INTERNATIONAL STANDARD

ISO 8571-1

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 1 : General introduction

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

Partie 1 : Introduction générale

Reference number
ISO 8571-1 : 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 8571-1 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 and B are for information only.

INTERNATIONAL STANDARD ISO 8571-1 : 1988
TECHNICAL CORRIGENDUM 1

Published 1991-06-01

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

Part 1: General introduction

TECHNICAL CORRIGENDUM 1

Systèmes de traitement de l'information — Interconnexion de systèmes ouverts — Gestion, accès et transfert de fichier — Partie 1: Introduction générale

RECTIFICATIF TECHNIQUE 1

Technical corrigendum 1 to International Standard ISO 8571-1 : 1988 was prepared by ISO/IEC JTC 1, *Information technology.*

Page ii

Foreword

Fifth paragraph, add the following: *"Part 5: Protocol Implementation Conformance Statement Proforma"*

Page 1

Clause 0

Fifth paragraph, line 1, delete "four" and insert "five"

After Part 4 insert the following: "Part 5: Protocol Implementation Conformance Statement Proforma"

UDC 681.3.01 **Ref. No. ISO 8571-1 : 1988/Cor.1 : 1991 (E)**

Descriptors: data processing, information interchange, network interconnection, open systems interconnection, files, communication procedure, management.

Clause 1

Line 2, delete "parts 2 to 4" and insert "parts 2 to 5"

Clause 2

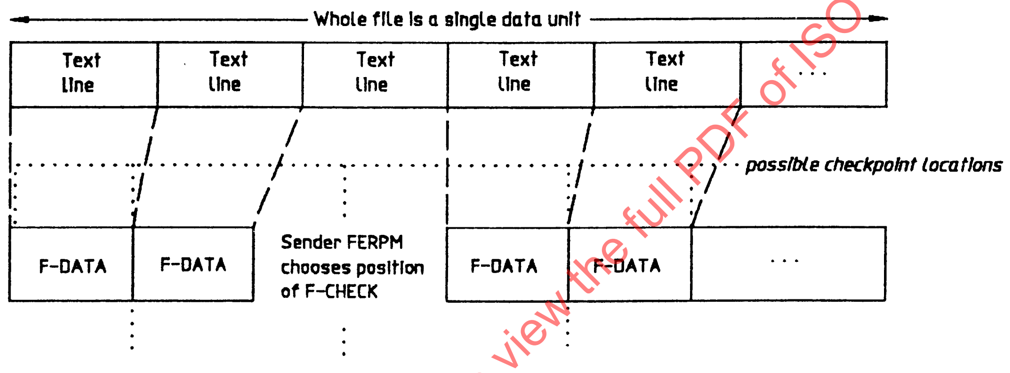
After ISO 8571, Part 4, insert the following: *"Part 5: Protocol Implementation Conformance Statement Proforma."*

Delete reference to ISO 9804 and insert the following: "ISO/IEC 9804, *Information technology — Open Systems Interconnection — Service definition for the Commitment, Concurrency and Recovery service element."*

Page 15

Figure 9

Change figure 9 by extending the *possible checkpoint locations* as shown below:


Whole file is a single data unit
Text line
Text line
Text line
Text line
Text line
...
possible checkpoint locations
F-DATA
F-DATA
Sender FERPM chooses position of F-CHECK
F-DATA
F-DATA
...

Contents

Figures

This page intentionally left blank

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

Part 1 : General introduction

0 Introduction

ISO 8571 is one of a set of International Standards produced to facilitate the interconnection of computer systems. Its relation to other International Standards in the set is 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, and 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.

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

The definitions in this part of ISO 8571 are used in the subsequent parts of ISO 8571 which specify the virtual filestore, services and protocols.

This part of ISO 8571 contains the fo..owing annexes which do not form part of the standard:
Annex A - Examples of the use of FTAM
Annex B - Summary of objects identified

1 Scope and field of application

This part of ISO 8571 provides a general introduction to the concepts and mechanisms specified in parts 2 to 4 of ISO 8571.

