CCS-Name* muss im lokalen System bekannt sein.

  Wird keine Codierung angegeben, wird der bei openFT per Betriebsparameter eingestellte Standardwert für die Codierung verwendet. Der Parameter *locccsn* darf nicht mit `FT_BINARY` kombiniert werden

remccsn
: gibt den Namen der fernen Codierung an (CCS-NAME), mit der die Daten der Standardausgabe des fernen Kommandos gelesen werden. *CCS-Name* muss im fernen System bekannt sein.

  Wird keine Codierung angegeben, wird der durch XHCS (BS2000/OSD) bzw. per openFT-Betriebsparameter (andere Plattformen) eingestellte Zeichensatz für die Codierung verwendet. Der Parameter *remccsn* darf nicht mit `FT_BINARY` kombiniert werden.

Wird nur für das openFT-Protokoll und für Partner mit openFT ab V10.0 unterstützt. Beachten Sie bitte, dass nicht jedes Partnersystem alle im lokalen System möglichen Zeichensätze unterstützt.

retcode
: Returncode der fernen Kommandoausführung

outlen
: Anzahl der Datenbytes für *stdout*, die gelesen wurden.

errlen
: Anzahl der Datenbytes für *stderr*, die gelesen wurden.

fud
: Adresse eines Datenbereichs für die sogenannten "Further Details", die im Fehlerfall eine genauere Fehlerursache angeben können. Bei Angabe von `NULL` wird keine weiterführende Fehlerursache ausgegeben.

fudlen
: Länge des Datenbereichs von *fud*.

errorinfo
: Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17).
Die Angabe des Parameters ist optional. Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert NULL angeben.

options
: *options* muss mit dem Wert `FT_APIV3` versorgt werden. Die Angabe des Parameters ist obligatorisch. Der Aufbau der Struktur *ft_options* ist im Abschnitt „Version der Programmschnittstelle“ auf Seite 18 beschrieben.

**Rückgabewert**

id
: Id des Aufrufs. Diese muss bei *ft_sdinfo()* und *ft_sdclose()* angegeben werden.

NULL
: Fehler. Die Fehlerart wird in *errorinfo* hinterlegt.
Der Aufruf von *ft_sdclose()* ist im Fehlerfall nicht notwendig.

## 3.17 ft_xcinfo - Vom Kommando erzeugte Daten auslesen

*ft_xcinfo()* liest die Ausgabedaten des mit *ft_xcopen()* im fernen System ausgeführten Kommandos aus.

*ft_xcinfo* kann mehrfach für jeden Ausgabekanal (*stdout*, *stderr*) aufgerufen werden. Dabei werden jeweils die nächsten, noch nicht gelesenen Daten in den Puffer *buf* geschrieben.

### Syntax

```
#include <ftapi.h>

int ft_xcinfo(void *id,                          /* Eingabe */
              struct ft_xcipar *par,
              int buflen,                        /* Eingabe */
              char *buf,
              struct ft_err *errorinfo);
```

### Parameter

id Id des Aufrufs (Rückgabewert von *ft_xcopen*)

par Auswahl des Ausgabekanals, die Sie mit der Struktur *ft_xcipar* bekanntgeben:

```
struct ft_xcipar
{
   int xciparvers;                 /* Eingabe */
   enum ft_chn channel;            /* Eingabe */
   char *fud;                      /* Eingabe */
   int fudlen;                     /* Eingabe */
};
```

Die Felder der Struktur *ft_xcipar* haben folgende Bedeutung:

xciparvers
: Version der Datenstruktur; *xciparvers* muss mit dem Wert `FT_XCIPARV1` versorgt werden.

channel
: Auswahl des Kanals. Folgende Werte sind zulässig:

`FT_STDOUT`
: stdout-Kanal

`FT_STDERR`)
: stderr-Kanal

fud Adresse eines Datenbereichs für die sogenannten "Further Details", die im Fehlerfall eine genauere Fehlerursache angeben können. Bei Angabe von `NULL` wird keine weiterführende Fehlerursache ausgegeben.

fudlen Länge des Datenbereichs von *fud*.

buflen
Größe des Datenbereichs für die Ausgabedaten

buf Adresse des Datenbereichs für die Ausgabedaten

errorinfo
Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17).
Die Angabe des Parameters ist optional. Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert NULL angeben.

