INTERNATIONAL STANDARD

ISO 8571-3

First edition
1988-10-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE DE NORMALISATION
МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ

Information processing systems — Open Systems Interconnection — File Transfer, Access and Management —

Part 3 :
File Service Definition

Systèmes de traitement de l'information — Interconnexion de systèmes ouverts — Gestion, accès et transfert de fichier —

Partie 3 : Définition du service de transfert de fichier

Reference number
ISO 8571-3 : 1988 (E)

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

Draft International Standards adopted by the technical committees are circulated to the member bodies for approval before their acceptance as International Standards by the ISO Council. They are approved in accordance with ISO procedures requiring at least 75 % approval by the member bodies voting.

International Standard ISO 8571-3 was prepared by Technical Committee ISO/TC 97, *Information processing systems.*

Users should note that all International Standards undergo revision from time to time and that any reference made herein to any other International Standard implies its latest edition, unless otherwise stated.

ISO 8571 consists of the following parts, under the general title *Information processing systems — Open Systems Interconnection — File Transfer, Access and Management*

- *Part 1 : General introduction*
- *Part 2 : Virtual Filestore Definition*
- *Part 3 : File Service Definition*
- *Part 4 : File Protocol Specification*

Annexes A, B, C, D and E form an integral part of this International Standard.

Printed in Switzerland

Contents

Annexes

Figures

Tables

Information processing systems — Open Systems Interconnection — File Transfer, Access and Management —

Part 3 :
File Service Definition

0 Introduction

ISO 8571 is one of a set of International Standards produced to facilitate the interconnection of computer systems. It is related to other International Standards in the set as defined by the Reference Model for Open Systems Interconnection (ISO 7498). The Reference Model subdivides the area of standardization for interconnection into a series of layers of specification, each of manageable size.

The aim of Open Systems Interconnection is to allow, with a minimum of technical agreement outside the interconnection standards, the interconnection of computer systems

a) from different manufacturers

b) under different managements

c) of different levels of complexity

d) of different ages.

ISO 8571 defines a File Service and specifies a File Protocol available within the application layer of the Reference Model. The service defined is of the category Application Service Element (ASE). It is concerned with identifiable bodies of information which can be treated as files, which may be stored within open systems or passed between application processes.

ISO 8571 defines a basic file service. It provides sufficient facilities to support file transfer, and establishes a framework for file access and file management. ISO 8571 does not specify the interfaces to a file transfer or access facility within the local system.

It is recognised that, with respect to Communication Quality of Service, (described in 14.1.2.16), work is still in progress to provide an integrated treatment of quality of service across all of the layers of the OSI Reference Model and to ensure that the individual treatments in each layer service satisfy overall quality of service objectives in a consistent manner. As a consequence, an addendum may be added to this International Standard at a later time which reflects further quality of service developments and integration.

ISO 8571 consists of the following four parts.

Part 1: General introduction
Part 2: Virtual Filestore definition
Part 3: File Service definition
Part 4: File Protocol specification

This part of ISO 8571 contains the following annexes which form part of the standard.

Annex A - Diagnostic parameter values
Annex B - Relation of attributes to primitives
Annex C - File transfer with commitment control
Annex D - Reference to FTAM control information
Annex E - State transition diagrams

1 Scope and field of application

This part of ISO 8571 defines in an abstract way the externally visible file transfer, access and management service within the OSI Application Layer in terms of:

a) the primitive actions and events of the service;

b) the parameter data associated with each primitive action and event;

c) the relationship between, and the valid sequences of, these actions and events.

The service defined in ISO 8571-3 is that which is provided by the OSI file transfer, access and management protocol ISO 8571-4 in conjunction with the Association Control Service Elements ISO 8649 and with the Presentation service ISO 8822.

ISO 8571-3 does not specify individual implementations or products, nor does it constrain the implementation of entities and interfaces within a computer system. There is, therefore, no conformance to this part of ISO 8571.

2 References

ISO 7498, *Information Processing Systems - Open Systems Interconnection - Basic Reference Model.*

ISO/TR 8509, *Information Processing Systems - Open Systems Interconnection - Service Conventions.*

ISO 8571, *Information Processing Systems - Open Systems Interconnection - File transfer, access and management.*
- *Part 1: General introduction.*
- *Part 2: Virtual Filestore definition.*
- *Part 4: File Protocol specification.*

ISO 8649, *Information Processing Systems -Open Systems Interconnection - Service definition for the Association Control Service Element.*

ISO 8822, *Information Processing Systems -Open Systems Interconnection - Connection-oriented presentation service definition.*

ISO 8831, *Information Processing Systems -Open Systems Interconnection - Job Transfer and Manipulation Concepts and Services.*

ISO 9804, *Information Processing Systems -Open Systems Interconnection - Definition of Application Service Elements - Commitment, Concurrency and Recovery.* [1)]

ISO 9805, *Information Processing Systems -Open Systems Interconnection - Specification of protocols for Application Service Elements - Commitment, Concurrency and Recovery.* [1)]

3 Definitions

Terms used in ISO 8571-3 are defined in ISO 8571-1.

4 Abbreviations

Abbreviations used in ISO 8571-3 are defined in ISO 8571-1.

5 Conventions

ISO 8571-3 uses the descriptive conventions in the OSI Service Conventions in ISO/TR 8509.

[1)] At present at the stage of draft; publication anticipated in due course.

Section one: General

6 Model of the file service

6.1 File service provider and file service users

ISO 8571-3 uses the abstract model for a service defined in the OSI service conventions in ISO TR 8509 (see note 1). The model defines the interactions between the two file service users and the file service provider. Information is passed between a file service user and the file service provider by file service primitives which may carry parameters.

One of the file service users is defined as the initiator and the other is defined as the responder (see ISO 8571-1).

The responder is an application-entity handling a virtual filestore. The virtual filestore has the properties defined in ISO 8571-2, and may be realized in a real system by a real filestore or by an application process. Attributes of the virtual filestore manipulated by the service primitives defined in ISO 8571-3 are listed in annex B.

The file service defines a single activity between an initiator and a responder (see note 2).

NOTES

1 ISO/TR 8509 defines a model for the service provided by a layer of the OSI Reference Model. The file service does not correspond to such a layer (it is a division within the application layer) but the model used is identical in all other respects.

2 At any one time, an application entity may be involved in more than one instance of the file service activity and each instance is based on a separate application association.

6.2 File service levels

Two levels of file service are defined:

a) the external file service (EFS), in which the user states its FTAM quality of service requirements, but has no awareness of error recovery, delegating such considerations to the service provider. Transfer of file data is modelled in the external file service as a series of error-free operations. Thus within the external file service there is no visibility of recoverable errors or the error recovery actions;

b) the internal file service (IFS) used by the error recovery protocol machine. This service includes primitives giving its users facilities for error recovery and control of the checkpointing mechanisms. The protocol specification which relates the external to the internal file service therefore consists of a standard set of procedures for error recovery and the protocol machine which executes these procedures is the user of the internal file service. The choice of the error recovery procedures to be used is based on the analysis of cost from the FTAM and communications quality of service requested in the External Service and local management information.

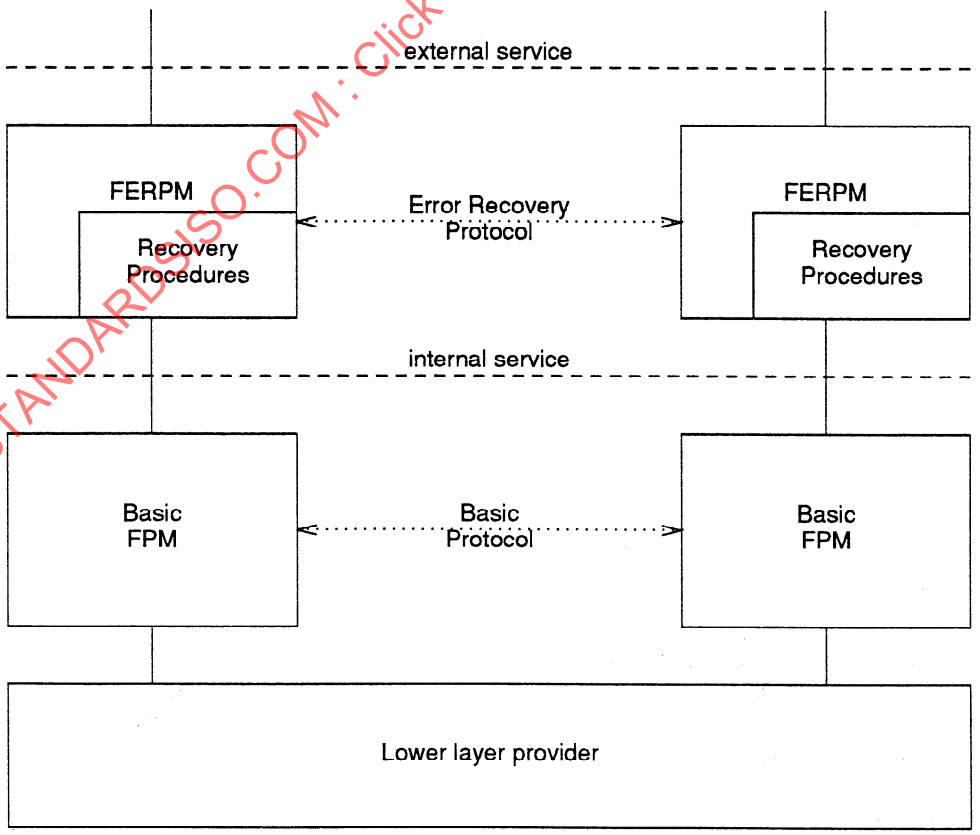

Figure 1 — Service Levels

The relationship between the internal and external file service services is shown diagrammatically in figure 1.

6.3 Regimes of the file service

Four types of file service regime are defined:

a) the FTAM regime which exists while the application association is used for the FTAM protocol.

b) the file selection regime during which a particular file is associated with the FTAM regime;

c) the file open regime during which a particular set of processing mode, presentation contexts and concurrency controls is in operation;

d) the data transfer regime during which a particular bulk data transfer specification and direction of transfer are in force.

There is at most one instance of each type of regime at any one time.

The file service provides for:

e) a sequence of file selection regimes in an FTAM regime;

f) a sequence of file open regimes in a file selection regime;

g) a sequence of data transfer regimes within a file open regime; the data transfer regimes may each be for either read or write data transfer. Write data transfer permits the operations insert, replace or extend.

Termination of a regime implies termination of all regimes nested within that regime. The nesting of regimes is shown in figure 2.

7 Services of the file service

This clause provides a short description of the services of the file service. The services and the primitives by which they are invoked are defined in sections two and three. For each service, the user of the service (the application entity that invokes the sequence of primitives) is stated. The external and internal file service levels are defined in 6.2.

7.1 FTAM regime control

Three services are associated with FTAM regime control:

a) the FTAM regime establishment service (see 14.1) is used by the initiator to create and bind an FTAM regime to the application association linking the two file service users;

b) the FTAM regime termination service (orderly) (see 14.2) is used by the initiator to dissolve the FTAM regime and unbind it from the application association between the file service users and the file service provider;

c) the FTAM regime termination service (abrupt) (see 14.3) is used by either of the service users or the service provider to dissolve the FTAM regime and its binding to the application association unconditionally.

7.2 Filestore management

ISO 8571-3 does not define any filestore management operations.

NOTE - Such operations may be included in future addenda to ISO 8571-3.

7.3 File selection regime control

Four services are associated with file selection regime control:

a) the file selection service (see 15.1) is used by the initiator to select an existing file and to bind the specified file to the FTAM regime;

b) the file deselection service (see 15.2) is used by the initiator to release the binding between the FTAM regime and the specified file;

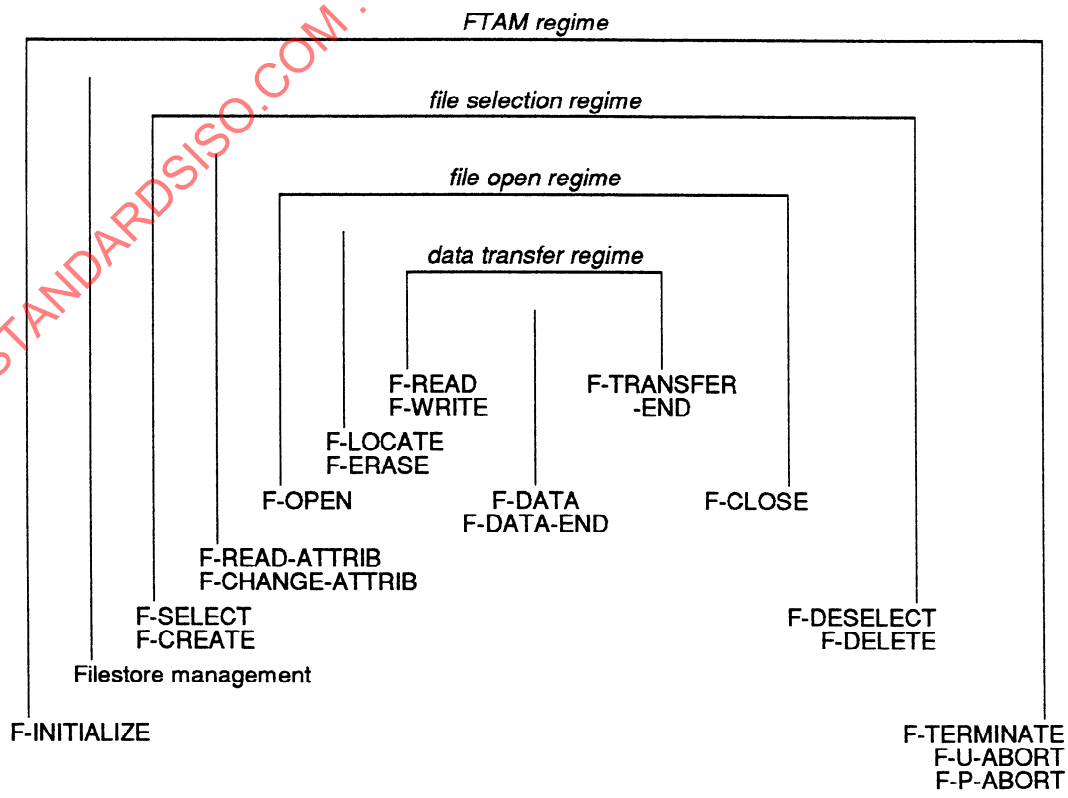

Figure 2 — File service regimes and related primitives

c) the file creation service (see 15.3) is used by the initiator either

1) to create a specified file and to select the newly created file; or

2) depending on the override parameter of F-CREATE, to select an existing file;

and then to bind the specified file to the FTAM regime;

d) the file deletion service (see 15.4) is used by the initiator to release the binding between the FTAM regime and the specified file in such a way that the previously selected file ceases to exist.

7.4 File management

Two services are associated with file management:

a) the read attributes service (see 16.1) is used by the initiator to interrogate the file attributes of the selected file;

b) the change attributes service (see 16.2) is used by the initiator to modify the file attributes of the selected file.

7.5 File open regime control

Two services are associated with file open regime control:

a) the file open service (see 17.1) is used by the initiator to establish the processing mode, presentation contexts and concurrency controls for data transfer or access;

b) the file close service (see 17.2) is used by the initiator to release the context established by the file open service.

7.6 Grouping control

Two services are associated with grouping control:

a) the beginning of grouping service (see 18.1) is used by the initiator to indicate the start of a set of primitives which are to be processed and responded to as a group;

b) the end of grouping service (see 18.2) is used by the initiator to indicate the end of a set of grouped primitives which are to be processed and responded to as a group.

7.7 Access to file content

The transfer of the FADUs of a file is performed by a bulk data transfer procedure, which forms a self contained unit. The services which make up this procedural unit are described in 7.9 and 7.10. There are two additional services associated with file access:

a) the locate file access data unit service (see 20.2) is used by the initiator to specify the identity of a file access data unit which is to be located by the responder;

b) the erase file access data unit service (see 20.3) is used by the initiator to remove a file access data unit from the file.

7.8 Bulk data transfer

Bulk data transfer refers to the transfer, optionally with checkpointing, of single file access data units (see 20.1). There are six additional services associated with different stages in bulk data transfer:

a) the read bulk data service (see 24.1) is used by the initiator to initiate a bulk data transfer from the responder (in the role of sender), to the initiator (in the role of receiver);

b) the write bulk data service (see 24.2) is used by the initiator to initiate a bulk data transfer from the initiator (in the role of sender) to the responder (in the role of receiver);

c) the data unit transfer service (see 24.3) is used by the sender to transmit bulk data;

d) the end of data transfer service (see 24.4) is used by the sender to indicate completion of the data transfer;

e) the end of transfer service (see 24.5) is used by the initiator to confirm that the data transfer is complete;

f) the cancel data transfer service (see 24.6) is used by either the sender or the receiver to cancel a data transfer activity.

7.9 Recovery

One service is associated with recovery. This service is visible only in the internal file service.

The regime recovery service (see 19.1) is used by the initiator to recreate the open regime after a failure within the open regime. Errors occurring outside the open regime are not recoverable by this service.

NOTE - Concurrency controls remain in force during a recovery attempt, or if so specified by another ASE. Otherwise, concurrency controls are released on a permanent error (see 13.2).

7.10 Checkpointing and restarting

Two services are associated with checkpointing and restarting. These services are visible only in the internal file service:

a) the checkpointing service (see 25.1) is used by the sender of data to establish marks in the flow of data for the purpose of subsequent recovery or restart;

b) the restarting data transfer service (see 25.2) is used by the sender or the receiver of data to interrupt a transfer in progress and negotiate a point at which it is to be restarted.

8 Functional units and service classes

Functional units and file service classes are logical groupings of related services defined in ISO 8571-3 for the purpose of:

a) negotiation of the file service user's requirements during FTAM regime establishment;

b) reference by other International Standards.

NOTE - The constraint set applicable to the file restricts the functionality given in 8.1 and 8.2.

8.1 Functional units

The services associated with each functional unit are specified in tables 1 and 2.

8.1.1 Kernel functional unit

The kernel functional unit supports the basic file services for the establishment and release of the FTAM and file selection regimes.

8.1.2 Read functional unit

The read functional unit supports establishment and release of the open regime and the transfer of data from the responder to the initiator.