2 References

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

ISO 8326, *Information Processing Systems - Open Systems Interconnection - Basic Connection Oriented Session Service Definition.*

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 2: Virtual Filestore definition.*
- *Part 3: File Service 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 8824, *Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).*

ISO 8825, *Information Processing Systems - Open Systems Interconnection - Specification of Basic encoding rules for Abstract Syntax Notation One (ASN.1).*

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

ISO 9834-2, *Information Processing Systems - Procedures for specific OSI registration authorities - Part 2: Registration of Document Types.* [1)]

3 Reference model definitions

ISO 8571 is based on the concepts developed in ISO 7498 and makes use of the following terms defined in it:

a) application-entity;

b) application process;

c) application service element;

d) (N)-connection;

e) open system;

f) (N)-protocol;

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

g) (N)-protocol-control-information;
h) (N)-protocol-data-unit;
i) (N)-service-access-point;
j) (N)-service-access-point-address;
k) (N)-service-data-unit;
l) sublayer;
m) (N)-user-data;
n) user element.

4 Service conventions definitions

ISO 8571 makes use of the following terms defined in ISO/TR 8509 as they apply to the file service:

a) confirm;
b) indication;
c) primitive;
d) request;
e) response;
f) service provider;
g) service user.

5 FTAM definitions

Unless otherwise stated, all terms apply to the view of a system presented for the purpose of open interconnection. This implies that the terms relate to a virtual filestore rather than to any real filestore (see clause 7).

The definitions are grouped into major categories, and ordered alphabetically within each category.

For the purpose of ISO 8571, the following definitions apply:

5.1 General

5.1.1 empty file: A file whose file contents consist of only a root node with no associated data unit, and no node name.

5.1.2 file access: The inspection, modification, replacement or erasure of part of a file's contents.

5.1.3 file contents: The data units, node names and structuring information contained in the file, which may be manipulated during the file open regime; the file attributes do not form part of the file's contents.

5.1.4 file management: The creation and deletion of files, and the inspection or manipulation of the file attributes.

5.1.5 file transfer: A function which moves a part or the whole of a file's contents between open systems.

5.1.6 hierarchical file model: A model of the internal structure of a file which takes the form of a tree of nameable file access data units.

5.1.7 real file: The named collection of information and its attributes which reside in a real system and to which the references to virtual files made in the OSIE are mapped.

5.1.8 real filestore: An organized collection of files, including their attributes and names, which reside in a real system and to which the virtual file references in the OSIE are mapped.

5.1.9 virtual file; file: An unambiguously named collection of structured information having a common set of attributes.

5.1.10 virtual filestore: An abstract model for describing files and filestores, and the possible actions on them. Where no ambiguity exists, the term is shortened in ISO 8571 to "filestore".

5.2 Architectural

5.2.1 accounting regime: The period during which a particular set of accounting information applies.

5.2.2 commitment unit: A set of filestore actions which either succeeds, the effect then being made visible to other processes, or fails completely with no effect visible to other processes.

NOTE - Nullification of the actions in a commitment unit is possible at any time up to the completion of the commitment unit.

5.2.3 docket: That collection of information which can be associated with a file service regime and which must be preserved if error recovery is to be possible.

5.2.4 external file service: File Transfer, Access and Management as seen by the file service user.

5.2.5 file service user: That portion of the application entity which conceptually invokes the FTAM service.

5.2.6 initiator: That file service user which requests FTAM regime establishment.

5.2.7 internal file service: The service used by the file error recovery protocol machine to transmit both file error recovery protocol control information and normal file protocol control information.

5.2.8 open systems interconnection environment: The set of definitions of the standardized services, protocols and data structures which enable the interconnection of systems.

5.2.9 phase: The period of time in which protocol exchanges have a particular purpose, such as establishing or releasing an application context; for each phase a set of valid messages is defined in terms of state transitions.

NOTE - An entity is in one phase at any time.

5.2.10 presentation data value: The unit of information, specified at the abstract syntax level, which is transferred by the presentation service.

5.2.11 real system environment: The implementation aspects which support an application process within a real system.

5.2.12 receiving entity; receiver: The entity which receives part or all of the file's contents during the file data transfer regime.

5.2.13 regime: The period during which the entity is in a subset of its possible states for which particular actions are permitted.

NOTE - Regimes may be nested.

5.2.14 responder: That file service user which accepts an FTAM regime establishment requested by the initiator.

5.2.15 rollback: The nullification of uncommitted actions.

5.2.16 sending entity; sender: The entity which sends part or all of the file's contents during the file data transfer regime.

5.2.17 service element: A unit of standardization specifying a complete group of functions.

5.2.18 service primitive: The smallest defined interaction between the user and the provider of a communication service.