**Rückgabewert**

n Anzahl der in den Puffer *buf* geschriebenen Bytes.

0 Alle Daten wurden bereits gelesen, der Puffer ist leer.

-1 Fehler. Die Fehlerart wird in *errorinfo* hinterlegt.

## 3.18 ft_xcclose - Kommandoausführung beenden

*ft_xcclose()* beendet das Auslesen der Ausgabedaten des mit *ft_xcopen()* im fernen System ausgeführten Kommandos.

Diese Funktion muss nach erfolgreichem Aufruf von *ft_xcopen()* als letzter Schritt aufgerufen werden. *ft_xcclose()* gibt nicht mehr benötigte Ressourcen frei. Anschließend können Sie sich nicht mehr auf diese Id beziehen.

### Syntax

```
#include <ftapi.h>

int ft_xcclose(void *id,                              /* Eingabe */
               struct ft_err *errorinfo);
```

### Parameter

id Id des Aufrufs (Rückgabewert von *ft_xcopen*)

errorinfo
Bereich, in dem genauere Informationen hinterlegt sind, wenn ein Fehler aufgetreten ist (siehe Abschnitt „ft_err“ auf Seite 17).
Die Angabe des Parameters ist optional. Wenn Sie keine genaueren Fehlerinformationen benötigen, können Sie für *errorinfo* den Wert NULL angeben.

### Rückgabewert

0 Kein Fehler.

-1 Fehler. Die Fehlerart wird in *errorinfo* hinterlegt.

# 4 Fehlercodes

Fehlermeldungen, die in die Struktur *ft_err* eingetragen werden (siehe Abschnitt „ft_err“ auf Seite 17), bestehen aus folgenden Feldern:

– main (Fehlerklasse)
– detail (Fehler)
– additional (zusätzliche Fehlerinformation)

Die Fehlermeldungen sind in der folgenden Auflistung nach Fehlerklassen sortiert. Es gibt die Fehlerklassen:

| | |
|---|---|
| FTEM_INT | Interne Fehler |
| FTEM_PAR | Parameterfehler |
| FTEM_LOCERR | Ablauffehler im lokalen System |
| FTEM_CONNERR | Ablauffehler in der Verbindung zum Partner |
| FTEM_REMERR | Ablauffehler im fernen System |

In der folgenden Aufstellung sind Fehlermeldungen der Fehlerklasse Parameterfehler den Funktionsaufrufen zugeordnet, soweit sie nicht allgemein gelten.

## 4.1 Interne Fehler

| Fehler-klasse | Fehler | zusätzliche Fehler-information | Bedeutung |
|---|---|---|---|
| FTEM_INT | FTED_MEM | 0 | Fehler bei der Speicheranforderung |
| FTEM_INT | FTED_CRFILE | 0 | Fehler beim Erzeugen einer Datei |
| FTEM_INT | FTED_INIT | 0 | Der Server kann nicht initialisiert werden |
| FTEM_INT | FTED_SIGNAL | *signal* | Befehl wurde durch *signal* unterbrochen. *signal* bezeichnet das Signal, das die Unterbrechung verursachte. |
| FTEM_INT | *funktion* | *errno* | Fehler bei einem Systemaufruf.<br>*funktion* bezeichnet den fehlerhaften Systemaufruf:<br>FTED_FORK fork<br>FTED_OPEN open<br>FTED_OPENDIR opendir<br>FTED_PIPE pipe<br>FTED_READ read<br>FTED_RMFILE rmfile<br>FTED_STAT stat<br>FTED_SYSTEM system<br>FTED_WRITE write<br>*errno* ist der Wert der errno-Variablen, der durch den fehlerhaften Systemaufruf gesetzt wird. Wenn nicht alle Bytes geschrieben werden konnten, hat *errno* den Wert −1. |
| FTEM_INT | FTED_INTERNAL | *reason* | Sonstiger interner Fehler.<br>*reason* bezeichnet die Ursache des Fehlers, wenn diese bekannt ist:<br>FTEA_FN vom Server nicht unterstützte Funktion<br>FTEA_VERS vom Server nicht unterstützte Version der Datenstruktur |