8.1.3 Write functional unit

The write functional unit supports establishment and release of the open regime and the transfer of data from the initiator to the responder.

8.1.4 File access functional unit

The file access functional unit allows an FADU in the file access structure to be located for file access and allows an FADU to be manipulated within the file access structure.

8.1.5 Limited file management functional unit

The limited file management functional unit supports file management for the creation and deletion of files and for the interrogation of file attributes.

8.1.6 Enhanced file management functional unit

The enhanced file management functional unit extends the capabilities of the limited file management functional unit to include the modification of file attributes.

8.1.7 Grouping functional unit

The grouping functional unit allows several regimes to be established in one exchange, by combining several independent primitives into a group for procedural purposes.

8.1.8 FADU Locking functional unit

The FADU locking functional unit allows the invocation of concurrency control locks on a per FADU basis in addition to a file basis.

8.1.9 Recovery functional unit

The recovery functional unit allows the initiator to recreate an open regime which has been destroyed by some failure. The recovery may be immediate or deferred on the existing, or different a, association.

8.1.10 Restart data transfer functional unit

The restart data transfer functional unit allows a data transfer to be interrupted and restarted immediately at a negotiated point within the current transfer.

8.1.11 Service classes and functional units

Table 1 shows which of the functional units are mandatory and which optional in each of the service classes: transfer, access, management, transfer and management, and unconstrained. Service classes are defined in 8.2.

The recovery and restart functional units are never explicitly visible in the external file service. Where they are shown as optional in table 2 it is in the internal file service that they are optional. The external file service and the internal file service are defined in 6.2.

8.2 Service classes

Five file service classes are defined in terms of combinations of functional units:

a) file transfer class (see 8.2.1);

b) file access class (see 8.2.2);

c) file management class (see 8.2.3);

d) file transfer and management class (see 8.2.4);

e) unconstrained class (see 8.2.5).

8.2.1 File transfer class

The file transfer class consists of;

a) the kernel functional unit;

b) the grouping functional unit;

c) one or both of the read or write functional units;

d) optionally, the limited file management functional unit;

e) optionally, but only if the limited file management functional unit is present, the enhanced file management functional unit;

f) optionally, in the internal file service, the recovery functional unit.

g) optionally, in the internal file service, the restart data transfer functional unit.

In the file transfer service class, the use of the services is constrained so that there is a sequence of zero or more FTAM events on the application association. Each FTAM event is a sequence of:

1) a single grouped sequence to establish a file open regime. This sequence consists of:

- an F-BEGIN-GROUP primitive;
- an F-SELECT or F-CREATE primitive;
- optionally, an F-READ-ATTRIB primitive;
- optionally, an F-CHANGE-ATTRIB primitive;
- an F-OPEN primitive;
- an F-END-GROUP primitive.

2) a single bulk data transfer procedure, for either a read transfer or a write transfer. The processing mode parameter on the F-OPEN primitive is set to either a read or a valid write action, as defined in the constraint set, but not both.

3) a single grouped sequence to release the file open and select regimes. This sequence consists of:

- an F-BEGIN-GROUP primitive;
- an F-CLOSE primitive;
- optionally, an F-READ-ATTRIB primitive;
- optionally, an F-CHANGE-ATTRIB primitive;
- an F-DESELECT or F-DELETE primitive;
- an F-END-GROUP primitive.

The threshold parameter is set equal to the number of primitives between F-BEGIN-GROUP and F-END-GROUP. The threshold parameter is defined in clause 18.

NOTE - Each of the primitives shown as optional in these sequences may only be present if the corresponding functional unit was negotiated during the FTAM regime establishment.

8.2.2 File access class

The file access class consists of:

a) the kernel functional unit;

b) both of the read and write functional units;

c) the file access functional unit;

d) optionally, the grouping functional unit. If the grouping functional unit is successfully negotiated, its valid use in any instance by the initiator is optional but its acceptance by the responder is always mandatory.

Table 1 — Services and functional units of the External File Service

Functional Unit	Services	Service classes					Clause
		T	A	M	TM	U	
U1 Kernel	FTAM regime establishment FTAM regime termination (orderly) FTAM regime termination (abrupt) File selection File deselection	M	M	M	M	M	14.1 14.2 14.3 15.1 15.2
U2 Read	Read bulk data Data unit transfer End of data transfer End of transfer Cancel data transfer File open File close	*	M		*	O	24.1 24.3 24.4 24.5 24.6 17.1 17.2
U3 Write	Write bulk data Data unit transfer End of data transfer End of transfer Cancel data transfer File open File close	*	M		*	O	24.2 24.3 24.4 24.5 24.6 17.1 17.2
U4 File access	Locate FADU Erase FADU (requires U2 or U3)		M			O	20.2 20.3
U5 Limited file management	File creation File deletion Read attributes	O	O	M	M	O	15.3 15.4 16.1
U6 Enhanced file management	Change attributes (requires U5)	O	O	O	O	O	16.2
U7 Grouping	Beginning of grouping End of grouping	M	O	M	M	O	18.1 18.2
U8 FADU locking	FADU Locking (requires U2 or U3) and U4		O			O	

Table 2 — Services and functional units of the Internal File Service

Functional Unit	Services	Service classes					Clause
		T	A	M	TM	U	
U9 Recovery	Regime Recovery Checkpointing Cancel data transfer (for recoverable errors)	O	O		O	O	19.1 25.1 24.6
U10 Restart data transfer	Restarting data transfer Checkpointing Cancel data transfer (for recoverable errors)	O	O		O	O	25.2 25.1 24.6

Key to tables 1 and 2

Service class abbreviations

T = File transfer class
A = File access class
M = File management class
TM = File transfer and management class
U = Unconstrained class

Abbreviations within service class

M = Mandatory
O = Optional
* = At least one of U2 or U3
blank = not permitted

e) optionally, the limited file management functional unit;

f) optionally, but only if the limited file management functional unit is present, the enhanced file management functional unit;

g) optionally, the FADU locking functional unit;

h) optionally, in the internal file service, the recovery functional unit.

i) optionally, in the internal file service, the restart data transfer functional unit.

NOTES

1 The threshold constraints applicable to the file transfer class do not apply to the file access class.

2 The constraints on grouping and sequence of events defined in 8.2.1 do not apply to the file access service class.

8.2.3 File management class

The file management class consists of:

a) the kernel functional unit;

b) the limited file management functional unit;

c) optionally, the enhanced file management functional unit;

d) the grouping functional unit.

In the file management service class the use of the services is constrained so that there is a sequence of zero or more FTAM events on the application association. Each FTAM event is a sequence of:

- an F-BEGIN-GROUP primitive;
- an F-SELECT or F-CREATE primitive;
- optionally, an F-READ-ATTRIB primitive;
- optionally, an F-CHANGE-ATTRIB primitive;
- an F-DESELECT or F-DELETE primitive;
- an F-END-GROUP primitive.

The threshold parameter, defined in clause 18, is set equal to the number of primitives between F-BEGIN-GROUP and F-END-GROUP.

8.2.4 File transfer and management class

The file transfer and management class consists of:

a) the kernel functional unit;

b) the grouping functional unit;

c) one or both of the read or write functional units;

d) the limited file management functional unit;

e) optionally, the enhanced file management functional unit;

f) optionally, in the internal file service, the recovery functional unit;

g) optionally, in the internal file service, the restart data transfer functional unit.

In the file transfer and management service class, the use of the services is constrained so that there is a repeated sequence of FTAM events on the application association. Each FTAM event is either;

a transfer comprising:

1) a single grouped sequence to establish a file open regime. This sequence consists of:

- an F-BEGIN-GROUP primitive;
- an F-SELECT or F-CREATE primitive;
- optionally, an F-READ-ATTRIB primitive;
- optionally, an F-CHANGE-ATTRIB primitive;
- an F-OPEN primitive;
- an F-END-GROUP primitive.

2) a single bulk data transfer procedure, for either a read transfer or a write transfer. The processing mode parameter on the F-OPEN primitive is set to either read or a valid write action, as defined in the constraint set, but not both.

3) a single grouped sequence to release the file open and select regimes. This sequence consists of:

- an F-BEGIN-GROUP primitive;
- an F-CLOSE primitive;
- optionally, an F-READ-ATTRIB primitive;
- optionally, an F-CHANGE-ATTRIB primitive;
- an F-DESELECT or F-DELETE primitive;
- an F-END-GROUP primitive.

or a single grouped sequence to effect file management consisting of:

- an F-BEGIN-GROUP primitive;
- an F-SELECT or F-CREATE primitive;
- optionally, an F-READ-ATTRIB primitive;
- optionally, an F-CHANGE-ATTRIB primitive;
- an F-DESELECT or F-DELETE primitive;
- an F-END-GROUP primitive.

The threshold parameter, defined in clause 18, is set equal to the number of primitives between F-BEGIN and F-END-GROUP.

NOTE - The primitives in the functional units may only be present in these sequences if the corresponding functional units were negotiated during the FTAM regime establishment.

8.2.5 Unconstrained class

The unconstrained class consists of :

a) the kernel functional unit;

b) optionally, any other functional units.

NOTES

1 The threshold parameter constraints do not apply to the unconstrained class.

2 The unconstrained service class is provided for definition of non standard applications. No restrictions are imposed by ISO 8571 on this class.

8.3 Application Entity roles

An application entity keeps the same role (either initiator or responder) for the duration of the FTAM regime. This specific role applies to each of the available functional units.

9 Levels of file service

Two levels of file service are defined in 6.2. They are:

a) the external file service (EFS), in which the user states its FTAM quality of service requirements, but has no awareness of error recovery.

b) the internal file service (IFS) used by the error recovery protocol machine. This service includes primitives giving its users facilities for error recovery and control of the checkpointing mechanisms.

The functional units providing visible services in the external file service and in the internal file service are defined in table 3.

The external file service is supported by the error recovery protocol, which may be null, and indirectly by the basic protocol specified in ISO 8571-4. The internal file service is supported by the basic protocol.

Selection of a named functional unit in the external service implies selection of the same named functional unit in the internal file service. In the internal file service the recovery and restart functional units are optional. The primitives of the restart and recovery functional units will not be visible to the external file service user.

Table 3 — Functional units in the file services

External file service	Internal file service
Kernel	Kernel
Read	Read
Write	Write
File access	File access
Limited file management	Limited file management
Enhanced file management	Enhanced file management
Grouping	Grouping
FADU locking	FADU locking
	Recovery
	Restart data transfer

10 Negotiation of service class, FTAM QoS and functional units

Service class and FTAM quality of service are negotiated independently on the F-INITIALIZE exchange. Each of these negotiations contributes to the final selection of functional units available on the association.

The kernel functional unit is always available and is not included in the negotiations.

The availability of the functional units, read, write, file access, limited file management, enhanced file management, grouping and FADU locking is controlled by the service class negotiated (see Tables 1 and 2).

The proposal of either, or both, of the restart or recovery functional units is as a result of a local decision based on the underlying service and the FTAM quality of service negotiated between the two file service users.

10.1 Service Class

The initiator states in the service class parameter of the F-INITIALIZE request primitive the service classes required. The valid combinations are defined in table 4. Each of the combinations listed may have included with it the unconstrained class.

Table 4 — Service Class Combinations

Abbreviation	Service Class Capability
T	Transfer class
M	Management class
A	Access class
T,A	Transfer class, Access class
T,M,TM	Transfer class, Management class, Transfer and Management class
A,T,M,TM	Access class, Transfer class, Management class or Transfer and Management class

During negotiation the service provider removes any service classes it is unable to support and signals the remaining set to the responder on the F-INITIALIZE indication.

The responder removes from the service class list all classes it is unable to support. It then selects from the remaining classes the highest service class it is able to support. It returns this on the service class parameter of the F-INITIALIZE response primitive. Support of both the transfer class and the management class requires support of the transfer and management class.

The order of service classes is defined as, highest to lowest; access, transfer and management, transfer, management, unconstrained.

The agreed service class is signalled to the initiator on the F-INITIALIZE confirm primitive. This negotiation of service classes always results in a single service class being agreed. Table 5 details the outcome of the negotiation.

The functional unit constraints of each service class are defined in tables 1 and 2. The functional unit negotiation is defined in 10.3.

10.2 FTAM Quality of Service

The initiator states on the F-INITIALIZE request primitive the FTAM quality of service indicating the error classes the application is susceptible to.

The service provider adds to the functional units included in the functional units parameter neither, one or both of the recovery and restart functional units, according to the file service user's requested FTAM quality of service, local knowledge and which of the error recovery and restart functional units it is able to support. The service provider indicates to the responder, on the FTAM quality of service parameter of the F-INITIALIZE indication primitive, the value requested by the initiator and possibly reduced by the service provider.

The responding file protocol machine uses the initiator's FTAM quality of service, local knowledge and capability to determine whether it is able to support the requested FTAM quality of service. If the responder is able to provide the requested FTAM quality of service it returns the parameter unchanged. The recovery and/or restart functional units are removed from the set of functional units parameter if the responding file protocol machine is unable to support them.

If the responder is unable to support the requested FTAM quality of service, it returns on the F-INITIALIZE response primitive the value it is able to support. The service provider may reduce the value of the FTAM quality of service if the resulting functional units do not provide the capability required. The final value of the FTAM quality of service is signalled to the file service user on the F-INITIALIZE confirm primitive. The negotiation of this parameter does not, in itself, prevent the establishment of an FTAM regime. If the quality of service is below that acceptable to the file service user the regime may be terminated by the issue of an F-TERMINATE request primitive by the file service user.

10.3 Functional units

The initiator states on the F-INITIALIZE request primitive its requirement in terms of functional units. The complete list proposed comprises all the mandatory functional units, except the kernel, in the service classes proposed in the service class parameter plus additional optional functional

units. The recovery and restart functional units are only inserted by the service provider.

The service provider removes from this set any functional units it is unable to support and signals the remaining set to the responder on the F-INITIALIZE indication primitive. The responder removes from this set any functional units it is unable to support and signals the remaining set to the service provider on the F-INITIALIZE response primitive.

The service provider signals the same set to the initiator on the F-INITIALIZE confirm primitive. This negotiated set of functional units is then available for use in the established FTAM regime.

Table 5 — Service Class Negotiation

Initiator capability	Responder capability					
	T	M	A	T,A	T,M,TM	A,T,M,TM
T	T	—	—	T	T	T
M	—	M	—	—	M	M
A	—	—	A	A	—	A
T,A	T	—	A	A	T	A
T,M,TM	T	M	—	T	TM	TM
A,T,M,TM	T	M	A	A	TM	A

Key to Table 5

In table 5 the symbols have the meaning defined in table 4. Additionally the — symbol has the following meaning:

a) if both initiator and responder capabilities included the unconstrained service class then that class is the result of the negotiation.

b) if either or both lack the unconstrained class capability then the responder rejects the FTAM regime establishment attempt with a state result parameter indicating failure and, optionally, an appropriate diagnostic value.

Section two : Definition of file service primitives

11 File service primitives

Each of the services constituting the file service is achieved by invoking a sequence of file service primitives.

Table 6 lists for each service;

a) the primitives associated with the service;

b) the parameters associated with the primitives;

c) the file service user that is permitted to issue the request primitive;

In table 6, parameters and primitives which are only visible in the internal service are enclosed in square brackets.

The semantics of the primitives and their parameters are defined in clauses 13 to 20.

The primitives of the bulk data transfer service are listed in clause 21 and defined in clauses 23 to 25.

12 Sequences of primitives

This clause defines the constraints on the permissible sequences for the EFS in which the primitives defined in clauses 14 to 20 can occur. The individual primitives in a service may occur only in the sequences given as part of the primitive definitions. The F-RECOVER, F-RESTART and F-CHECK service primitives are not issued in the external file service.

12.1 Normal sequences

The normal progress of use of the file service is illustrated by the state transition diagram shown in figure 3, applicable separately to each application entity. Full state transition diagrams are included in annex E.