## 4.2 Parameterfehler

### Allgemeine Fehler

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_PAR | FTED_INVSESS | 0 | Die Sitzungsnummer (session) ist ungültig. |
| FTEM_PAR | FTED_LEN | *parameter* | *parameter* bezeichnet den Parameter, der zu lang ist:<br>FTEA_REMACC remaccount<br>FTEA_REMADM remadmis<br>FTEA_REMPWD rempasswd<br>FTEA_REMSYS remsys |
| FTEM_PAR | FTED_MAND | FTEA_REMSYS | Das ferne System wurde nicht angegeben. |
| FTEM_PAR | FTED_VALUE | FTEA_APIVERS | Die in den *ft_options* angegebene API-Version ist ungültig. |
| FTEM_PAR | FTED_MAND | *parameter* | *parameter* bezeichnet den fehlenden Pflichtparameter:<br>FTEA_REMSYS remsys oder admis wurde nicht angegeben.<br>FTEA_OPTIONS options wurde nicht angegeben. |
| FTEM_PAR | FTED_VALUE | FTEA_RID | Die Request-Id (rid) ist ungültig. |
| FTEM_PAR | FTED_VERS | 0 | Die Version der Datenstruktur (Parameterliste oder Ausgabebereich) ist ungültig. |

### Fehler bei ft_cancel

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_PAR | FTED_TERM | 0 | Der Auftrag ist schon beendet. |

### Fehler bei ft_credir

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_PAR | FTED_LEN | *parameter* | *parameter* bezeichnet den Parameter, der zu lang ist:<br>FTEA_FPWD mgmtpasswd<br>FTEA_REMFN dn |
| FTEM_PAR | FTED_MAND | 0 | Die Parameterliste *par* wurde nicht angegeben. |
| FTEM_PAR | FTED_REMOTE | FTEA_NOACCESS | Keine Berechtigung zum Erzeugen im fernen System. |
| FTEM_PAR | FTED_REMOTE | FTEA_EXIST | Verzeichnis existiert bereits im fernen System. |
| FTEM_PAR | FTED_VALUE | *parameter* | *parameter* bezeichnet den Parameter, der ungültig ist:<br>0 unbekannter Parameter/ Parameter sind inkompatibel.<br>FTEA_FPWD mgmtpasswd<br>FTEA_REMACC remaccount<br>FTEA_REMADM remadmis<br>FTEA_REMFN remfn<br>FTEA_REMPWD rempasswd<br>FTEA_REMSYS remsys |

### Fehler bei ft_delete

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_PAR | FTED_LEN | *parameter* | *parameter* bezeichnet den Parameter, der zu lang ist:<br>FTEA_FPWD mgmtpasswd<br>FTEA_REMFN fn |
| FTEM_PAR | FTED_MAND | 0 | Die Parameterliste *par* wurde nicht angegeben. |
| FTEM_PAR | FTED_REMOTE | FTEA_NOACCESS | Keine Berechtigung zum Löschen im fernen System |
| FTEM_PAR | FTED_REMOTE | FTEA_NOTEMPTY | Das Verzeichnis im fernen System ist nicht leer. |
| FTEM_PAR | FTED_REMOTE | FTEA_NOTEXIST | Datei/Verzeichnis existiert nicht im fernen System. |
| FTEM_PAR | FTED_VALUE | *parameter* | *parameter* bezeichnet den Parameter, der ungültig ist:<br>0 unbekannter Parameter/ Parameter sind inkompatibel.<br>FTEA_FPWD mgmtpasswd<br>FTEA_FTYPE filetype<br>FTEA_REMACC remaccount<br>FTEA_REMADM remadmis<br>FTEA_REMFN remfn<br>FTEA_REMPWD rempasswd<br>FTEA_REMSYS remsys |

### Fehler bei ft_open

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_PAR | FTED_DIRAC | *errno* | *errno* bezeichnet den Wert der errno-Variablen, der durch den stat()-Aufruf gesetzt wurde. Die errno-Variable hat den Wert 0, wenn für das Dateiverzeichnis kein Schreibrecht vergeben ist. |
| FTEM_PAR | FTED_LEN | 0 | Der Name des Arbeitsverzeichnisses (workdir) ist zu lang. |
| FTEM_PAR | FTED_MAND | 0 | Der Name des Arbeitsverzeichnisses (workdir) wurde nicht angegeben. |
| FTEM_PAR | FTED_NODIR | 0 | Der angegebene Name (workdir) bezeichnet kein Dateiverzeichnis. |
| FTEM_PAR | FTED_OPEN | 0 | In einem Programm wurde einer Sitzung bereits dasselbe Dateiverzeichnis (workdir) zugeordnet. |
| FTEM_PAR | FTED_VALUE | 0 | Der Name für das Arbeitsverzeichnis (workdir) ist ungültig. |

### Fehler bei ft_reqstat

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_PAR | FTED_MAND | 0 | Der Ausgabebereich *stat* wurde nicht angegeben. |

### Fehler bei ft_reqterm

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_PAR | FTED_NOTERM | 0 | Der Auftrag ist noch aktiv. |

### Fehler bei ft_show und ft_showdir

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_PAR | FTED_LEN | *parameter* | *parameter* bezeichnet den Parameter, der zu lang ist:<br>FTEA_FPWD mgmtpasswd<br>FTEA_REMFN fn |
| FTEM_PAR | FTED_MAND | 0 | Die Parameterliste *par* wurde nicht angegeben. / Der Ausgabebereich *info* wurde nicht angegeben (nur bei *ft_show()*). |
| FTEM_PAR | FTED_REMOTE | FTEA_NOACCESS | Keine Berechtigung zum Lesen der Attribute im fernen System |
| FTEM_PAR | FTED_REMOTE | FTEA_NOTEXIST | Datei/Verzeichnis existiert nicht im fernen System. |
| FTEM_PAR | FTED_VALUE | *parameter* | *parameter* bezeichnet den Parameter, der ungültig ist:<br>0 unbekannter Parameter/Parameter sind inkompatibel.<br>FTEA_FPWD mgmtpasswd<br>FTEA_REMACC remaccount<br>FTEA_REMADM remadmis<br>FTEA_REMFN remfn<br>FTEA_REMPWD rempasswd<br>FTEA_REMSYS remsys |

### Fehler bei ft_transfer

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_PAR | FTED_LEN | *parameter* | *parameter* bezeichnet den Parameter, der zu lang ist:<br>FTEA_ACCOUNT ftamext -> account<br>FTEA_CRPWD ftamext -> crpasswd<br>FTEA_FPWD filepasswd<br>FTEA_LEGALQ ftamext -> legalq<br>FTEA_LOCCCSN locccsn<br>FTEA_LOCFN locfn<br>FTEA_LOCPR Summe der Längen von locsuccproc und locfailproc<br>FTEA_REMCCSN remccsn<br>FTEA_REMFN remfn<br>FTEA_REMPR Summe der Längen von remsuccproc und remfailproc |
| FTEM_PAR | FTED_MAND | *parameter* | *parameter* bezeichnet den Parameter, der nicht angegeben wurde:<br>0 Die Parameterliste *par* wurde nicht angegeben.<br>FTEA_LOCFN locfn |