12.2 Constraints on the issue of primitives

The primitives may be issued in any sequence consistent with the constraints given in tables 7 to 10. The sequences

Table 6 — File service primitives

Primitive	Confirmed	Request by	Parameters
F-INITIALIZE	Yes	Initiator	State Result Action Result Called application title Calling application title Responding application title Called presentation address Calling presentation address Responding presentation address Presentation context management Application context name Service class Functional units Attribute groups Shared ASE information FTAM QoS Communication QoS Contents type list Initiator Identity Account Filestore password Diagnostic [Checkpoint window]
F-TERMINATE	Yes	Initiator	Shared ASE information Charging
F-U-ABORT	No	Either	Action Result Diagnostic
F-P-ABORT	No	Provider	Action result Diagnostic
F-SELECT	Yes	Initiator	State Result Action Result Attributes Requested Access Access passwords Concurrency control Shared ASE information Account Diagnostic

Table 6 (continued) — File service primitives

Primitive	Confirmed	Request by	Parameters
F-DESELECT	Yes	Initiator	Action Result Charging Shared ASE information Diagnostic
F-CREATE	Yes	Initiator	State Result Action Result Override Initial Attributes Create password Requested Access Access passwords Concurrency control Shared ASE information Account Diagnostic
F-DELETE	Yes	Initiator	Action Result Shared ASE information Charging Diagnostic
F-READ-ATTRIB	Yes	Initiator	Action Result Attribute names Attributes Diagnostic
F-CHANGE-ATTRIB	Yes	Initiator	Action Result Attributes Diagnostic
F-OPEN	Yes	Initiator	State Result Action result Processing mode Contents type Concurrency control Shared ASE information Enable FADU Locking Diagnostic [Activity identifier] [Recovery mode]
F-CLOSE	Yes	Initiator	Action Result Shared ASE information Diagnostic
F-BEGIN-GROUP	Yes	Initiator	Threshold
F-END-GROUP	Yes	Initiator	-
[F-RECOVER]	Yes	Initiator	[State Result Action result Activity identifier Bulk transfer number Requested Access Access passwords Contents type Recovery point Diagnostic]
F-LOCATE	Yes	Initiator	Action Result FADU identity FADU Lock Diagnostic
F-ERASE	Yes	Initiator	Action Result FADU identity Diagnostic

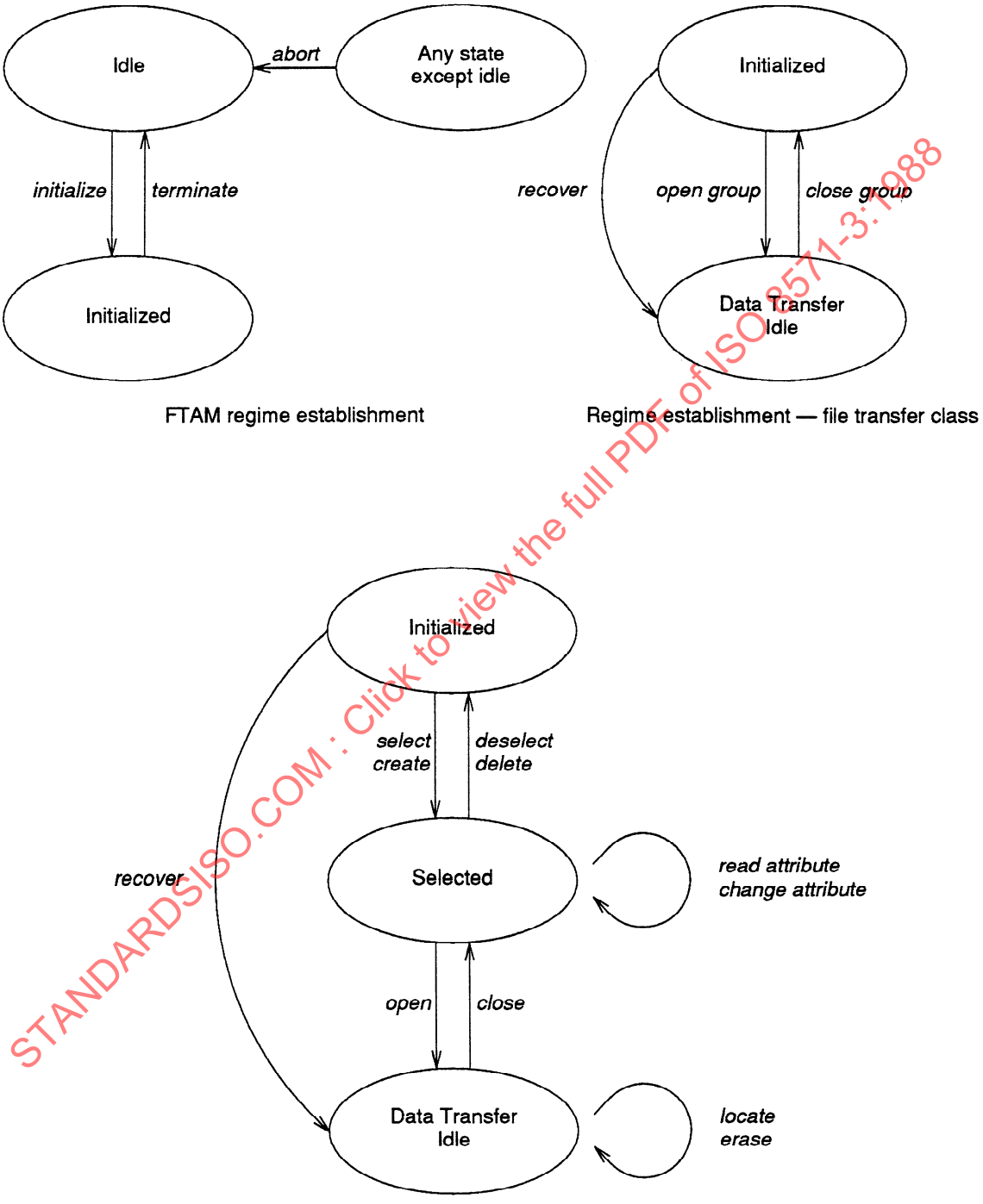

Figure 3 — Simplified State Diagram for successful activity (see Annex E)

of primitives are defined under the individual services. Individual primitive sequences may be interleaved to form the following grouped sequences, which are expressed using the notation defined in annex E.

a) F-BEGIN-GROUP
(F-SELECT|F-CREATE)
[F-READ-ATTRIB] [F-CHANGE-ATTRIB]
F-OPEN
F-END-GROUP

b) F-BEGIN-GROUP
F-CLOSE
[F-READ-ATTRIB] [F-CHANGE-ATTRIB]
(F-DESELECT|F-DELETE)
F-END-GROUP

c) F-BEGIN-GROUP
(F-SELECT|F-CREATE)
[F-READ-ATTRIB] [F-CHANGE-ATTRIB]
(F-DESELECT|F-DELETE)
F-END-GROUP

d) F-BEGIN-GROUP
(F-SELECT|F-CREATE)
[F-READ-ATTRIB] [F-CHANGE-ATTRIB]
F-END-GROUP

e) F-BEGIN-GROUP
[F-READ-ATTRIB] [F-CHANGE-ATTRIB]
(F-DESELECT|F-DELETE)
F-END-GROUP

NOTES

1 Not all the above sequences are allowed in the file transfer, file management and file transfer and management classes (see 8.2.1, 8.2.3, 8.2.4). In the file transfer class, only sequences (a) and (b),are valid, and the threshold parameter is set so that the sequences either succeed or fail as a whole, i.e. set to the number of primitives between the begin group and end group primitives. In the file management class only sequence (c) is valid. (see 8.2.3). In the file transfer and management class only the sequences (a), (b) and (c) are valid.

2 Other constraints will affect the ability of the file service user or file service provider to invoke the various procedures, such as flow control constraints on sending data or constraints on the ability of a file service user to accept spontaneous F-P-ABORT indications from the file service provider.

12.3 Conventions

The following conventions apply to entries in tables 7 to 10.

12.3.1 Conventions for tables 7 and 8

In tables 7 and 8, the entry "yes" implies that the succession given can occur. The column "start of file regimes" indicates entry to tables 9 and 10. The row "end of file regimes" indicates exit from tables 9 and 10, and the row "within file regimes" indicates any other entry in tables 9 and 10.

12.3.2 Conventions for tables 9 and 10

In tables 9 and 10, the entries indicate the functional units required for the succession to occur. The entries are:

Kernel	kernel functional unit
Grouping	grouping functional unit
Lmgt	limited file management functional unit
Emgt	enhanced file management functional unit
Recover	recover functional unit
Access	file access functional unit
G-Lmgt	grouping and limited file management functional units
G-Emgt	grouping and enhanced file management functional units
Lmgt-Rec	limited file management and recovery functional units
Rec-Acc	recovery and file access functional units
G-Rec	grouping and recovery functional units
G-Acc	grouping and file access functional units

The row "start file regimes" indicates entry from tables 7 and 8, and the column "end file regimes" indicates return to tables 7 and 8. The row and column "bulk data transfer" indicate use of the bulk data transfer service defined in section three.

In tables 7 to 10, the value of the state result parameter is indicated by:

(+ve) positive response or confirm; the primitive carries a state result indicating a successful transition;

(-ve) negative response or confirm; the primitive carries a state result indicating an unsuccessful transition or regime establishment attempt.

Table 7 — Sequence of service primitives for FTAM regime establishment — initiator

Previous file service event	May next issue			
	F-INITIALIZE request	F-TERMINATE request	F-U-ABORT request	Start File Selection Regimes
Idle	Yes			
F-INITIALIZE request			Yes	
F-INITIALIZE confirm (+ve)			Yes	Yes
F-INITIALIZE confirm (-ve)	Yes			
F-TERMINATE request			Yes	
F-TERMINATE confirm	Yes			
F-U-ABORT request	Yes			
F-U-ABORT indication	Yes			
F-P-ABORT indication	Yes			
Within file selection regimes			Yes	
End file selection regimes		Yes	Yes	Yes

Table 8 — Sequence of service primitives for FTAM regime establishment — responder

Previous file service event	May next issue			
	F-INITIALIZE response	F-TERMINATE response	F-U-ABORT request	Start File Selection Regimes
Idle				
F-INITIALIZE indication	Yes		Yes	
F-INITIALIZE response (+ve)			Yes	Yes
F-INITIALIZE response (-ve)				
F-TERMINATE indication		Yes	Yes	
F-TERMINATE response				
F-U-ABORT request				
F-U-ABORT indication				
F-P-ABORT indication				
Within file selection regimes			Yes	
End file selection regimes			Yes	Yes

Table 9 — Sequence of service primitives for file service regimes — initiator

Previous file service event	May next issue					
	F-SELECT request	F-DESELECT request	F-CREATE request	F-DELETE request	F-READ -ATTRIB request	F-CHANGE -ATTRIB request
Start file selection regimes	Kernel		Lmgt			
F-SELECT request		Grouping		G-Lmgt	G-Lmgt	G-Emgt
F-SELECT confirm (+ve)		Kernel		Lmgt	Lmgt	Emgt
F-SELECT confirm (-ve)	Kernel		Lmgt			
F-DESELECT request						
F-DESELECT confirm	Kernel		Lmgt			
F-CREATE request		Grouping		G-Lmgt	G-Lmgt	G-Emgt
F-CREATE confirm (+ve)		Lmgt		Lmgt	Lmgt	Emgt
F-CREATE confirm (-ve)	Lmgt		Lmgt			
F-DELETE request						
F-DELETE confirm	Lmgt		Lmgt			
F-READ-ATTRIB request		G-Lmgt		G-Lmgt		G-Emgt
F-READ-ATTRIB confirm		Lmgt		Lmgt	Lmgt	Emgt
F-CHANGE-ATTRIB request		G-Emgt		G-Emgt		
F-CHANGE-ATTRIB confirm		Emgt		Emgt	Emgt	Emgt
F-OPEN request						
F-OPEN confirm (+ve)						
F-OPEN confirm (-ve)		Kernel		Lmgt	Lmgt	Emgt
F-CLOSE request		Grouping		G-Lmgt	G-Lmgt	G-Emgt
F-CLOSE confirm		Kernel		Lmgt	Lmgt	Emgt
F-BEGIN-GROUP request	Grouping	Grouping	G-Lmgt	G-Lmgt	G-Lmgt	G-Emgt
F-BEGIN-GROUP confirm						
F-END-GROUP request						
F-END-GROUP confirm	Grouping	Grouping	G-Lmgt	G-Lmgt	G-Lmgt	G-Emgt
F-RECOVER request						
F-RECOVER confirm (+ve)						
F-RECOVER confirm (-ve)	Recover		Lmgt			
Bulk data transfer						
F-LOCATE request						
F-LOCATE confirm						
F-ERASE request						
F-ERASE confirm						

May next issue								
F-OPEN request	F-CLOSE request	F-BEGIN -GROUP request	F-END -GROUP request	F-RECOVER request	Bulk Data Transfer	F-LOCATE request	F-ERASE request	End file selection regimes
		Grouping		Recover				Kernel
Grouping								
Kernel		Grouping						
		Grouping		Recover				Kernel
			Grouping					
		Grouping		Recover				Kernel
G-Lmgt								
Lmgt		G-Lmgt						
		G-Lmgt		Lmgt-Rec				Lmgt
			G-Lmgt					
		G-Lmgt		Lmgt-Rec				Lmgt
G-Lmgt			G-Lmgt					
Lmgt		G-Lmgt						
G-Emgt			G-Emgt					
Emgt		G-Emgt						
			Grouping					
	Kernel	Grouping			Kernel	Access	Access	
Kernel		Grouping						
Kernel		Grouping						
	Grouping							
Grouping	Grouping	Grouping		G-Rec	Grouping	Access	Access	Grouping
	Recover	G-Rec			Recover	Rec-Acc	Rec-Acc	
		G-Rec		Recover				Recover
	Kernel	Grouping			Kernel	Access	Access	
	Access	G-Acc			Access	Access	Access	
	Access	G-Acc			Access	Access	Access	

Table 10 — Sequence of service primitives for file service regimes — responder

Previous file service event	May next issue					
	F-SELECT response	F-DESELECT response	F-CREATE response	F-DELETE response	F-READ -ATTRIB response	F-CHANGE -ATTRIB response
Start file selection regimes						
F-SELECT indication	Kernel					
F-SELECT response (+ve)		Grouping		G-Lmgt	G-Lmgt	G-Emgt
F-SELECT response (-ve)						
F-DESELECT indication		Kernel				
F-DESELECT response						
F-CREATE indication			Lmgt			
F-CREATE response (+ve)		G-Lmgt		G-Lmgt	G-Lmgt	G-Emgt
F-CREATE response (-ve)						
F-DELETE indication				Lmgt		
F-DELETE response						
F-READ-ATTRIB indication					Lmgt	
F-READ-ATTRIB response		G-Lmgt		G-Lmgt		G-Emgt
F-CHANGE-ATTRIB indication						Emgt
F-CHANGE-ATTRIB response		G-Emgt		G-Emgt		
F-OPEN indication						
F-OPEN response (+ve)						
F-OPEN response (-ve)						
F-CLOSE indication						
F-CLOSE response		Grouping		G-Lmgt	G-Lmgt	G-Emgt
F-BEGIN-GROUP indication						
F-BEGIN-GROUP response	Grouping	Grouping	G-Lmgt	G-Lmgt	G-Lmgt	G-Emgt
F-END-GROUP indication						
F-END-GROUP response						
F-RECOVER indication						
F-RECOVER response (+ve)						
F-RECOVER response (-ve)						
Bulk data transfer						
F-LOCATE indication						
F-LOCATE response						
F-ERASE indication						
F-ERASE response						

May next issue								
F-OPEN response	F-CLOSE response	F-BEGIN -GROUP response	F-END -GROUP response	F-RECOVER response	Bulk Data Transfer	F-LOCATE response	F-ERASE response	End file selection regimes
								Kernel
Grouping								
			Grouping					Kernel
			Grouping					Kernel
G-Lmgt								
			G-Lmgt					Lmgt
			G-Lmgt					Lmgt
G-Lmgt			G-Lmgt					
G-Emgt			G-Emgt					
Kernel								
			Grouping					
			Grouping					
	Kernel							
	Grouping							
		Grouping						
								Grouping
				Recover				
								Recover
						Access		
							Access	

12.4 Confirmed Services

For all confirmed services the sequence of events in a successful exchange is shown in figure 4, where F-XXX indicates the name of the service element.

A request to establish a new regime (F-INITIALIZE, F-SELECT, F-CREATE or F-OPEN) may be rejected by use of a response with state result parameter indicating failure (see 13.1).

13 Common file service parameters

File service parameters which apply to several primitives are defined in this clause and referenced by the primitive definitions. Parameters which apply to only one primitive are defined in association with the primitive concerned. Annex B relates the primitives to any attributes they modify.

13.1 State result

The state result parameter conveys information relating to the service state machine. This parameter is returned on the response and confirm primitives of services which can have failed to change regimes as required by the request and indication primitives. The state result parameter is not present on primitives which do not cause a state change, neither is it present on primitives which force state changes but cannot fail. (for example F-DESELECT). The values of the state result are "success" or "failure". This parameter is not used to set any activity attribute.

13.2 Action result

The action result parameter conveys information which summarises that available in the diagnostic parameter (See 13.13). The value is never less severe than the most severe diagnostic value. In the IFS the valid values of the action result parameter are "success", "transient error" or "permanent error". In the external service the only valid values are success or permanent error. The term "unsuccessful" is used in ISO 8571-3 to indicate either transient or permanent error. When a response or confirm primitive includes a state result indicating failure the action result is set to "transient error" or "permanent error". A "successful" action result may be accompanied by a diagnostic of an informative error type. This parameter is not used to set any activity attribute.

13.3 Account

The account parameter identifies the account to which costs incurred in the regime which is being established are to be charged. The range of values the parameter may take is equal to that defined for the current account activity attribute. This parameter is used to set the current account activity attribute, for the duration of the regime being entered. If the parameter is not present, no value is assigned to the activity attribute, which is then either unset or retains some previous value based on local considerations. The current account activity attribute reverts to its previous value at the end of a regime.

13.4 Charging

The charging parameter conveys information on the costs attributed to the account during the regime which is being released. The value of the parameter is a list of triples; the elements of each triple are: a GraphicString resource identifier, a GraphicString charging unit and an integer charge value. The charging parameter is present at the end of a regime, only if the account parameter was present at the beginning of that regime. It is not mandatory to return a charge if that charge is zero. The resource identifier and charging unit values are implementation dependent. The return of a charge may be associated with the restoration of the current account activity attribute to its previous value.

13.5 Attributes

The attributes parameter conveys a list of file attribute names and file attribute values associated with the file. F-INITIALIZE negotiates the attribute groups which are available for the duration of the FTAM regime. Subsequent primitives only use those attributes which were negotiated.

On primitives other than F-SELECT issued by the initiator, the parameter conveys new values to be assigned to the file attributes. On primitives issued by the responder, the parameter reports the current values of the file attributes, or indicates that there is no value available.

The filename attribute value on F-SELECT identifies the file to be selected, or on F-CREATE the name of the file to be created and selected. In the request and indication primitives it indicates the file required, and in the response and confirm primitives it indicates the file actually selected.

NOTE - If, for example, the filename requested gave a generic name or a generation name, the name selected might differ from that requested.

The file attributes, the range of values they may take and the actions to be taken are defined in the virtual filestore definition (ISO 8571-2). Annex B specifies which file attributes may be manipulated by each primitive. This parameter is not used to set any activity attribute.

13.6 Requested access

The requested access parameter indicates the basis on which the file is being selected or recovered. The value gives, as a vector, the actions to be performed during the selection. The elements of the vector correspond to the read, insert, replace, extend, erase, read attribute, change attribute and delete file actions, and each element indicates whether the action is required or not.

The value of the parameter on request and indication primitives indicates the initiator's requirements.

NOTE - When a file is selected by use of F-SELECT, F-CREATE or F-RECOVER the requested access parameter is used to state the maximum facilities the current user will require during the whole select regime. These access requirements may be password protected by the access control file attribute which is matched to the access passwords parameter (see 13.7).

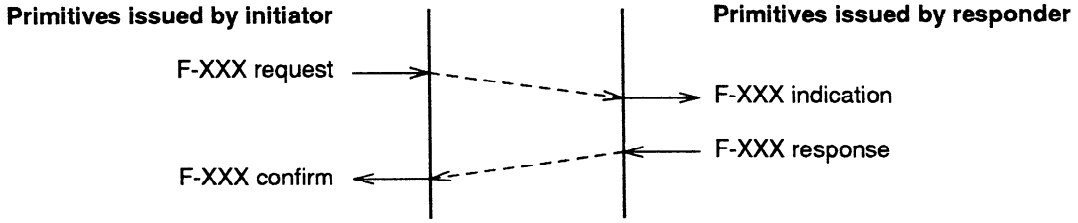

Figure 4 — Confirmed service

The requested access is further qualified by the concurrency control parameter which specifies how multiple users are to be restricted when requiring access to the same file. The requested access parameter includes only those actions to be performed by the user, any action not included is not available later in the select regime.