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_PAR | FTED_VALUE | *parameter* | *parameter* bezeichnet den Parameter, der ungültig ist:<br>FTEA_ACCESS ftamext -> accessmode<br>FTEA_AVAIL ftamext -> available<br>FTEA_CANTIME cantime<br>FTEA_COMPR compress<br>FTEA_DIR direction<br>FTEA_ENCRYPT encryption<br>FTEA_FTYPE filetype<br>FTEA_MAXREC maxrecsize<br>FTEA_PRIO priority<br>FTEA_REMADM remadm. Die Benutzerkennung/ Zugangsberechtigung im fernen System ist ungültig.<br>FTEA_REMFN remfn. Die angegebene Datei existiert nicht/der Zugriff ist nicht erlaubt.<br>FTEA_REMSYS remsys. Das angegebene ferne System ist unbekannt.<br>FTEA_SYNC synchron<br>FTEA_RFORM rform<br>FTEA_STARTTIME starttime<br>FTEA_TABEXP tabexp<br>FTEA_TRANSP transparent<br>FTEA_WMODE writemode |

### Fehler bei ft_sdopen

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_PAR | FTED_LEN | *parameter* | *parameter* bezeichnet den Parameter, der zu lang ist:<br>FTEA_TRANSP transparent<br>FTEA_REMFN fn |
| FTEM_PAR | FTED_MAND | 0 | 0 Die Parameterliste *par* wurde nicht angegeben. |
| FTEM_PAR | FTED_REMOTE | FTEA_NOACCESS | Keine Berechtigung zum Lesen der Attribute im fernen System |
| FTEM_PAR | FTED_REMOTE | FTEA_NOTEXIST | Datei/Verzeichnis existiert nicht im fernen System. |

### Fehler bei ft_sdinfo

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_PAR | FTED_MAND | FTEA_ID | Der Parameter *id* wurde nicht angegeben. |
| FTEM_PAR | FTED_VALUE | FTEA_BUFL | Der Puffer wurde nicht angegeben (buf=NULL oder bufsize<=0). |

### Fehler bei ft_xcopen

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_PAR | FTED_MAND | 0 | Die Parameterliste *par* wurde nicht angegeben. |
| FTEM_PAR | FTED_MAND | FTEA_CMD | Das Kommando *cmd* wurde nicht angegeben. |
| FTEM_PAR | FTED_LEN | FTEA_CMD | Das Kommando *cmd* ist zu lang. |

### Fehler bei ft_xcinfo

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_PAR | FTED_MAND | FTEA_ID | Der Parameter *id* wurde nicht angegeben. |
| FTEM_PAR | FTED_MAND | FTEA_BUFL | Der Puffer wurde nicht angegeben (buf=NULL oder bufsize<=0). |
| FTEM_PAR | FTED_LEN | FTEA_CHAN | Der Ausgabe-Kanal wurde nicht angegeben (FT_STDOUT oder FT_STDERR). |

## 4.3 Ablauffehler

### Allgemeine Fehler

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_PAR | FTED_FTMSG | *code* | *code* bezeichnet die Meldungsnummer des entsprechenden Kommandos (siehe openFT-Benutzerhandbuch). Der zugehörige Meldungstext kann auch mit dem Kommando *fthelp code* ermittelt werden. |

### Fehler im lokalen System

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_LOCERR | FTED_EXIST | 0 | Die lokale Datei existiert bereits. |
| FTEM_LOCERR | FTED_FTAC | 0 | Der Auftrag wurde vom lokalen FTAC abgewiesen. |
| FTEM_LOCERR | FTED_INCONS | 0 | Die lokale Datei ist inkonsistent. |
| FTEM_LOCERR | FTED_MEM | 0 | Die lokale Datei bekommt keinen Speicher. |
| FTEM_LOCERR | FTED_NOACCESS | 0 | Auf die lokale Datei kann nicht zugegriffen werden. |
| FTEM_LOCERR | FTED_NOCREAT | 0 | Die lokale Datei kann nicht angelegt werden. |
| FTEM_LOCERR | FTED_NOTEXIST | 0 | Die lokale Datei kann nicht gefunden werden. |

### Fehler in der Verbindung zum Partner

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_CONNERR | FTED_NOCONN | 0 | keine freie Transportverbindung |
| FTEM_CONNERR | FTED_NOTAVAIL | 0 | Das ferne System ist nicht verfügbar. |
| FTEM_CONNERR | FTED_UNKNOWN | 0 | Das ferne System ist unbekannt. |

### Fehler im fernen System

| Fehler-klasse | Fehler | zusätzliche Fehlerinformation | Bedeutung |
|---|---|---|---|
| FTEM_REMERR | FTED_EXIST | 0 | Die ferne Datei existiert bereits. |
| FTEM_REMERR | FTED_INCONS | 0 | Die ferne Datei ist inkonsistent. |
| FTEM_REMERR | FTED_MEM | 0 | Die ferne Datei bekommt keinen Speicher. |
| FTEM_REMERR | FTED_NOACCESS | 0 | Auf die ferne Datei kann nicht zugegriffen werden. |
| FTEM_REMERR | FTED_NOCREAT | 0 | Die ferne Datei kann nicht angelegt werden. |
| FTEM_REMERR | FTED_NOTEXIST | 0 | Die ferne Datei kann nicht gefunden werden. |
| FTEM_REMERR | FTED_REMADM | 0 | Die ferne Zugangsberechtigung ist ungültig. |

# 5 Beispielprogramme

Die Beispielprogramme, die mit openFT ausgeliefert werden, zeigen Ihnen verschiedene Möglichkeiten, die Programmschnittstelle einzusetzen. Die Quellcodes dieser Programme stehen im folgenden Unterverzeichnis des openFT-Installationsverzeichnisses zur Verfügung:

– in Windows: *openFT\samples\ftapi*
– in Unix-Systemen: */opt/openFT/samples*

Die Zugangsberechtigung zum Partnersystem muss bei den Beispielprogrammen eine FTAC-Berechtigung sein, d.h. die Angabe einer Benutzerkennung mit Passwort wird von den Beispielprogrammen nicht unterstützt.

## Beispiel 1: Asynchrone Übertragung einer Datei

Das Programm *sample1* wird folgendermaßen aufgerufen:

`sample1` *datei1 datei2*

Name und Zugangsberechtigung für das ferne System werden dann im Dialog erfragt. Damit das Programm ablaufen kann, muss folgendes Verzeichnis vorhanden sein:

– in Windows: das Arbeitsverzeichnis `%TMP%\ft`
– in Unix-Systemen: das Arbeitsverzeichnis `$HOME/ft`

Das Programm überträgt die Datei *datei1* asynchron aus dem lokalen System ins ferne System und speichert sie dort unter dem Namen *datei2* im HOME-Verzeichnis des Benutzers bzw. in der im Berechtigungsprofil festgelegten Benutzerkennung ab. Voraussetzung dafür ist, dass sich die zu sendende Datei *datei1* in demselben Verzeichnis befindet, in dem das Programm aufgerufen wird. Wird vom Benutzer z.B. in Windows durch Eingabe von STRG+C ein SIGINT-Signal erzeugt, solange die Datei noch nicht übertragen ist, wird der Übertragungsauftrag abgebrochen.