The relationship between attributes and actions is detailed in ISO 8571-2. This parameter is used to set the current access request activity attribute.

13.7 Access Passwords

The access passwords parameter provides passwords associated with the actions specified in the requested access parameter. This parameter is only available if the security attribute group has been negotiated. The range of values the parameter takes is equal to that defined for the current access passwords activity attribute. This parameter is used to set the current access passwords activity attribute.

13.8 Concurrency Control

The concurrency control parameter indicates the relation of the file select, or file open, regime to other activities on the same file. The value is a vector whose elements indicate, for each of the actions specified in the requested access parameter elements (see 13.6), which access locks are required. The locks define the access available to the user and the access available to any other user. The concurrency control parameter qualifies the requested access for the regime being entered. The shared and exclusive locks are only permitted for the actions negotiated as available by the requested access parameter. For those actions not available the only locks permitted are no access and not required.

The locks available are as follows:

a) not required - I will not perform the operation - others may;

b) shared - I may perform the operation - so may others;

c) exclusive - I may perform the operation - others may not;

d) no access - no one may perform the operation.

They are available on the following actions: read, insert, replace, extend, erase, read attribute, change attribute and delete file.

If FADU locking has not been invoked by the Enable FADU Lock parameter, the locks have a scope of the open regime (see 17.1.2.7).

If FADU locking is negotiated, the locks have the scope of the data transfer regime and values of "not required" are assumed elsewhere in the open regime, except where specifically overridden by explicit FADU locks.

This parameter is used to set the current concurrency control activity attribute.

13.9 FADU Lock

This parameter may be used to set individual FADU locks on or off, and if it is absent the state of the locks is unchanged. Switching the locks changes the value agreed at select and/or open from "not required" to "no access" and from "shared" to "exclusive" until the lock is explicitly switched off, or the FADU is erased, or the file is closed. Switching off the lock returns the FADU to its previous state for both reading and writing.

If for a BDT operation the lock is being set on, then the action of FADU locking is performed before the transfer. If the FADU lock is being set off then the transfer is preformed before the FADU lock is released.

The FADU lock parameter is available if;

a) the FADU locking functional unit is negotiated; and

b) the Enable FADU locking parameter on the F-OPEN request was set on

This parameter is not used to set any activity attribute.

13.10 Shared ASE information

The shared ASE Information parameter allows information of other ASEs to be associated with FTAM primitives. The information to be conveyed and the symbiotic relationship established between FTAM and other ASEs is defined by the application context. One example of such a combination is given in Annex C, which defines the combination of FTAM and CCR for the purpose of file transfer. Other such combinations may be established by other standards or by the registration of application contexts. This parameter is not used to set any activity attribute.

13.11 Activity Identifier

The activity identifier parameter is only visible in the internal file service, and then only if the recovery functional unit has been negotiated on F-INITIALIZE (see 14.1.2.12). It gives an unambiguous identifier for the file activity to be performed within the open regime. A different activity identifier value is allocated to each activity involving a particular pair of initiating and responding entities. This identifier is used in re-establishing the data transfer regime after errors.

An activity identifier may be reused following receipt of an F-CLOSE confirm of an open regime having the same activity identifier. The responder loses all knowledge of an activity identifier on issue of the F-CLOSE response. An F-U-ABORT or F-P-ABORT between these events followed by an F-RECOVER will result in a characteristic error being generated. The value of the activity identifier is an integer. This parameter is not used to set any activity attribute.

13.12 File Access Data Unit Identity

The file access data unit identity specifies the target FADU to which a series of one or more specific file service operations is related. The parameter may take one of the values specified in ISO 8571-2. In addition in access context FL, the file access data unit identity is qualified by an integer level number of the requested DUs, relative to the root of the addressed FADU. This parameter is used to set the current location activity attribute.

NOTE - Further constraints may be imposed by the constraint set, the permitted actions file attribute (see ISO 8571-2) and the access contexts to be used if reading the file.

The FADU addressed by the file access data unit identity depends on the operation to be performed.

a) LOCATE: The FADU identity address the FADU to be located.

b) READ: The FADU identity addresses the FADU(s) to be read. If read in an access context returning structuring information (HA, HN, FA, FL and FS), the node descriptor returned contains the node name, which need not necessarily be identical to the FADU identity.

c) INSERT, REPLACE and EXTEND: The location of each node to be inserted, replaced or extended is

determined in a way specified for that action in the constraint set in use (see ISO 8571-2), based on either:

1) the FADU identity, if no structuring information is transmitted (i.e. only the data unit contents are transferred);

2) the FADU identity and the node name in the first node descriptor of each FADU if structuring information is transmitted.

If performed in an access context requiring structuring information (HA, FA and FS) the node name in the first node descriptor of an FADU transmitted in the transferred data is identical to the node name stored in the virtual filestore, otherwise the operation will fail.

d) ERASE: The FADU identity addresses the FADU to be erased.

13.13 Diagnostic

The diagnostic parameter conveys detailed information on the failure of a requested action. The diagnostic parameter amplifies the information given in the action result parameter (see 13.2). Three diagnostic types are defined to distinguish between the provision of information to qualify a successful action, supplemental detail to a transient error (in the IFS only) and supplemental detail to a permanent error. The possible values for the diagnostic parameter are defined in annex A. This parameter is not used to set any activity attribute.

14 FTAM regime control

The services described in this clause control the FTAM regime characterizing the file service association. There are groups of primitives concerned with regime establishment, orderly regime termination and abrupt regime termination.

In the following tables the following definitions apply:

Mandatory	The parameter is present on all occurrences of the primitives.
Optional	If the option is present on the request it is also present on the indication. If the parameter is present on the response it is also present on the confirm.
Dependent	If the response and confirm are dependent they are both present if, and only if, the parameter is present on the request and indication.
Conditional	Indicates that the availability of the parameter is conditional on a successful fulfillment or negotiation of another parameter or functional unit in a previous primitive. If the condition is fulfilled then the presence of the parameter is mandatory.

14.1 FTAM regime establishment service

14.1.1 Function

The establishment of an FTAM regime is the first phase in any instance of file activity. The F-INITIALIZE request primitive is issued by the file transfer initiator.

The F-INITIALIZE primitives are used only to create an FTAM regime and may not be issued within an existing regime.

14.1.2 Types of primitives and parameters

Table 11 indicates the types of primitives and parameters needed for FTAM regime establishment.

14.1.2.1 State result

The state result, defined in 13.1, indicates whether the FTAM regime has been established.

14.1.2.2 Action result

The Action result parameter is defined in 13.2.

14.1.2.3 Called Application Title

The called application title is the title used to identify the filestore . The value is an application entity title. This parameter is not used to set any activity attribute.

14.1.2.4 Calling Application Title

The calling application title is the title of the initiating FTAM entity. The value is an application entity title. This parameter is used to set the current calling application entity title activity attribute.

14.1.2.5 Responding Application Title

The responding application title is that title returned by the responder and that which will be used for re-establishing the association after failure. The value is an application entity title. This parameter is used to set the current responding application entity title activity attribute at the initiator.

NOTE - Application Entity Titles are used by the file service users to provide one another with naming information which is stable on a longer timescale than any particular FTAM regime. They are used, for example, in error recovery and access control.

14.1.2.6 Called Presentation Address

The called presentation address is the address used by the calling service user to identify the PSAP to which the association is to be established. The value is a PSAP-address. This parameter is not used to set any activity attribute

14.1.2.7 Calling Presentation Address

The calling presentation address is the address from which the association is established. The value is a PSAP-address. This parameter is not used to set any activity attribute.

14.1.2.8 Responding Presentation Address

The responding presentation address is the address which is used in re-establishing the association after failure. The value is a PSAP-address. This parameter is not used to set any activity attribute.

NOTES

1 The responding address is not necessarily textually identical to the called address. It may differ, for example, if generic addressing or redirection are in use.

2 Presentation Addresses are used by the service users to specify the addressing requirements of the particular FTAM regime being initialized. The relation between a presentation Address and an Application Entity Title is determined by the specification of the Application Entity as a whole.

14.1.2.9 Presentation Context Management

The requested presentation context management parameter indicates whether or not the Context management functional unit in the Presentation service is to be used during the FTAM open and recovery procedures. The value is a boolean. The responder may refuse an initiator proposal to use presentation context management, even if the facility is available on the underlying presentation connection, and still establish an FTAM regime. The responder may not indicate use of presentation context management if the initiator has not done so. This parameter is not used to set any activity attribute.

14.1.2.10 Application Context Name

The application context name parameter carries a name used to represent the properties of the association as a whole. The initiator proposes a name which may be accepted, and returned by the responder, or the responder may return a different name. In either case the name returned by the responder is the application context name applicable to the established association.

NOTE - This parameter is in general specific to an application. However ISO 8571-4 defines a name for use when the primary intent is to transfer files as an activity in its own right.

14.1.2.11 Service Class

The service class parameter on the request and indication takes one of the values defined in 10.1. This represents the capability of the initiator. (see 8.2 and 10.1). The service class set proposed by the initiator is reduced, by the responder, to a single class (see 10.1) which is returned on the response and confirm primitives. If no service class is acceptable to the responder the FTAM regime

Table 11 — F-INITIALIZE parameters

Parameter	F-INITIALIZE request	F-INITIALIZE indication	F-INITIALIZE response	F-INITIALIZE confirm
State result			Mandatory	Mandatory
Action result			Mandatory	Mandatory
Called Application Title	Mandatory	Mandatory (=)		
Calling Application Title	Mandatory	Mandatory (=)		
Responding Application Title			Mandatory	Mandatory (=)
Called Presentation Address	Mandatory	Mandatory (=)		
Calling Presentation Address	Mandatory	Mandatory (=)		
Responding Presentation Address			Mandatory	Mandatory (=)
Presentation Context Management	Mandatory	Mandatory	Mandatory	Mandatory (=)
Application Context Name	Optional	Optional (=)	Optional	Optional (=)
Service Class	Mandatory	Mandatory	Mandatory	Mandatory (=)
Functional Units	Mandatory	Mandatory (=)	Mandatory	Mandatory (=)
Attribute Groups	Mandatory	Mandatory	Mandatory	Mandatory (=)
Shared ASE Information	Optional	Optional (=)	Optional	Optional (=)
FTAM Quality of Service	Mandatory	Mandatory	Mandatory	Mandatory
Communications Quality of Service	Mandatory	Mandatory	Mandatory	Mandatory (=)
Contents Type List	Conditional	Conditional	Dependent	Dependent (=)
Initiator Identity	Optional	Optional (=)		
Account	Optional	Optional (=)		
Filestore Password	Optional	Optional (=)		
Diagnostic			Optional	Optional
Additional Parameters in the Internal File Service				
Checkpoint Window	Conditional	Conditional (=)	Conditional	Conditional (=)

(=) signifies that the value is not modified by the file service provider.

establishment fails. This parameter is not used to set any activity attributes.

14.1.2.12 Functional Units

The requested functional units parameter negotiates the set of file service functional units, excluding the kernel, to be available from the negotiated service class on the application association (see 8.1 and 10.3). On the request and indication primitives the parameter carries the full functional unit capability of the initiator. On the response and confirm primitives it carries all the functional units to be available on the association. When the required service class permits, the set may contain the optional functional units "read", "write", "file access", "limited file management", "enhanced file management", "grouping" and "FADU locking". For use by the internal file service, if the service class permits, it may also contain the "recovery" and "restart" functional units. The recovery and restart functional units are only visible in the IFS and are thus never explicitly requested in the external file service, they are indirectly requested via the FTAM quality of service parameter (see 10.2). This parameter is not used to set any activity attributes.

14.1.2.13 Attribute Groups

The requested attribute group parameter negotiates the set of optional file attribute groups to be available on the application association. Specification of the attribute groups is defined in ISO 8571-2. The set may be empty or it may contain some combination of the attribute groups "storage", "security" or "private" (see ISO 8571-2). The responder may reduce the attribute groups proposed by the initiator,

within the limits of ISO 8571-2. The kernel attribute group is not proposed or negotiated, hence negotiation of the attribute groups will not prevent the establishment of an FTAM regime. This parameter is not used to set any activity attributes.

14.1.2.14 Shared ASE information

The shared ASE information parameter is defined in 13.10.

14.1.2.15 FTAM Quality of Service

The FTAM quality of service parameter is used to convey information relating to the error susceptibility of the external file service user. The detailed negotiation rules are specified in clause 10. The values of this parameter indicate the application is susceptible to errors of one of the following types:

a) not susceptible to errors. No error recovery to be provided.

b) errors which damage the data transfer regime.

c) errors which damage the open or data transfer regimes.

d) errors which damage the select, open or data transfer regimes, or errors which lose the association.

This parameter is not used to set any activity attributes.

14.1.2.16 Communication Quality of Service

The requested communication quality of service parameter conveys the quality of service to be negotiated on the association. On the request primitive it indicates the quality of service requested by the initiator. On the indication primitive it indicates the quality of service requested, reduced as necessary to signal that achievable, by the service provider. On the response and confirm primitives it indicates the quality of service achieved. The values taken by the Communication Quality of Service parameter are as defined in ISO 8649. This parameter is not used to set any activity attributes.

NOTES

1 This parameter is referenced, via the ASE association control service definition (ISO 8649), to the Presentation Service Definition (ISO 8822), then to the Session Service Definition (ISO 8326).

2 As an International Standard ISO 8571 has no intrinsic basis on which to select or manipulate values of any aspect of communications quality of service (CQoS).

14.1.2.17 Contents Type List

The requested contents type list parameter carries a list of document types and/or abstract syntaxes. This parameter is mandatory (in the transfer class, the transfer and management class and the access class), if the presentation context management functional unit is not being negotiated. This parameter allows the establishment of the necessary presentation contexts at the time of establishing the FTAM regime. The initiator proposes a list of elements which are each either document type names or abstract syntax names from which a unique list of the required abstract syntaxes is constructed. The latter is used to construct the presentation context definition list parameter in the ACSE A-ASSOCIATE service primitive. The service provider reduces the value of the contents type list to remove any contents types which require abstract syntaxes rejected by the presentation service provider. The responder further reduces the list to remove any contents types it will not support, and any consequential reduction in the required abstract syntax list is made. The Contents Type List response returns the list of those agreed document type names and abstract syntax names for which there is a supported abstract syntax and so is used to build the presentation context definition result list by the service provider. This parameter is not used to set any activity attribute.

14.1.2.18 Initiator Identity

The initiator identity parameter identifies the calling user. The value of the optional initiator identity parameter is a GraphicString. The range of values the parameter may take is equal to that defined for the current initiator identity activity attribute. If the parameter is not present, the current initiator identity activity attribute remains unset. If this parameter value, or its omission, is unacceptable to the responder, the responder generates a state result indicating failure to establish the FTAM regime and/or an action result indicating permanent error with an optional diagnostic detailing the reason. This parameter is used to set the current initiator identity activity attribute.

14.1.2.19 Account

The account parameter is defined in 13.3. The account given is charged for all costs incurred by the FTAM regime, on the application association; this excludes nested costs associated with file selection regimes where an explicit overriding account parameter is given when the regime is established. The value of this optional parameter is a GraphicString. If this parameter value, or its omission, is unacceptable to the responder, the responder generates a state result indicating failure to establish the FTAM regime and/or an action result indicating permanent error with an optional diagnostic detailing the reason. This parameter is used to set the current account activity attribute.

14.1.2.20 Filestore Password

The filestore password parameter conveys a password which is used to authenticate the initiator to the responder. This parameter is a GraphicString or an OCTET STRING used by the responder to authenticate the initiator identity parameter. If this parameter value, or its omission, is unacceptable to the responder, the responder generates a state result indicating failure to establish the FTAM regime and/or an action result indicating permanent error with an optional diagnostic detailing the reason. This parameter is not used to set any activity attributes.

14.1.2.21 Diagnostic

The diagnostic parameter is defined in 13.13.

14.1.2.22 Checkpoint Window

The requested checkpoint window parameter indicates, for each direction of transmission, the maximum number of checkpoints which may remain unacknowledged. This parameter is conditional upon the recovery or restart functional units having been selected in the IFS, in which case it is mandatory. Checkpoints are only inserted by the sender. Values of this parameter never cause diagnostics on F-INITIALIZE, but may be the reason for subsequent termination. The continued progress of the service is only guaranteed if the entity acting as receiver gives acknowledgements within this limit. The window size is stated independently by each entity as the maximum value for when that entity is the sending entity. There is no negotiation.

Table 12 — F-TERMINATE parameters

Parameter	F-TERMINATE request	F-TERMINATE indication	F-TERMINATE response	F-TERMINATE confirm
Shared ASE Information	Optional	Optional (=)	Optional	Optional (=)
Charging			Conditional	Conditional

(=) signifies that the value is not modified by the file service provider

The stated value from each entity, as the sending entity, is maintained by the corresponding entity for use when it is the receiving entity. The values for each direction of data transfer are not necessarily the same. The parameter is an integer. The parameter is not used to set any activity attributes.

14.2 FTAM regime termination service (orderly)

14.2.1 Function

An FTAM regime may be terminated by an exchange of F-TERMINATE primitives. This primitive will only be issued when there are no actions in progress. The F-TERMINATE request primitive can be issued by the file transfer initiator (the issuer of the F-INITIALIZE request) at any time after the receipt of an F-INITIALIZE confirm primitive, providing no file is selected. The issue of an F-TERMINATE request does not imply the success of any previous activity. Indications of success or failure are given on the completion of each activity.

14.2.2 Types of primitives and parameters

Table 12 indicates the types of primitive and the parameters needed for orderly FTAM regime termination.

14.2.2.1 Shared ASE Information

The shared ASE information parameter is defined in 13.10.

14.2.2.2 Charging

The charging parameter is defined in 13.4. The parameter reflects costs incurred during the FTAM regime, on the basis of the account parameter quoted when the FTAM regime was established. This process excludes charges assigned to accounts within nested regimes. The presence of this parameter is conditional on the account parameter having been provided by the initiator on the F-INITIALIZE primitive.