Das Programm ist folgendermaßen aufgebaut:

- Weil die Datei asynchron übertragen werden soll, wird zunächst mit der Funktion *ft_open()* eine Sitzung eröffnet, wobei als Arbeitsverzeichnis `%TMP%\ft` (in Windows) bzw. `$HOME/ft` (in Unix-Systemen) fest zugeordnet wird.
- *ft_open()* liefert eine Sitzungsnummer zurück, die die Sitzung kennzeichnet und bei weiteren Funktionsaufrufen angegeben werden muss.
- Die asynchrone Dateiübertragung wird mit der Funktion *ft_transfer()* eingeleitet, die die Request-Id des Auftrags zurückliefert.
- Fortwährend fragt das Programm ab, ob vom Benutzer ein SIGINT-Signal erzeugt wurde. Ist dies der Fall, so wird der Auftrag mit der Funktion *ft_cancel()* abgebrochen.
- Solange die Dateiübertragung nicht beendet ist oder nicht abgebrochen wurde, fragt die Funktion *ft_reqstat()* den Status des Übertragungsauftrags ständig ab.
- Wurde die Dateiübertragung beendet oder abgebrochen, wird der Auftrag mit der Funktion *ft_reqterm()* als beendet gekennzeichnet und die Sitzung mit der Funktion *ft_close()* geschlossen.

## Beispiel 2: Mehrere Dateiübertragungsaufträge mit Folgeverarbeitung

Das Programm *sample2* wird folgendermaßen aufgerufen:

`sample2` *datei1* [*datei2*] [*datei3*] [*datei4*]

Name und Zugangsberechtigung zum fernen System werden dann im Dialog erfragt. Damit das Programm ablaufen kann, muss das folgende Verzeichnis vorhanden sein:

– in Windows: das Arbeitsverzeichnis `%TMP%\ft`
– in Unix-Systemen: das Arbeitsverzeichnis `$HOME/ft`

Das Programm holt jede der angegebenen Dateien asynchron aus dem HOME-Verzeichnis des Benutzers bzw. aus der im Berechtigungsprofil festgelegten Benutzerkennung im fernen System. Im lokalen System wird die Datei unter gleichem Namen in dem Verzeichnis abgespeichert, aus dem das Programm aufgerufen wurde. Wenn dort bereits eine Datei mit diesem Namen besteht, wird sie überschrieben.

Ist die Übertragung erfolgreich beendet, wird die Datei anschließend im lokalen System ausgedruckt. Wenn die Datei nicht übertragen wurde, erhält der Benutzer eine Meldung. Wird vom Benutzer z.B. in Windows durch Eingabe von STRG+C ein SIGINT-Signal erzeugt, solange eine Datei noch nicht übertragen ist, wird der laufende Übertragungsauftrag abgebrochen. Die folgenden Übertragungsaufträge werden nicht angestoßen.

Das Programm ist folgendermaßen aufgebaut:

- Zunächst wird mit der Funktion *ft_open()* eine Sitzung eröffnet, wobei als Arbeitsverzeichnis `%TMP%\ft` (in Windows) bzw. `$HOME/ft` (in Unix-Systemen) fest zugeordnet wird.
- *ft_open()* liefert eine Sitzungsnummer zurück, die die Sitzung kennzeichnet und bei weiteren Funktionsaufrufen angegeben werden muss.
- Für jede der zu übertragenden Dateien wiederholen sich folgende Vorgänge:
  – Die asynchrone Dateiübertragung wird mit der Funktion *ft_transfer()* eingeleitet.
  – Fortwährend fragt das Programm ab, ob vom Benutzer ein SIGINT-Signal erzeugt wurde. Ist dies der Fall, so wird der Auftrag mit der Funktion *ft_cancel()* abgebrochen, falls der Status „Waiting“ oder „Running“ ist.
  – Solange die Dateiübertragung nicht beendet ist oder nicht abgebrochen wurde, fragt die Funktion *ft_reqstat()* den Status des Übertragungsauftrags ständig ab.
  – Ist der Status des Auftrags „Terminated“, so beginnt die Folgeverarbeitung, d.h. die ins lokale System übertragene Datei wird ausgedruckt.
  – Ist der Status des Auftrags „Aborted“, so wird eine Meldung ausgegeben.

  - Der Übertragungsauftrag wird in allen Fällen mit der Funktion *ft_reqterm()* als beendet gekennzeichnet.
- Die Sitzung mit der Funktion *ft_close()* geschlossen, nachdem alle angegebenen Dateien bearbeitet wurden.