14.3 FTAM regime termination service (abrupt)

14.3.1 Function

Either external file service user may issue an F-U-ABORT request primitive at any time after an F-INITIALIZE request primitive has been issued, or an F-INITIALIZE indication primitive has been received. The file service provider may issue an F-P-ABORT primitive at any time after an F-INITIALIZE request primitive has been received, or an F-INITIALIZE indication primitive issued. The F-U-ABORT or F-P-ABORT primitives terminate the FTAM regime unconditionally, abandoning any file activity that was in progress and leaving the selected file in an undefined state. If error recovery is to be performed, the responsibility for initiating the recovery lies with the initiator. Once an F-U-ABORT request or F-P-ABORT request primitive has been issued, the FTAM regime is terminated; the request cannot be rejected.

NOTE - In the external file service data transfer phase the use of F-CANCEL is preferred over that of F-U-ABORT. In the external file service if the F-U-ABORT fails to reach the peer entity because of an underlying service failure the issuer of the F-U-ABORT will release locks while the peer entity will receive an F-P-ABORT from the lower layer failure and will retain locks pending a recovery action. No mechanisms are defined in ISO 8571 to resolve this difference of view of the file status.

The filestore provider performs the local close file and deselect file actions on receipt of an F-U-ABORT indication or an F-P-ABORT indication if the file is open, and the local deselect file action if the file is closed but selected.

NOTE - Abrupt termination may lose information about charges which are levied. The initiator will not know which unconfirmed services have affected the file. An internal service abort for a transient error will not be visible in the external service, and concurrency controls will stay in place while recovery is in progress. An abort in the external service will leave the initiator in doubt about the state of the file, with concurrency controls released.

14.3.2 Types of primitives and parameters

Tables 13 and 14 indicate the types of primitives and parameters needed for abrupt FTAM regime termination.

Table 13 — F-U-ABORT parameters

Parameter	F-U-ABORT request	F-U-ABORT indication
Action Result	Mandatory	Mandatory
Diagnostic	Optional	Optional

Table 14 — F-P-ABORT parameters

Parameter	F-P-ABORT indication
Action Result	Mandatory
Diagnostic	Optional

14.3.2.1 Action Result

The action result parameter is defined in 13.2.

14.3.2.2 Diagnostic

The diagnostic parameter is defined in 13.13.

14.3.3 Sequence of primitives

The sequence of events in a user initiated abrupt termination is defined in figure 5.

The sequence in a service provider initiated abrupt termination is defined in the time sequence diagram shown in figure 6.

Collision of an F-P-ABORT and an F-U-ABORT can result in the loss of the F-U-ABORT indication shown in figure 7.

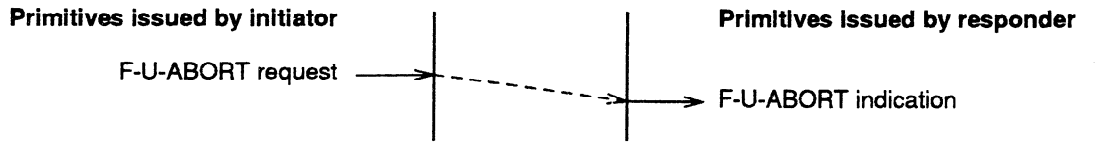

Figure 5 — F-U-ABORT service

Figure 6 — F-P-ABORT service

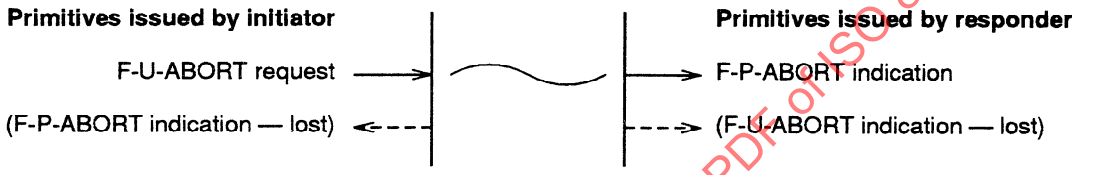

Figure 7 — F-P-ABORT collision

15 File selection regime control

These services control the regime which binds an identified file to an FTAM regime. Different services are used depending on whether the file exists before the selection or not, and on whether it is to exist after the selection or not.

15.1 File selection service

15.1.1 Function

This service selects a file which already exists, by specifying a filename attribute to identify the file unambiguously. These primitives may only be issued if there is no current file selection within the current FTAM regime. The filestore provider performs the select file action after receiving the F-SELECT indication and before issuing an F-SELECT response primitive with an action result indicating success.

15.1.2 Types of primitives and parameters

Table 15 indicates the types of primitives and the parameters needed for file selection.

15.1.2.1 State result

The state result parameter is defined in 13.1. The parameter indicates whether the select regime has been successfully established.

15.1.2.2 Action result

The action result parameter is defined in 13.2.

15.1.2.3 Attributes

The attributes parameter is defined in 13.5. On F-SELECT the parameter carries only a single file name. The file name is the only file attribute used to identify the file in the file

Table 15 — F-SELECT parameters

Parameter	F-SELECT request	F-SELECT indication	F-SELECT response	F-SELECT confirm
State result			Mandatory	Mandatory
Action result			Mandatory	Mandatory
Attributes	Mandatory	Mandatory (=)	Mandatory	Mandatory (=)
Requested Access	Mandatory	Mandatory (=)		
Access Passwords	Conditional	Conditional (=)		
Concurrency Control	Optional	Optional (=)		
Shared ASE Information	Optional	Optional (=)	Optional	Optional (=)
Account	Optional	Optional (=)		
Diagnostic			Optional	Optional

(=) signifies that the value is not modified by the file service provider.

selection process. On the F-SELECT response and confirm primitives only the filename is returned which may not be identical to that issued on the request and indication.

15.1.2.4 Requested Access

The requested access parameter is defined in 13.6. If the requested access is not permitted by the responder the select regime is not established.

15.1.2.5 Access Passwords

The access passwords parameter is defined in 13.7. It is conditional on the security group being selected. If the members of the set of access passwords do not match any of the non-empty password strings in the access control condition (see ISO 8571-2), the select regime is not established. This parameter is used to set the current access passwords activity attribute.

15.1.2.6 Concurrency Control

The concurrency control parameter is defined in 13.8. If the required concurrency control is not available the select regime is not established.

15.1.2.7 Shared ASE Information

The shared ASE information parameter is defined in 13.10.

15.1.2.8 Account

The account parameter is defined in 13.3. The value given overrides any value previously set by the F-INITIALIZE request for the duration of this select regime.

15.1.2.9 Diagnostic

The diagnostic parameter is defined in 13.13.

15.2 File deselection service

15.2.1 Function

The F-DESELECT service releases the binding between the file select regime and the file. The file continues to exist and is available for subsequent selection. The primitives may only be used while a file is selected. The file select regime is always terminated following an F-DESELECT response or confirm for all values of the action result parameter.

The filestore provider performs the deselect file action after receiving the F-DESELECT indication primitive, and before issuing the F-DESELECT response primitive. In the external service all concurrency control is released when the file is deselected, except when performed within a CCR atomic action. The current account activity attribute is reset to that of the FTAM regime.

15.2.2 Types of primitives and parameters

Table 16 indicates the types of primitives and the parameters needed for file deselection.

15.2.2.1 Action result

The action result parameter is defined in 13.2. The selection regime is terminated whatever the value of the action result parameter.

15.2.2.2 Charging

The charging parameter is defined in 13.4. Charges are only reported on deselection against an overriding account set up at file selection.

15.2.2.3 Shared ASE Information

The shared ASE information parameter is defined in 13.10.

15.2.2.4 Diagnostic

The diagnostic parameter is defined in 13.13.

15.3 File creation service

15.3.1 Function

The F-CREATE service generally causes a file to be created and establishes a selection of the newly created file. If the identified file exists then, depending on the override parameter, the request may be rejected, or the existing file may be selected, possibly after recreation. The create service may only be used if there is no currently selected file.

The filestore provider performs the create file action after receiving the F-CREATE indication primitive, and before issuing the F-CREATE response primitive with action result parameters indicating success. If the override parameter is appropriately set and the named file already exists, the responder performs the select action instead of the create action.

15.3.2 Types of primitives and parameters

Table 17 indicates the types of primitives and the parameters needed for file creation.

15.3.2.1 State result

The state result parameter is defined in 13.1. The parameter indicates the select regime is established, either for an existing file or a newly created one, or that the select has failed.

15.3.2.2 Action result

The action result parameter is defined in 13.2. The parameter indicates whether the create was successful or not. It is possible for the create to be successful and the

Table 16 — F-DESELECT parameters

Parameter	F-DESELECT request	F-DESELECT indication	F-DESELECT response	F-DESELECT confirm
Action result			Mandatory	Mandatory
Charging			Conditional	Conditional
Shared ASE Information	Optional	Optional (=)	Optional	Optional (=)
Diagnostic			Optional	Optional

(=) signifies that the value is not modified by the file service provider.

Table 17 — F-CREATE parameters

Parameter	F-CREATE request	F-CREATE indication	F-CREATE response	F-CREATE confirm
State result			Mandatory	Mandatory
Action result			Mandatory	Mandatory
Override	Mandatory	Mandatory (=)		
Initial Attributes	Mandatory	Mandatory (=)	Mandatory	Mandatory (=)
Create Password	Optional	Optional (=)		
Requested Access	Mandatory	Mandatory (=)		
Access Passwords	Conditional	Conditional (=)		
Concurrency Control	Conditional	Conditional (=)		
Shared ASE Information	Optional	Optional (=)	Optional	Optional (=)
Account	Optional	Optional (=)		
Diagnostic			Optional	Optional

(=) signifies that the value is not modified by the file service provider.

subsequent select to fail, in which case a new file is known to the filestore but it is not currently selected.

15.3.2.3 Override

The override parameter defines the action to be taken if the named file already exists. The values of the parameter are:

a) fail the create action if the file already exists;

b) select file if it already exists;

c) delete file if it already exists and create a new file using the old file's attributes (effectively delete the contents of the existing file and select it);

d) delete file if it already exists and create a new file using the initial attributes provided on the F-CREATE primitive.

The values (c) and (d) are dependent upon the provision of a delete password in the access passwords parameter where required by the access control attribute of the existing file. This parameter is not used to set any activity attributes.

15.3.2.4 Initial Attributes

The initial attributes parameter has the same form as the attributes parameter, which is defined in 13.5. For those attribute groups negotiated on F-INITIALIZE, the attribute values given, are associated with the newly created file. If, for any particular attribute a value is not given, a default is determined locally by the responder; this value may be "no value available". For attributes other than filename and permitted actions, the responder may change a value proposed by the initiator to "no value available" but may not assign any different value to it. The responder does report any local modification to the values of the filename or permitted actions attributes. In addition to the attributes in the attribute groups negotiated on F-INITIALIZE the attributes in the kernel attribute group are also associated with the newly created file. This parameter is used to set the current access passwords activity attribute if a new file has been created.

15.3.2.5 Create Password

The create password is an optional parameter which may be required to establish that the user identified by the initiator identity parameter of the FTAM regime establishment service has permission to create files in the current filestore. The value is an OCTET STRING or GraphicString. This parameter is not used to set any activity attributes.

15.3.2.6 Requested Access

The requested access parameter is defined in 13.6. The parameter is used to establish the access available in the select regime following the create action. The requested access during the select regime need not be the full capability of the file. Hence the access requested must be a subset of those capabilities established with the initial attributes parameter. The established capabilities include all responder provided defaults which are within the kernel attribute group or the other attribute groups negotiated at F-INITIALIZE.

If the requested access parameter is incompatible with the newly created file, the select will fail even though the file was created.

15.3.2.7 Access Passwords

The access passwords parameter is defined in 13.7. The access passwords are used to authenticate the requested access and, if appropriate, the requested concurrency control for the select regime. If the override parameter creates a new file or recreates a file with new attributes the access password parameter is only used to verify any delete actions; it is not used to verify other actions on the selected file since the appropriate passwords are included in the initial attributes parameter. This parameter is used to set the current access passwords activity attribute if the override parameter resulted in the selection of an existing file.

15.3.2.8 Concurrency Control

The concurrency control parameter is defined in 13.8.

15.3.2.9 Shared ASE Information

The shared ASE information parameter is defined in 13.10.

15.3.2.10 Account

The account parameter is defined in 13.3. The value given overrides any value previously set on the F-INITIALIZE request for the duration of this selection.

15.3.2.11 Diagnostic

The diagnostic parameter is defined in 13.13.

15.4 File deletion service

15.4.1 Function

The F-DELETE service releases an existing file select regime in such a way that the selected file ceases to exist, and is not available for reselection. The primitives may only be issued while a file is selected. The selected regime is always released following an F-DELETE response or confirm, for all possible values of the action result parameter.

The filestore provider performs the delete file action after receiving the F-DELETE indication, and before issuing the F-DELETE response primitive, with an action result parameter of success. The delete file action can be performed only if the initiating entity has the "delete file" access control permission (see 13.6). This access control permission is established by the requested access, concurrency control and access passwords parameters on the F-SELECT or F-CREATE primitives which established the file selection regime.

15.4.2 Types of primitives and parameters

Table 18 indicates the types of primitives and the parameters needed for file deletion.

15.4.2.1 Action Result

The action result parameter, defined in 13.2, indicates success or failure of the deletion action. The file is always deselected whatever the value of the action result parameter.

15.4.2.2 Shared ASE Information

The shared ASE information parameter is defined in 13.10.

15.4.2.3 Charging

The charging parameter is defined in 13.4. Charges are only reported on deselection against an overriding account set up at file selection.

15.4.2.4 Diagnostic

The diagnostic parameter is defined in 13.13.

Table 18 — F-DELETE parameters

Parameter	F-DELETE request	F-DELETE indication	F-DELETE response	F-DELETE confirm
Action Result			Mandatory	Mandatory
Shared ASE Information	Optional	Optional (=)	Optional	Optional (=)
Charging			Conditional	Conditional
Diagnostic			Optional	Optional

(=) signifies that the value is not modified by the file service provider.

16 File management

These services provide file management capability to the file service user. They are not concerned with the establishment of regimes. The services provide facilities for interrogation and modification, where permitted, of file attributes.

16.1 Read attribute service

16.1.1 Function

The F-READ-ATTRIB service interrogates the file attributes of the selected file. The request and indication convey a list of the file attribute names for which values are to be read. The response and confirm then convey the corresponding returned values.

The filestore provider performs the read attribute action after receiving the F-READ-ATTRIB indication, and before issuing the F-READ-ATTRIB response primitive.

16.1.2 Types of primitives and parameters

Table 19 indicates the types of primitive and the parameters needed to read file attributes.

16.1.2.1 Action Result

The action result parameter is defined in 13.2.

16.1.2.2 Attribute Names

The attribute names parameter indicates which file attributes from the kernel or negotiated attribute groups, given in the virtual filestore definition, are to be read. The parameter is a list, each element of which names a file attribute defined in ISO 8571-2. This parameter is not used to set any activity attributes.

16.1.2.3 Attributes

The attribute parameter is defined in 13.5. The presence of the attributes parameter is conditional on the inclusion in the attribute names parameter of at least one attribute with a returnable value. Values for attributes not specifically requested are not returned. Values for attributes not in the kernel attribute group or a negotiated attribute group are not returned. The value of password attributes are not returned. The responder may return "no value available" for partially supported attributes. ISO 8571-2 details the attributes which are and are not returned.

16.1.2.4 Diagnostic

The diagnostic parameter is defined in 13.13.

16.2 Change attribute service

16.2.1 Function

The F-CHANGE-ATTRIB service modifies the file attributes of the selected file. The primitives convey a list of file attribute names and values.

The filestore provider performs the change attribute action and sets the two file attributes, date and time of last attribute modification and identity of last attribute modifier, after receiving the F-CHANGE-ATTRIB indication, and before issuing the F-CHANGE-ATTRIB response primitive. If any change causes an action result other than success to be generated, none of the requested changes are made; an "unsuccessful" action result parameter is returned.

16.2.2 Types of primitives and parameters

Table 20 indicates the types of primitive and the parameters needed to change file attributes.

16.2.2.1 Action Result

The action result parameter is defined in 13.2. An unsuccessful action result indicates none of the changes has been performed.

16.2.2.2 Attributes

The attribute parameter, defined in 13.5, indicates on the request and indication primitives which file attributes given in the virtual filestore definition are to be changed and what the new values are to be. The parameter is a list, each element of which names a file attribute defined in ISO 8571-2 and provides a new value for it.

The presence of the attributes parameter on the response and confirm is conditional on the inclusion in the attributes parameter of at least one attribute name and value. Values for attributes not specifically requested are not changed or returned. Values for attributes not in the kernel or negotiated attribute group are not changed or returned. ISO 8571-2 defines the actions permitted on attributes.

16.2.2.3 Diagnostic

The diagnostic parameter is defined in 13.13.

17 File open regime control

These services establish or release the file open regime within which file data can be transferred. This regime establishes processing mode, presentation contexts, and concurrency control for the data transfer activity which is to be performed.

Table 19 — F-READ-ATTRIB parameters

Parameter	F-READ-ATTRIB request	F-READ-ATTRIB indication	F-READ-ATTRIB response	F-READ-ATTRIB confirm
Action Result			Mandatory	Mandatory
Attribute Names	Mandatory	Mandatory (=)		
Attributes			Conditional	Conditional (=)
Diagnostic			Optional	Optional

(=) signifies that the value is not modified by the file service provider.

Table 20 — F-CHANGE-ATTRIB parameters

Parameter	F-CHANGE-ATTRIB request	F-CHANGE-ATTRIB indication	F-CHANGE-ATTRIB response	F-CHANGE-ATTRIB confirm
Action Result			Mandatory	Mandatory
Attributes	Mandatory	Mandatory (=)	Conditional	Conditional (=)
Diagnostic			Optional	Optional

(=) signifies that the value is not modified by the file service provider.

17.1 File open service

17.1.1 Function

The F-OPEN service establishes an open regime. The regime establishment determines the abstract syntaxes and document types necessary to access the file and establish that the association has the capability of communicating them. It also establishes concurrency and possible processing modes.

The open service may only be used if a file has been selected, and no open regime currently exists.

The filestore entity performs the open file action after receiving the F-OPEN indication primitive, and before issuing the F-OPEN response primitive with an action result parameter indicating success.