## Beispiel 3: Inhalt eines fernen Dateiverzeichnisses auflisten

Das Programm *sample3* wird folgendermaßen aufgerufen:

```
sample3 dvz1
```

Name und Zugangsberechtigung zum fernen System werden dann im Dialog erfragt.

Das Programm liest die Einträge des Dateiverzeichnisses *dvz1* auf einem fernen System und gibt sie auf dem Bildschirm aus. Das Dateiverzeichnis muss absolut angegeben werden oder relativ zum HOME-Verzeichnis des Benutzers (siehe Seite 12) bzw. zur im Berechtigungsprofil festgelegten Benutzerkennung im fernen System. Im Beispiel werden maximal die Informationen von 10 Einträgen ausgegeben, auch wenn im angegebenen Dateiverzeichnis mehr als 10 Dateien/Verzeichnisse vorhanden sind.

Das Programm ist folgendermaßen aufgebaut:

- Mit der Funktion *ft_showdir()* werden Informationen über den Inhalt des angegebenen Dateiverzeichnisses im fernen System gelesen.
- Dabei wird ein Puffer zur Verfügung gestellt, der die Informationen von insgesamt 10 Dateien oder Verzeichnissen aufnehmen kann.
- Zusätzlich wird die Anzahl der Einträge geliefert.

## Beispiel 4: Ferne Kommandoausführung

Das Programm *sample4* wird folgendermaßen aufgerufen:

```
sample4 <Kommando>
```

Name und Zugangsberechtigung zum fernen System werden dann im Dialog erfragt.

Das Programm führt das Kommando auf einem fernen System aus und gibt das Ergebnis (Returncode, *stdout*, *stderr*) auf dem Bildschirm aus. Das Kommando muss so wie bei *ftexec* angegeben werden. Das Berechtigungsprofil im fernen System muss eine Kommandoausführung erlauben.

Das Programm ist folgendermaßen aufgebaut:

- Mit der Funktion *ft_xcopen()* wird das Kommando im fernen System ausgeführt und die Ergebnisse werden intern zwischengespeichert.
- Der Exitcode des ausgeführten Kommandos und die Anzahl der auf *stdout* und *stderr* vorliegenden Datenbytes wird dem Aufrufer mitgeteilt.
- Die Daten von *stdout* und *stderr* werden nacheinander mittels *ft_xcinfo()* in einer Schleife ausgelesen und angezeigt.
- Am Ende wird durch Aufruf von *ft_xcclose()* die Kommandoausführung beendet und nicht mehr benötigte Resourcen freigegeben.

## Beispiel 5: Ein fernes Dateiverzeichnis speicherschonend auflisten

Das Programm *sample5* wird folgendermaßen aufgerufen:

```
sample5 dvz1
```

Name und Zugangsberechtigung zum fernen System werden dann im Dialog erfragt.

Das Programm ermittelt die Attribute aller Dateien des Dateiverzeichnisses *dvz1* auf einem fernen System und gibt sie auf dem Bildschirm aus. Das Dateiverzeichnis muss absolut angegeben werden oder relativ zum HOME-Verzeichnis des Benutzers (siehe Seite 12) bzw. zur Benutzerkennung im fernen System, die im Berechtigungsprofil festgelegt ist. Im Beispiel werden Informationen zu allen gefundenen Dateien ausgegeben.

Das Programm ist folgendermaßen aufgebaut:

- Mit der Funktion *ft_sdopen()* werden Informationen über den Inhalt des angegebenen Dateiverzeichnisses im fernen System gelesen und intern zwischengespeichert.
- Dann wird der Puffer mit *ft_sdinfo()* in Schleife ausgelesen (20 Einträge am Stück) und angezeigt.
- Am Ende wird durch Aufruf von f*t_sdclose()* die Dateiattributermittlung beendet und nicht mehr benötigte Ressourcen werden freigegeben.

# Stichwörter