The current location activity attribute is set by this service.

17.1.2 Types of primitives and parameters

Table 21 indicates the types of primitive and the parameters needed for file opening.

17.1.2.1 State Result

The state result parameter, defined in 13.1, indicates whether the open regime has been established or not.

17.1.2.2 Action Result

The action result parameter is defined in 13.2.

17.1.2.3 Processing Mode

The processing mode parameter establishes a subset of the valid actions negotiated in the select regime, for use within the open regime being established. The processing mode parameter indicates the valid actions to be performed as a result of access control and bulk data transfer requests; this determines the filestore actions which the responding entity can perform. The parameter value indicates whether read, insert, replace, extend or erase are to be permitted. The parameter is compared with the current access request activity attribute and the access constraints determined by the selected view of the document type referenced by the Contents Type parameter (see 17.1.2.4). If the requested processing mode is not available for the filestore, or the constraints set referenced by the contents type may potentially be violated, the open regime establishment fails. This parameter is used to set the current processing mode activity attribute.

17.1.2.4 Contents Type

The contents type parameter identifies the abstract data type of the contents of the file, including the structuring information. The contents type is a scalar set at file creation time. Its value is either a document type with optional parameters or an abstract syntax together with a constraint set name.

The request and indication primitives convey either the required contents type or indicate "contents type unknown". For the case of a document type the parameters may be present or they may be omitted if they are unknown.

Table 21 — F-OPEN parameters

Parameter	F-OPEN request	F-OPEN indication	F-OPEN response	F-OPEN confirm
State Result			Mandatory	Mandatory
Action Result			Mandatory	Mandatory
Processing Mode	Mandatory	Mandatory (=)		
Contents Type	Mandatory	Mandatory (=)	Mandatory	Mandatory (=)
Concurrency Control	Optional	Optional (=)	Dependent	Dependent (=)
Shared ASE Information	Optional	Optional (=)	Optional	Optional (=)
Enable FADU Locking	Conditional	Conditional (=)		
Diagnostic			Optional	Optional
Additional Parameters in the Internal File Service				
Activity Identifier	Conditional	Conditional (=)		
Recovery Mode	Conditional	Conditional	Conditional	Conditional (=)

(=) signifies that the value is not modified by the file service provider.

The response and confirm always indicate the valid file contents type. The initiator indicating on the request and indication primitives that the contents type is unknown does not prevent the file being opened. If the open regime establishment is successful the contents type parameter, on the response and confirm primitives, conveys the contents type file attribute from the file or, if a simplification or relaxation was requested that is returned. The initiator is then able to decide on further action.

If the contents type parameter on the request and indication primitives is a document type name it may take one of three forms:

a) a document type name only

b) a document type name with a null list of parameters

c) a document type name with a list of parameters.

the open only succeeds if:

d) the proposed document type name is identical to either the document type name in the contents type file attribute, or, if only read actions are allowed by the processing mode parameter, a simplification or relaxation defined in the document type referenced in the contents type file attribute;

and

e) if parameters were provided, they match exactly the parameters of the contents type file attribute. A null list of parameters match any parameters in the contents type file attribute.

If the contents type parameter on the request and indication primitives is a constraint set name and abstract syntax name pair then the open only succeeds if the pair exactly match the pair in the contents type file attribute. This parameter is used to set the active contents type activity attribute.

17.1.2.5 Concurrency Control

The concurrency control parameter, defined in 13.8, allows the concurrency control information, initially established at file creation or selection (see 15.3 and 15.1), to be modified when the file is opened. The concurrency control restrictions, set by the initiator, may be made more restrictive than that specified on the selection, but are not made less restrictive.

The F-OPEN fails if a file service user, having established shared access for an action on file selection, requests a more restrictive lock on F-OPEN and that lock is unavailable.

The sequence of locks, in ascending order of restrictiveness, is not required, shared, exclusive and no access.

If the FADU locking functional unit has been negotiated, and the Enable FADU locking parameter is set on (see 17.1.2.7), the concurrency controls established at file select and/or open are only in place during a data transfer regime. Outside a data transfer regime, an FADU is in the "not required" state unless explicitly locked by the FADU locking mechanism.

17.1.2.6 Shared ASE Information

The shared ASE information parameter is defined in 13.10.

17.1.2.7 Enable FADU Locking

Use of the enable FADU locking parameter modifies the semantics of concurrency control. The enable FADU locking parameter is available only if the FADU locking functional unit was negotiated on the FTAM regime establishment, in which case it is mandatory. The enable FADU locking parameter indicates whether locking is to be on a per FADU basis, as opposed to a file basis. The value of this parameter is a boolean which indicates whether FADU locking, in addition to whole file concurrency control locking, is required.

The enable FADU locking parameter is only available if:

a) the storage attribute group has been negotiated; and

b) the concurrency control parameter is present.

If FADU locking is selected the open will not fail for concurrency control reasons, however subsequent attempts to set FADU locks may fail. If FADU locking is not enabled the open may fail as described in 17.1.2.5. This parameter is used to set the current locking style activity attribute.

17.1.2.8 Diagnostic

The diagnostic parameter is defined in 13.13.

17.1.2.9 Activity Identifier

The activity identifier parameter is defined in 13.11; it is mandatory in the internal file service if the recovery functional unit has been negotiated.

17.1.2.10 Recovery Mode

The recovery mode parameter in the internal service indicates what error recovery facilities are to be available during the current open regime, and at what points in the file, data transfer can be resumed. The value is one of "none", "at start of file", or "at any active checkpoint". The recovery mode parameter is conditional and is only available in the internal file service, and then is mandatory if the restart or recovery functional units were negotiated on F-INITIALIZE. The value "none" allows the recovery facility to be made unavailable for a specific open regime. The recovery mode is negotiated by the initiator proposing one value and the responder returning that value, as agreement to it, or a value lower if the responder is not prepared to support the initiator's value. The value returned by the responder is that established for the open regime. The decreasing order of value is

a) any active checkpoint;

b) at the start of file;

c) none.

Whatever the value of this parameter it does not prevent the establishment of an open regime. If the value none is agreed then restart and recovery within the open regime is invalid. This parameter is not used to set any activity attribute.

17.2 File close service

17.2.1 Function

The F-CLOSE service releases an existing file open regime. The file open regime is always terminated following an F-CLOSE response and confirm primitive for all possible values of the action result parameter.

Table 22 — F-CLOSE

Parameter	F-CLOSE request	F-CLOSE indication	F-CLOSE response	F-CLOSE confirm
Action Result	Optional	Optional (=)	Mandatory	Mandatory
Shared ASE Information	Optional	Optional (=)	Optional	Optional (=)
Diagnostic	Optional	Optional (=)	Optional	Optional

(=) signifies that the value is not modified by the file service provider.

The filestore provider performs the close file action after receiving the F-CLOSE indication primitive, and before issuing the F-CLOSE response primitive.

The file attributes in the storage attribute group are updated as follows, dependent on the processing mode:

a) For read mode, the date and time of last read access and identity of last reader.

b) For insert, replace, extend and erase modes the date and time of last modification and the identity of last modifier.

17.2.2 Types of primitives and parameters

Table 22 indicates the types of primitives and the parameters needed for terminating a file open regime.

17.2.2.1 Action Result

The action result parameter is defined in 13.2. The open regime is terminated notwithstanding the value of the action result or diagnostic parameter. The action result parameter, and the associated diagnostic parameter, play different roles in the two file services. In the external file service the action result parameter is only present on the response and confirm primitives where it indicates either success or permanent error. In the internal file service the action result parameter may also be present on the request and indication where it is used to signal a transient error between the two error recovery protocol machines.

17.2.2.2 Shared ASE Information

The shared ASE information parameter is defined in 13.10.

17.2.2.3 Diagnostic

The diagnostic parameter is defined in 13.13. It is used to qualify the action result parameter on each primitive and is optionally present in each of the file services when the action result parameter is present. (see 17.2.2.1)

18 Grouping control

The grouping control mechanisms allow a number of regimes to be established or released in one interaction. The service user initiating the group of requests brackets them with grouping control primitives. The responding user analyses and retains its response to each of the requests until the group is complete or a state result failure is detected. It then returns its responses in a corresponding group.

A parameter of the F-BEGIN-GROUP request specifies a threshold value which is the number of primitives to be processed after the F-BEGIN-GROUP, without a "failure" state result (see 13.1), before any action is to be taken. Primitives within the group which do not convey state result parameters are always counted as being "successful". Action result failures do not affect the processing of the group.

If the threshold number of primitives is not processed before a state result failure is detected, a negative response is made for the group. This is signalled by all primitive responses after the begin group response, and before the state result failure response, having their action result parameter indicate "permanent error". The diagnostic on the response of these primitives is set to indicate "transient error" with an error identifier of "subsequent error". If the first response primitive has a state result of failure the diagnostic parameter carries normal detailed failure reasons.

Once a failure response is made on a state result parameter, no response is made to any further primitive within the group, except the F-END-GROUP primitive.

Once the threshold value has been equalled or exceeded by the success count, the successful actions are performed; all remaining primitives are attempted in order until a primitive generating state failure is encountered when no further primitives are processed; or F-END-GROUP is encountered.

18.1 Beginning of grouping service

18.1.1 Function

The F-BEGIN-GROUP service indicates the start of a set of grouped requests, which are to be processed and responded to as a group.

18.1.2 Types of primitives and parameters

Table 23 indicates the types of primitives and the parameters needed for beginning of group.

18.1.2.1 Threshold

The threshold parameter specifies the number of primitives within a group (i.e. after, but not including, the begin group) which are to be analysed without being unsuccessful before any part of the group can succeed. A primitive is categorized as successful or unsuccessful in terms of its state result parameters. Those primitives which only have an action result parameter are to be deemed successful for the purposes of the threshold count. For the groupings specified in the file transfer, file management and file transfer and management classes of ISO 8571 the threshold parameter value equals the number of primitives between (but not including) the begin group and end group primitives. This parameter is not used to set any activity attributes.

Table 23 — F-BEGIN-GROUP parameters

Parameter	F-BEGIN-GROUP request	F-BEGIN-GROUP indication	F-BEGIN-GROUP response	F-BEGIN-GROUP confirm
Threshold	Mandatory	Mandatory (=)		

(=) signifies that the value is not modified by the file service provider.

18.2 End of grouping service

The F-END-GROUP service indicates the end of a set of concatenated requests started by an F-BEGIN-GROUP. There is always an F-END-GROUP to match each F-BEGIN-GROUP, unless the application association is broken by an F-U-ABORT or an F-P-ABORT. The types of primitives defined are F-END-GROUP request, F-END-GROUP indication, F-END-GROUP response and F-END-GROUP confirm. The primitives do not carry parameters.

19 Recovery (Internal service only)

19.1 Regime recovery service

19.1.1 Function

The F-RECOVER service allows the recreation of an open regime after a failure. It allows the filestore entity to access records of suspended activities by reference to a previously established activity identifier. The regimes are recreated on the basis of information, including activity attribute values, stored by the initiator and the responder in secure storage. This body of information is called a docket.

Regime recovery is visible to the internal service user only, and then only if the recovery functional unit was negotiated on the F-INITIALIZE primitives.

NOTE - The mechanisms for recovery in the external service are specified as part of the error recovery protocol specified in ISO 8571-4.

19.1.2 Types of primitives and parameters

Table 24 indicates the types of primitives and the parameters needed for recovery. The parameters are used to identify the activity to be recovered and docket to be associated with the activity. These parameters and the docket information are subjected to the select and open algorithms in order to recover the open regime.

19.1.2.1 State Result

The state result parameter is defined in 13.1.

19.1.2.2 Action Result

The action result parameter is defined in 13.2.

19.1.2.3 Activity Identifier

The activity identifier parameter is defined in 13.11.

19.1.2.4 Bulk Transfer Number

The bulk transfer number parameter indicates, by reference to the number of a bulk data transfer procedure, which bulk data transfer is to be recovered. The recovery position within the bulk data transfer is defined by the recovery point parameter (see 19.1.2.8). The first data transfer within an open regime is numbered 1, and subsequent transfers are consecutively numbered. Read and write transfers are numbered from a single sequence.

19.1.2.5 Requested Access

The requested access parameter is defined in 13.6. The requested access and access password parameters confirm the authenticity of the initiator attempting to recover the activity. If the requested access does not completely satisfy the responder the regime establishment fails with a permanent error.

19.1.2.6 Access Passwords

The access passwords parameter is defined in 13.7.

Table 24 — F-RECOVER parameters

Parameter	F-RECOVER request	F-RECOVER indication	F-RECOVER response	F-RECOVER confirm
State Result			Mandatory	Mandatory
Action Result			Mandatory	Mandatory
Activity Identifier	Mandatory	Mandatory (=)		
Bulk Transfer Number	Mandatory	Mandatory (=)		
Requested Access	Mandatory	Mandatory (=)		
Access Passwords	Optional	Optional (=)		
Contents type			Mandatory	Mandatory
Recovery Point	Conditional	Conditional (=)	Conditional	Conditional (=)
Diagnostic			Optional	Optional

(=) signifies that the value is not modified by the file service provider.

19.1.2.7 Contents Type

The contents type parameter, defined in 17.1.2.4, returns the file contents type name to the initiator.

19.1.2.8 Recovery Point

The recovery point parameter indicates that recovery is to a point before the start of the bulk data transfer (value zero), or to a checkpoint within the bulk data transfer, or to a point following its completion (see note). The recovery point is within the bulk data transfer identified by the bulk data transfer number parameter (see 19.1.2.4). The recovery point is determined by the entity which was receiving data at the time of the failure. Consequently the parameter is conditional and is required to be present either on the request or the response primitive.

NOTE - Recovery to a point following a bulk data transfer completion is defined for the case where the receiver has received an F-DATA-END indication but the F-TRANSFER-END exchange has not completed. In such a situation it would be unnecessary to go back to the last confirmed checkpoint. The only requirement is for both sender and receiver to have an agreed understanding of the completion of the transfer. A value one larger than the highest checkpoint number issued by the sender is used to indicate recovery after bulk data transfer.

19.1.2.9 Diagnostic

The diagnostic parameter is defined in 13.13.

20 Access to file contents

20.1 Bulk data transfer service

The access to all or part of the content of a file is performed by the bulk data transfer procedures defined in section three.

These procedures start and finish in a single data transfer idle state, and so can be considered as a self-contained procedural unit, which is primitive in the definition of the remainder of the file service.

In using the bulk data transfer procedures, the following FTAM specific definitions apply:

a) Table 25 defines the sub-parameters in the Bulk Data Transfer Specification parameter for a read transfer:

Table 25 — BDT read sub-parameters

Sub-parameter	Status	Defined in
File Access Data Unit Identity	Mandatory	13.12
Access Context	Mandatory	20.1.2
FADU Lock	Optional	13.9

b) Table 26 defines the sub parameters in the Bulk Data Transfer Specification parameter for a write transfer:

Table 26 — BDT write sub-parameters

Sub-parameter	Status	Defined in
File Access Data Unit Operation	Mandatory	20.1.1
File Access Data Unit Identity	Mandatory	13.12
FADU Lock	Optional	13.9

The filestore provider performs the locate action after receiving an F-READ or F-WRITE indication. It then initiates a read, insert, replace or extend action as appropriate for the FADU requested or for each FADU received. The read action is completed before the issue of an F-DATA-END request and the write action or actions are completed before the issue of an F-TRANSFER-END response primitive.

The file remains open after a sequence of F-CANCEL primitives, although the result of interrupted operations is not defined. Further F-READ or F-WRITE operations, not necessarily related to any previous read or write attempt, may be attempted after the completion of the sequence of F-CANCEL primitives has disposed of any previous activity.

20.1.1 File Access Data Unit Operation

The file access data unit operation parameter indicates the action to be taken by the filestore provider on receipt of the data transferred. The values are defined in the constraint set in use.

20.1.2 Access Context

The access context parameter specifies a view of the file access structure, for read operations, which is to be used for a read transfer during the data transfer regime.

The valid values of the access context parameter are dependent on the constraint set in use.

The value of the parameter is given in table 27.

Table 27 — Access contexts

Access Context	Description
HA	Hierarchical All Data Units
HN	Hierarchical No Data Units
FA	Flat All Data Units
FL	Flat One Level Data Units
FS	Flat Single Data Unit
UA	Unstructured All Data Units
US	Unstructured Single Data Unit

The above access contexts are defined in ISO 8571-2.

20.2 Locate file access data unit service

20.2.1 Function

The F-LOCATE service specifies the identity of a file access data unit which is to be located by the filestore provider in preparation for file access.

20.2.2 Types of primitives and parameters

Table 28 indicates the types of primitives and the parameters needed for a locate interaction.

20.2.2.1 Action Result

The action result parameter is defined in 13.2.

20.2.2.2 File Access Data Unit Identity

The file access data unit identity parameter is defined for the request and indication primitives in 13.12. The identity in the request and indication may be any of the values defined in ISO 8571-2, including next, first or last.

The response and confirm primitives return an identity which is one of: begin, end, Node-name, Sequence of node-names or Node number.

The above identities are defined in ISO 8571-2.

This parameter is used to set the current location activity attribute.

Table 28 — F-LOCATE parameters

Parameter	F-LOCATE request	F-LOCATE indication	F-LOCATE response	F-LOCATE confirm
Action Result			Mandatory	Mandatory
File Access Data Unit Identity	Mandatory	Mandatory (=)	Optional	Optional (=)
FADU Lock	Optional	Optional (=)		
Diagnostic			Optional	Optional

(=) signifies that the value is not modified by the file service provider.

20.2.2.3 FADU Lock

The FADU lock parameter is defined in 13.9.

20.2.2.4 Diagnostic

The diagnostic parameter is defined in 13.13.

20.3 Erase file access data unit service

20.3.1 Function

The F-ERASE service specifies the identity of a file access data unit which is to be erased by the filestore provider. The filestore provider performs the erase action after receiving the F-ERASE indication, and before issuing the F-ERASE response primitive.

20.3.2 Types of primitives and parameters

Table 29 indicates the types of primitives and the parameters needed for an erase interaction.

20.3.2.1 Action Result

The action result parameter is defined in 13.2

20.3.2.2 File Access Data Unit Identity

The file access data unit identity parameter is defined in 13.12. See ISO 8571-2 for additional information.

20.3.2.3 Diagnostic

The diagnostic parameter is defined in 13.13.

Table 29 — F-ERASE parameters

Parameter	F-ERASE request	F-ERASE indication	F-ERASE response	F-ERASE confirm
Action Result			Mandatory	Mandatory
File Access Data Unit Identity	Mandatory	Mandatory (=)		
Diagnostic			Optional	Optional

(=) signifies that the value is not modified by the file service provider.

Section three: Definition of bulk data transfer primitives

21 Bulk data transfer service primitives

Each of the services constituting the bulk data transfer service is achieved by invoking a sequence of bulk data transfer service primitives.

Table 30 lists for each service;

a) the primitives associated with the service;

b) the parameters associated with the primitives;

c) the bulk data transfer service user that is permitted to issue the request primitive;

d) how failure of the service is notified to the issuer of the request.

In table 30 those parameters and primitives which occur only in the internal bulk data transfer service are enclosed in square brackets.

During a read operation the responder is the sender and the initiator the receiver of the file. During a write operation the initiator is the sender and the responder the receiver of the file.

The semantics of the primitives and their parameters are defined in clauses 23 to 25.

22 Sequences of bulk data transfer primitives

This clause gives the constraints on the valid sequences in which the primitives defined in clauses 24 and 25 can occur. The individual primitives in a service may occur only in the sequences given as part of the primitive definitions.

22.1 Normal sequences

The normal progress of use of the bulk data transfer service is illustrated by the state transition diagram shown in figure 8. Full state transition diagrams are contained in annex E.

22.2 Constraints on issue of primitives

The primitives may be issued in any sequence consistent with the constraints given in tables 31 and 32 The sequences given with the individual primitive definitions apply.

22.2.1 Key to tables 31 and 32

In tables 31 and 32, the entries indicate the functional units required for the succession to occur. The entries are:

Read	read functional unit
Write	write functional unit
Either	read or write functional unit

The row "start of bulk data" indicates entry from tables 9 and 10 of section two, and the column "end of bulk data" indicates return to tables 9 and 10.

Table 30 — Bulk data transfer service primitives

Primitive Name	Confirmed Service	Request by	Parameters	Failure Notification
F-READ	No	Initiator	Bulk data transfer specification	F-DATA-END action result
F-WRITE	No	Initiator	Bulk data transfer specification	F-CANCEL
F-DATA	No	Sender	Data value	F-CANCEL
F-DATA-END	No	Sender	Action result Diagnostic	F-CANCEL
F-TRANSFER-END	Yes	Initiator	Action result Shared ASE information Diagnostic	action result
F-CANCEL	Yes	Either	Action result Shared ASE information Diagnostic	none
[F-CHECK	Yes	Sender	Checkpoint identifier	F-CANCEL]
[F-RESTART	Yes	Either	Checkpoint identifier	F-CANCEL]

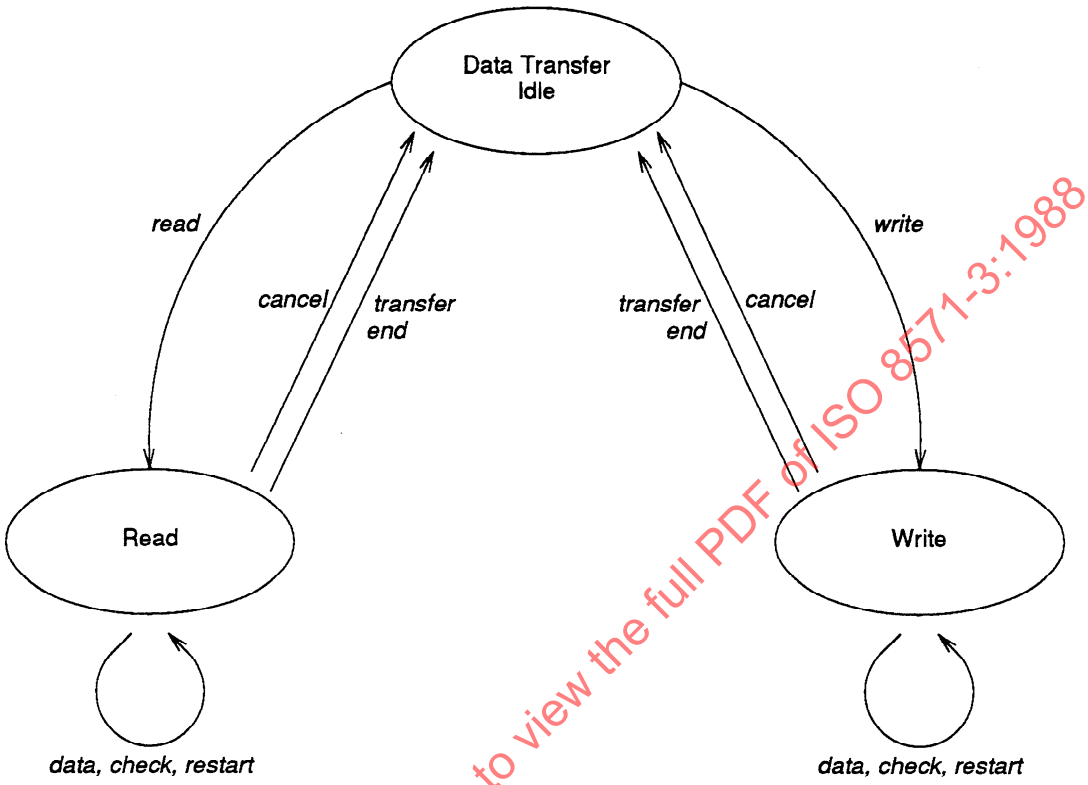

Figure 8 — Simplified State Diagram for Bulk Data Transfer (see Annex E)

Table 31 — Sequence of service primitives for bulk data transfer — initiator

Previous file service event	May next issue				
	F-READ request	F-WRITE request	F-DATA request	F-DATA-END request	F-TRANSFER-END request
Start file Bulk Data	Read	Write			
F-READ request					
F-WRITE request			Write	Write	
F-DATA request			Write	Write	
F-DATA indication					
F-DATA-END request					Write
F-DATA-END indication					Read
F-TRANSFER-END request					
F-TRANSFER-END confirm					
F-CANCEL request					
F-CANCEL indication					
F-CANCEL response					
F-CANCEL confirm					
F-CHECK request			Write	Write	
F-CHECK indication					
F-CHECK response					Read
F-CHECK confirm			Write	Write	Write
F-RESTART request					
F-RESTART indication					
F-RESTART response			Write	Write	
F-RESTART confirm			Write	Write	

May next issue						
F-CANCEL request	F-CANCEL response	F-CHECK request	F-CHECK response	F-RESTART request	F-RESTART response	End of Bulk Data
Read				Read		
Write		Write		Write		
Write		Write		Write		
Read			Read	Read		
Read			Read	Read		
						Either
	Either					
						Either
						Either
Write		Write		Write		
Read			Read	Read		
Read			Read	Read		
Write		Write		Write		
Either						
Either					Either	
Either		Write		Either		
Either		Write		Either		

Table 32 — Sequence of service primitives for bulk data transfer — responder

Previous file service event	May next issue			
	F-DATA request	F-DATA-END request	F-TRANSFER -END response	F-CANCEL request
Start file Bulk Data				
F-READ indication	Read	Read		Read
F-WRITE indication				Write
F-DATA request	Read	Read		Read
F-DATA indication				Write
F-DATA-END request				
F-DATA-END indication				Write
F-TRANSFER-END indication			Either	Write
F-TRANSFER-END response				
F-CANCEL request				
F-CANCEL indication				
F-CANCEL response				
F-CANCEL confirm				
F-CHECK request	Read	Read		Read
F-CHECK indication				Write
F-CHECK response			Write	Write
F-CHECK confirm	Read	Read		Read
F-RESTART request				Either
F-RESTART indication				Either
F-RESTART response	Read	Read		Either
F-RESTART confirm	Read	Read		Either

May next issue					
F-CANCEL response	F-CHECK request	F-CHECK response	F-RESTART request	F-RESTART response	End of Bulk Data
	Read		Read		
			Write		
	Read		Read		
		Write	Write		
		Write	Write		
			Write		
					Either
Either					
					Either
					Either
	Read		Read		
		Write	Write		
		Write	Write		
	Read		Read		
				Either	
	Read		Either		
	Read		Either		

23 Common bulk data transfer parameters

23.1 Bulk Data Transfer Specification

The Bulk Data Transfer Specification parameter identifies the data to be transferred and for read transfer specifies the access context in which the data transfer is tc take place. It may also specify the actions to be taken to generate or use the data transferred. The detailed form of the parameter depends on whether the bulk data transfer to be performed is read or write.

NOTE - The details of the parameter for FTAM use are given in 20.1.

23.2 Checkpoint Identifier

The checkpoint identifier parameter gives an unambiguous identification to allow reference to the checkpoints. The value of the parameter is an integer. For the F-CHECK primitive, the value is between 1 and 999998 inclusive. The value supplied in the first F-CHECK request in a bulk data transfer, either read or write, is 1, and thereafter subsequent values increment by 1. For the F-RESTART primitive, the value is between 0 and 999998.

The value zero is reserved to indicate "beginning of FADU" in restart and recovery activities. The value one greater than the last checkpoint identifier issued by the sender in a series of FADUs before data end is signalled indicates the end of the bulk data transfer. The correlation of the checkpoint identifier range constraint with that of the Session serial number constraint is the responsibility of the application entity as a whole.

24 Bulk data transfer

This group of services performs the transfer of bulk data. The procedure begins with the service initiator issuing either an F-READ request or an F-WRITE request. This leads to the issue of a sequence of F-DATA requests followed by an F-DATA-END request by the sender of the data. The procedure is completed by the initiator issuing an F-TRANSFER-END request. The primitives are defined in 24.1 to 24.6, and the two valid sequences are defined in 24.7 and 24.8.

In the internal service, the checkpointing and recovery primitives defined in clause 25 may occur within or abutted to the sequence of F-DATA primitives.

24.1 Read bulk data service

24.1.1 Function

The F-READ service specifies a data transfer from the service responder (that is the sender) to the service initiator (that is the receiver). Only one F-READ procedure may be in progress at any time on a single application association. The direction of data flow established continues until the exchange of F-TRANSFER-END primitives. Rejection of an F-READ indication is by issue of an F-DATA-END with an action result indicating unsuccessful.

These primitives signal a transfer of control from the initiator to the sender. They mark a reversal of the service asymmetry for the duration of the data transfer.

24.1.2 Types of primitives and parameters

Table 33 indicates the types of primitives and the parameters needed for a read bulk data interaction.

24.1.2.1 Bulk Data Transfer Specification

The bulk data transfer specification is defined in 23.1.

Table 33 — F-READ parameters

Parameter	F-READ request	F-READ indication
Bulk Data Transfer Specification	Mandatory	Mandatory (=)

(=) signifies that the value is not modified by the service provider.

24.2 Write bulk data service

24.2.1 Function

The F-WRITE service specifies a data transfer from the service initiator (that is the sender) to the service provider (that is the receiver). Only one F-WRITE procedure may be in progress at any time on a single application association. The direction of data flow established continues until the exchange of F-TRANSFER-END primitives. An F-WRITE indication can be rejected by issuing an F-CANCEL request (see 24.6). If the transfer is rejected, no further F-DATA indication primitives are received by the responder.

24.2.2 Types of primitives and parameters

Table 34 indicates the types of primitives and the parameters needed for a write bulk data interaction.

Table 34 — F-WRITE parameters

Parameter	F-WRITE request	F-WRITE indication
Bulk Data Transfer Specification	Mandatory	Mandatory (=)

(=) signifies that the value is not modified by the service provider.

24.2.2.1 Bulk Data Transfer Specification

The bulk data transfer specification is defined in 23.1

24.3 Data unit transfer service

24.3.1 Function

The F-DATA service transfers data between the users of the service. The data are transferred as values of known data types, using the underlying Presentation Service mechanisms (see ISO 8822). The data transfer may be from either entity, depending on whether a read or a write transfer has been requested by the initiator.

24.3.2 Types of primitives and parameters

Table 35 indicates the types of primitives and the parameters needed for data value transfer.

24.3.2.1 Data Value

The data value parameter is a value of a known data type; the type identifies the abstract syntax applicable to the data value and the syntactic description of the data value within that abstract syntax.

NOTE - In the supporting protocol, a series of data values may be conveyed as an equivalent series of presentation data values in a single P-DATA primitive.

Table 35 — F-DATA parameters

Parameter	F-DATA request	F-DATA indication
Data Value	Mandatory	Mandatory (=)

(=) signifies that the value is not modified by the service provider.

24.4 End of data transfer service

24.4.1 Function

The completion of the data transfer is indicated by the F-DATA-END primitives. The sender issues an F-DATA-END request primitive when it has sent all the necessary data. The sender may issue an F-DATA-END request with an unsuccessful action result as a rejection of an F-READ indication.

24.4.2 Types of primitives and parameters

Table 36 indicates the types of primitives and the parameters needed to end a data transfer.

Table 36 — F-DATA-END parameters

Parameter	F-DATA-END request	F-DATA-END indication
Action Result	Mandatory	Mandatory
Diagnostic	Optional	Optional

24.4.2.1 Action Result

The action result is defined in 13.2. When the F-DATA-END primitive is used to reject a read indication the reason is carried by the action result and diagnostic. The initiator responds with an F-TRANSFER-END as in the non error case.

24.4.2.2 Diagnostic

The diagnostic parameter is defined in 13.13.

24.5 End of transfer service

24.5.1 Function

The completion of a transfer is indicated by an exchange of F-TRANSFER-END primitives. This exchange is initiated by the initiator after having issued or received an F-DATA-END primitive. Receipt of the F-TRANSFER-END indication or confirm as appropriate informs the sender that no further error recovery actions, involving that bulk data transfer, will be requested.

24.5.2 Types of primitives and parameters

Table 37 indicates the types of primitives and the parameters needed for data transfer end.

24.5.2.1 Action Result

The action result parameter is defined in 13.2.

24.5.2.2 Shared ASE Information

The shared ASE information parameter is defined in 13.10.

24.5.2.3 Diagnostic

The diagnostic parameter is defined in 13.13.

24.6 Cancel data transfer service

24.6.1 Function

Either of the service users may cancel a data transfer activity by issuing an F-CANCEL request primitive. The F-CANCEL primitive may be issued during data transfer after the issue or receipt of an F-READ or F-WRITE request or indication. At the end of data transfer, it may not be issued:

a) by the initiator acting as sender after issue of an F-DATA-END request;

b) by the responder acting as receiver after issue of an F-TRANSFER-END response;

c) by the responder acting as sender after the issue of an F-DATA-END request;

d) by the initiator acting as receiver after the issue of an F-TRANSFER-END request.

If either the cancel data transfer service or the end of data transfer service is used, the data transfer regime ceases. Where use of these services collides, the cancel data transfer service takes precedence.

After an F-CANCEL procedure the two users may have different views of the state of the activity. The F-CANCEL primitives interrupt any activity in progress (including an F-RESTART sequence) and any undelivered indications or confirms may be discarded.

The file remains open after a sequence of F-CANCEL primitives, although the result of interrupted operations is not defined. Further F-READ or F-WRITE operations, not necessarily related to any previous read or write attempt, may be attempted after the completion of the sequence of F-CANCEL primitives has disposed of any previous activity.

24.6.2 Types of primitives and parameters

Table 38 indicates the types of primitives and the parameters needed to cancel data transfer.

Table 37 — F-TRANSFER-END parameters

Parameter	F-TRANSFER-END request	F-TRANSFER-END indication	F-TRANSFER-END response	F-TRANSFER-END confirm
Action Result			Mandatory	Mandatory
Shared ASE Information	Optional	Optional (=)	Optional	Optional (=)
Diagnostic			Optional	Optional

(=) signifies that the value is not modified by the service provider.

Table 38 — F-CANCEL parameters

Parameter	F-CANCEL request	F-CANCEL indication	F-CANCEL response	F-CANCEL confirm
Action Result	Mandatory	Mandatory	Mandatory	Mandatory
Shared ASE Information	Optional	Optional (=)	Optional	Optional (=)
Diagnostic	Optional	Optional	Optional	Optional

(=) signifies that the value is not modified by the service provider.

24.6.2.1 Action Result

The action result parameter, defined in 13.2, and the diagnostic parameter are used to convey the reason for the cancel.

24.6.2.2 Shared ASE Information

The shared ASE information parameter is defined in 13.10.

24.6.2.3 Diagnostic

The diagnostic parameter is defined in 13.13.

24.7 Sequence of primitives on write

The sequence of events in a successful write operation is defined in the time sequence diagram in figure 9. The F-DATA primitives in the diagram stand for an arbitrary sequence of F-DATA primitives in the internal service.

24.8 Sequence of primitives on read

The sequence of events in a successful read operation is defined in the time sequence diagram in figure 10. The F-DATA primitives in the diagram stand for an arbitrary sequence of F-DATA primitives in the internal service.

25 Checkpointing and restart (Internal BDT Service Only)

The checkpointing service is only available if the restart or/and recovery functional units are negotiated on F-INITIALIZE. The restart service is only available if the restart functional unit is negotiated on F-INITIALIZE. The checkpointing and restart primitives may be issued while a data transfer activity is in progress (after an F-READ or F-WRITE and before an F-DATA-END, for the sender and F-TRANSFER-END for the receiver) to control the progress of the data transfer.

NOTE - The receipt of a restart indication after the issue of an F-DATA-END may result in the sender returning to the data transfer activity it has just left.

25.1 Checkpointing service

Checkpoint insertion is only visible in the internal service. The protocol error recovery procedures (see ISO 8571-4) specify how the primitives are used in supporting the external service.

25.1.1 Function

The F-CHECK group of primitives mark and acknowledge points in a sequence of F-DATA primitives. The receipt of the F-CHECK confirm informs the sender of data, that data before the point marked and acknowledged has been received and secured, such that no earlier point is requested in an F-RESTART primitive. The sender may continue to send F-DATA primitives or further checkpoints before receipt of any particular primitive has been acknowledged. The number of checkpoints which may be outstanding is negotiated when the FTAM regime in which the bulk data is to be transferred is established. However, all outstanding checkpoints are acknowledged before the F-TRANSFER-END request or response is issued.

25.1.2 Types of primitives and parameters

Table 39 indicates the types of primitives and parameters needed for checkpointing.

25.1.2.1 Checkpoint Identifier

The checkpoint identifier parameter is defined in 23.2.

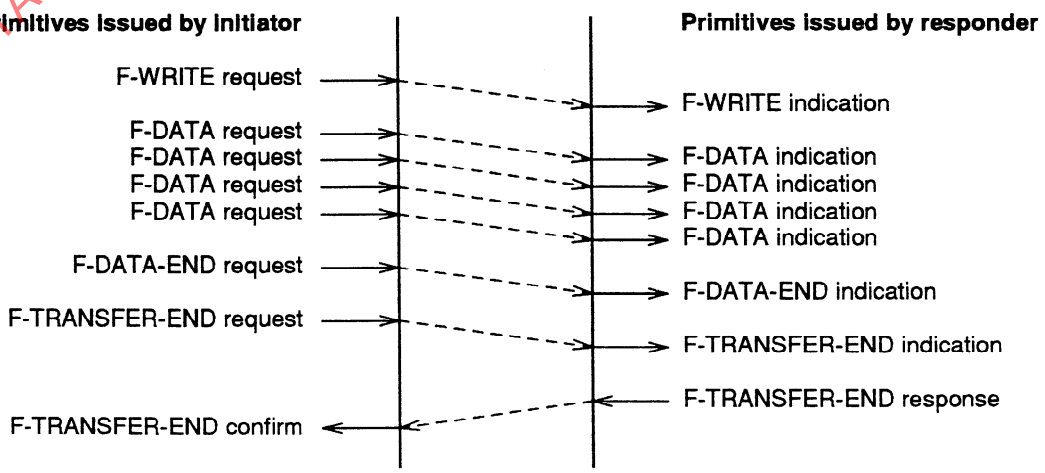

Figure 9 — Sequence of primitives on write

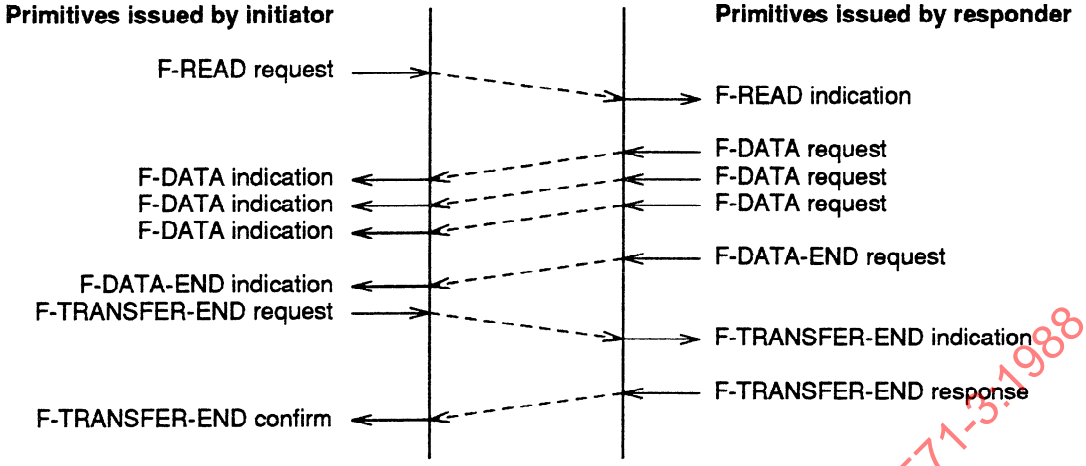

Figure 10 — Sequence of primitives on read

Table 39 — F-CHECK parameters

Parameter	F-CHECK request	F-CHECK indication	F-CHECK response	F-CHECK confirm
Checkpoint Identifier	Mandatory	Mandatory (=)	Mandatory	Mandatory (=)

(=) signifies that the value is not modified by the service provider.

25.2 Restarting data transfer service

Restarting of the data transfer is only visible in the internal service. The protocol error recovery procedures (see ISO 8571-4) specify how the primitives are to be used in supporting the external service.

25.2.1 Function

The F-RESTART group of primitives interrupts any bulk data transfer activity in progress, with possible loss of any undelivered indications or confirms. It negotiates a point at which data transfer is to be restarted. At the end of data transfer, it may not be issued:

a) by the initiator acting as sender after issue of an F-DATA-END request;

b) by the responder acting as receiver after issue of an F-TRANSFER-END response;

c) by the responder acting as sender after the issue of an F-DATA-END request;

d) by the initiator acting as receiver after the issue of an F-TRANSFER-END request.

If the receiver and the sender of data both issue an F-RESTART request, the service provider resolves the collision and issues a confirmation to each user giving the point at which the transfer is to be restarted (see ISO 8571-4). An F-RESTART indication is rejected by using an F-CANCEL request primitive.

25.2.2 Types of primitives and parameters

Table 40 indicates the types of primitives and the parameters needed to restart data transfer.

25.2.2.1 Checkpoint Identifier

The checkpoint identifier parameter is defined in 23.2. The use of the parameter depends on the direction of transfer. The issuer of the request identifies a checkpoint which is:

a) for the sender, the last point acknowledged;

b) for the receiver, the last point received and secured.

The issuer of the response identifies a checkpoint which is:

c) for the sender, equal to the value provided by the issuer of the request;

d) for the receiver, the last point received and secured.

In a collision, the value provided by the receiver is returned to both the sender and the receiver.

Table 40 — F-RESTART parameters

Parameter	F-RESTART request	F-RESTART indication	F-RESTART response	F-RESTART confirm
Checkpoint Identifier	Mandatory	Mandatory (=)	Mandatory	Mandatory (=)

(=) signifies that the value is not modified by the service provider.

Annex A
Diagnostic parameter values

(This annex forms part of the standard.)

A.1 Introduction

This annex defines the components which form the diagnostic parameter. It details for each component the valid values and meaning of those values.

A.2 Form of the diagnostic parameter

The value of the diagnostic parameter is structured into the following items:

a) an error type (see A.3), indicating "permanent error" or "transient error" or "informative". A permanent error occurs every time the sequence of events is repeated, and implies the failure of at least the present operation being performed. A transient error may not re-occur if the sequence is repeated but does imply the failure of the operation being performed. An informative error does not require recovery and does not affect the current state of the file service;

b) an error identifier (see A.4); the error identifiers categorize errors in terms of concepts defined in the virtual filestore definition or in terms of ISO 7498.

c) the type of entity which detected the error, called the observer (see A.5).

d) the type of entity which is believed to have caused the error called the presumed source (see A.5).

e) for transient errors, optionally, a suggested delay before recovery is attempted. The value is an integer *x* which indicates a delay time of 2 to power *x* seconds.

f) optionally, a text message in natural language giving further details of the cause of the error; it may include unstandardized concepts relating to the local system environment of the filestore provider. The values are of the type GraphicString.

A.3 Error type

Table 41 defines error type values for use in the diagnostic parameter. The values in the table identify classes of error of different severity.

A.4 Error identifiers

The diagnostics are classified into related groups and numbered accordingly.

A.5 Observer and source

The observer of the error and the presumed source are indicated from the categories defined in table 42.

A.6 Applicability of the diagnostic parameter

The diagnostic parameter occurs on the following services:

F-INITIALIZE
F-U-ABORT
F-P-ABORT
F-SELECT
F-DESELECT
F-CREATE
F-DELETE
F-READ-ATTRIB
F-CHANGE-ATTRIB
F-OPEN
F-CLOSE
F-RECOVER
F-LOCATE
F-ERASE
F-DATA-END
F-TRANSFER-END
F-CANCEL

Table 41 — Error types

Error type value	Error type
0	informative
1	transient error
2	permanent error

Table 42 — Sources and observers of errors

Observer and source identifier	Description	Qualification
0	no categorization possible	source only
1	initiating file service user	
2	initiating file protocol machine	
3	service supporting the file protocol machines	source only
4	the responding file protocol machine	
5	the responding file service user (filestore)	

Table 43 — General FTAM diagnostics

Type	Identifier	Observer	Source	Reason
12	0	0-5	0-5	No reason
012	1	5	5	Responder error (unspecific)
12	2	0-5	0-5	System shutdown
012	3	5	1	FTAM management problem (unspecific)
0 2	4	5	1	FTAM management, bad account
0 2	5	5	1	FTAM management, security not passed
0	6	5	5	Delay may be encountered
012	7	1,5	1	Initiator error (unspecific)
012	8	0-5	0-5	Subsequent error
012	9	0-5	0-5	Temporal insufficiency of resources
12	10	5	1	Access request violates VFS security
12	11	5	1	Access request violates local security

Table 44 — Protocol and supporting service related diagnostics

Type	Identifier	Observer	Source	Reason
2	1000	2	4	Conflicting parameter values
		4	2	"
2	1001	2	4	Unsupported parameter values
		4	2	"
2	1002	2	4	Mandatory parameter not set
		4	2	"
2	1003	2	4	Unsupported parameter
		4	2	"
2	1004	2	4	Duplicated parameter
		4	2	"
2	1005	2	4	Illegal parameter type
		4	2	"
2	1006	2	4	Unsupported parameter types
		4	2	"
2	1007	2	4	FTAM protocol error (unspecific)
		4	2	"
2	1008	2	4	FTAM protocol error, procedure error
		4	2	"
2	1009	2	4	FTAM protocol error, functional unit error
		4	2	"
2	1010	2	4	FTAM protocol error, corruption error
		4	2	"
2	1011	2-4	3	Lower layer failure
12	1012	3	2	Lower layer addressing error
12	1013	0-5	0-5	Timeout
12	1014	0-5	0-5	System shutdown
2	1015	2	4	Illegal grouping sequence
		4	2	"
2	1016	2	4	Grouping threshold violation
		4	2	"
2	1017	4	2	Specific PDU request inconsistent with the current requested access

Table 45 — Association related diagnostics

Type	Identifier	Observer	Source	Reason
2	2000	3,5	1	Association with user not allowed
	2001			(not assigned)
2	2002	4,5	1	Unsupported service class
0 2	2003	4,5	1	Unsupported functional unit
012	2004	5	1	Attribute group error (unspecific)
2	2005	5	1	Attribute group not supported
0 2	2006	5	1	Attribute group not allowed
0 2	2007	5	1	Bad account
012	2008	4	1	Association management (unspecific)
2	2009	4	1	Association management - bad address
12	2010	4	1	Association management - bad account
0 2	2011	4	2	Checkpoint window error - too large
0 2	2012	4	2	Checkpoint window error - too small
0 2	2013	4	2	Checkpoint window error - unsupported
012	2014	3	2	Communications QoS not supported
2	2015	4,5	1	Initiator identity unacceptable
0	2016	4	1	Context management refused
0	2017	4	1	Rollback not available
0	2018	5	1	Contents type list cut by responder
0	2019	3,5	1	Contents type list by Presentation service
2	2020	5	1	Invalid filestore password
2	2021	4,5	1	Incompatible service classes

Table 46 — Selection related diagnostics

Type	Identifier	Observer	Source	Reason
12	3000	5	5	Filename not found
12	3001	5	5	Selection attributes not matched
2	3002	5	5	Initial attributes not possible
2	3003	4	2	Bad attribute name
12	3004	5	5	Non-existent file
12	3005	5	5	File already exists
12	3006	5	5	File cannot be created
12	3007	5	5	File can not be deleted
0 2	3008	5	5	Concurrency control not available
0 2	3009	5	5	Concurrency control not supported
0 2	3010	5	5	Concurrency control not possible
01	3011	5	5	More restrictive lock
12	3012	5	5	File busy
12	3013	5	5	File not available
012	3014	5	5	Access control not available
012	3015	5	5	Access control not supported
012	3016	5	5	Access control inconsistent
0	3017	4,5	4,5	Filename truncated
0	3018	5	5	Initial attributes altered
12	3019	5	1	Bad account
0	3020	5	5	Override selected existing file
0	3021	5	5	Override deleted and recreated file with old attributes
0	3022	5	5	Create override deleted and recreate file with new attributes
12	3023	5	5	Create override - not possible
12	3024	5	5	Ambiguous file specification
2	3025	5	1	Invalid create password
2	3026	5	1	Invalid delete password on override
2	3027	5	1	Bad attribute value
2	3028	5	1	Requested access violates permitted actions
2	3029	5	1	Functional unit not available for requested access
01	3030	5	5	File created but not selected

Table 47 — File management related diagnostics

Type	Identifier	Observer	Source	Reason
012	4000	5	5	Attribute non - existent
12	4001	5	5	Attribute cannot be read
12	4002	5	5	Attribute cannot be changed
12	4003	4,5	4,5	Attribute not supported
2	4004	4	2	Bad attribute name
2	4005	5	1	Bad attribute value
0	4006	5	5	Attribute partially supported
0	4007	5	5	Additional set attribute value not distinct

Table 48 — Access related diagnostics

Type	Identifier	Observer	Source	Reason
12	5000	5	1	Bad FADU (unspecific)
2	5001	5	1	Bad FADU - size error
2	5002	5	1	Bad FADU - type error
2	5003	5	1	Bad FADU - poorly specified
2	5004	5	1	Bad FADU - bad location
01	5005	5	5	FADU does not exist
012	5006	5	5	FADU not available (unspecific)
12	5007	5	5	FADU not available for reading
12	5008	5	5	FADU not available for writing
12	5009	5	5	FADU not available for location
12	5010	5	5	FADU not available for erasure
12	5011	5	5	FADU cannot be inserted
12	5012	5	5	FADU cannot be replaced
012	5013	5	5	FADU cannot be located
2	5014	5	1	Bad data element type
		3	2,4	"
12	5015	5	5	Operation not available
12	5016	5	5	Operation not supported
0 2	5017	5	1	Operation inconsistent
012	5018	5	5	Concurrency control not available
0 2	5019	5	5	Concurrency control not supported
0 2	5020	5	1	Concurrency control inconsistent
012	5021	5	5	Processing mode not available
0 2	5022	5	5	Processing mode not supported
0 2	5023	5	1	Processing mode inconsistent
0 2	5024	5	5	Access context not available
0 2	5025	5	5	Access context not supported
12	5026	5	5	Bad write (unspecific)
12	5027	5	5	Bad read (unspecific)
012	5028	5	5	Local failure (unspecific)
012	5029	5	5	Local failure - filespace exhausted
012	5030	5	5	Local failure - data corrupted
012	5031	5	5	Local failure - device failure
2	5032	5	5	Future file size exceeded
0	5034	5	5	Future file size increased
0 2	5035	5	1	Functional unit invalid in processing mode
0 2	5036	5	1	Contents type inconsistent
0	5037	1	5	Contents type simplified
0	5038	5	5	Duplicate FADU name
12	5039	2	4	Damage to select/open regime
		4	2	"
12	5040	5	5	FADU locking not available on file
12	5041	5	5	FADU locked by another user

Table 49 — Recovery related diagnostics

Type	Identifier	Observer	Source	Reason
2	6000	4	2	Bad checkpoint (unspecific)
		2	4	"
2	6001	4	2	Activity not unique
2	6002	4	2	Checkpoint outside window
		2	4	"
2	6003	4	2	Activity no longer exists
2	6004	4	2	Activity not recognized
2	6005	4	4	No docket
2	6006	4	4	Corrupt docket
12	6007	4	4	File waiting restart
012	6008	4	2	Bad recovery point
2	6009	4	2	Non-existent recovery point
0 2	6010	4,5	2,1	Recovery mode not available
0 2	6011	4,5	2,1	Recovery mode inconsistent
0	6012	4,5	2,1	Recovery mode reduced
0 2	6013	5	1	Access control not available
0 2	6014	5	1	Access control not supported
0 2	6015	5	1	Access control inconsistent
0 2	6016	5	1	Contents type inconsistent
0	6017	5	1	Contents type simplified

Annex B
Relation of attributes to primitives

(This annex forms part of the standard.)

Tables 50 and 51 indicate which of the attributes defined in the virtual filestore are affected by the various primitives defined in the file service.

Primitives which do not occur in the tables 50 and 51 do not affect the virtual filestore attributes directly. However, the deselect primitive restores a previous value of the current account attribute.

Each entry in the table indicates the resultant effect of a particular primitive or group of primitives on one attribute. The table entries are:

a) blank; the attribute is not affected;

b) return; the attribute value is returned on the response and confirm primitives;

c) change; the attribute value is changed on the basis of a value transferred by the request and indication primitives; the value of a scalar attribute is replaced; the value of a vector attribute must be replaced completely; for set attributes individual elements may be changed; for creation, the initial value is assigned;

d) implicit; the attribute value is changed as a result of the use of the primitives, but no value is transferred; the new value results from the actions performed by the primitives;

e) set; the attribute value is set on the basis of a value transferred on a parameter of the request and indication primitives of F-CREATE, or to a local default if there is no value transferred;

f) compare; a value is transferred with the request and indication primitives, and the actions specified for the primitives are performed only if the given value matches the attribute value. The algorithm for matching values is part of the attribute definition.

Table 50 — File attributes

Attribute Name	F-SELECT	F-CREATE	F-READ -ATTRIB	F-CHANGE -ATTRIB	F-CLOSE & F-ABORT
filename	compare & return	set & return	return	change & return	
permitted actions	compare	set	return		
contents type		set	return		
storage account	compare	set	return	change & return	
date and time of creation		implicit	return		
date and time of last modification		implicit	return		implicit
date and time of last read access		implicit	return		implicit
date and time of last attribute modification		implicit	return	implicit	
identity of creator		set	return		
identity of last modifier		implicit	return		implicit
identity of last reader		implicit	return		implicit
identity of last attribute modifier		implicit	return	implicit	
file availability		set	return	change & return	
filesize		implicit	return		implicit
future filesize		set	return	change & return	
access control	compare	change	return (see note)	change & return (see note)	
legal qualifications		set	return	change & return	
private use		set	return	change & return	

NOTE - Any password values should not be returned

Table 51 — Activity attributes

Attribute Name	F-INITIALIZE	F-SELECT	F-CREATE	F-OPEN	F-READ & F-WRITE	F-LOCATE & F-ERASE	F-RECOVER
current access request		change	change	compare			change
current initiator identity	change						
current location				change	change	change	change
current processing mode				change			change
current calling application entity title	change						
current responding application entity title	change						
current account	change	change	change				
current concurrency control		change	change	change			
current locking style				change			change
current access passwords		change	change				change
active contents type				change			change
active legal qualifications				change			change