5.2.19 symbiotic service element: A service element which will support the operation of some second service by adopting its semantics and including parts of its abstract syntax at defined points in the abstract syntax of the first service's protocol control information.

5.3 Filestore schema

5.3.1 activity attributes: The attributes describing the activity of using the file service. The attributes are local to one FTAM regime (and any regime nested within it).

5.3.2 attribute: A piece of information stating a property of something, taking one of a set of defined values, each value having a defined meaning.

5.3.3 file attributes: The name and other identifiable properties of a file.

NOTE - The same value of a file attribute is observed at a particular time by any user of the file service, even when more than one user is active at that time.

5.4 Filestore access

5.4.1 access context: The specification of an algorithm defining a subset of the structuring information and user information in a file's contents, when reading the file for transfer or access.

5.4.2 data element: The smallest piece of data whose identity is necessarily preserved when transferred by the Presentation Service. A data element can convey file contents information, file structuring information or protocol control information.

5.4.3 data unit: The smallest unit of a file's contents which the filestore actions can manipulate. Each data unit is associated with a node of the file access structure. A data unit is a series of data elements.

5.4.4 file access data unit: A unit of the file access structure on which the actions of transfer, deletion, extension, replacement or insertion can be performed. A file access data unit contains zero or more data units.

5.4.5 filestore action: One of the actions specified as part of the definition of the virtual filestore.

5.4.6 file access structure: The data structure of a file that relates the file access data units, allowing their identification, description and manipulation.

5.5 File structure

5.5.1 arc: A directed link between two nodes.

5.5.2 arc length: A positive integer expressing the difference in levels between a child node and its parent node.

5.5.3 child (of a node): A node at which an outbound arc of the node concerned terminates.

5.5.4 leaf: A node of a tree that has no outbound arcs.

5.5.5 level (of a node): The sum of the arc lengths from the root to the node concerned.

5.5.6 long arc: An arc with an arc length greater than one.

5.5.7 node: The elementary component from which a tree is built up.

5.5.8 ordered tree: A tree in which there is a defined ordering of the children of each node in the tree.

5.5.9 parent (of a node): The node from which the inbound arc of the node concerned originates.

5.5.10 path: A sequence of arcs which links one node to another node in such a way that each arc is traversed in its defined direction.

5.5.11 root: The unique node of a tree that has no inbound arcs; it is at level 0.

5.5.12 sister (of a node): A node which shares the same parent node as the node concerned.

5.5.13 subtree: A part of a tree comprising an arbitrary node as the subtree root node and all the other nodes which can be reached by a path from the subtree root node.

5.5.14 traversal sequence: An ordering of the nodes in a tree such that each node occurs once and only once, and which is determined by an algorithm applicable to all possible trees.

NOTE - In general, many different trees may generate the same traversal sequence.

5.5.15 tree: A connected structure in which each node is linked to other nodes by directed arcs in such a way that one node has no inbound arcs and all other nodes have exactly one inbound arc.

5.6 Constraint set

5.6.1 constraint set: A set of restrictions and refinements of a general file model which specifies a less general model tailored to the needs of a particular class of applications.

5.6.2 file model: A model of the access structure of a file's contents.

5.6.3 flat (constraint set): A constraint set which, when applied to the general hierarchical file model, generates an access structure that consists of two levels, at the levels zero and one, and that may have data units at only the leaf nodes and has no data unit at the root node.

5.6.4 general hierarchical file model: a model in which the file access data units are organized in a hierarchical tree.

5.6.5 hierarchical (constraint set): A constraint set which, when applied to the general hierarchical file model, generates an access structure which is still hierarchical, but in which the form of the node descriptions and data units is restricted.

5.6.6 unstructured (constraint set): A constraint set which, when applied to the general hierarchical file model, generates an access structure that consists only of the root node with one data unit.

5.7 Document types

5.7.1 concatenation (of documents): The combination of two documents to form a single resultant document.

5.7.2 document: A collection of information with known abstract syntaxes and partially known semantics, and a known set of possible transfer syntaxes.

5.7.3 document type: The specification of a class of documents, which states their necessary semantics, abstract syntaxes, transfer syntaxes and dynamics.

5.7.4 dynamics (of a document): The concatenation and simplification properties of a document.

5.7.5 relaxation (of a document): The process of deriving one document from another by making the parameters describing it less restrictive.

5.7.6 simplification (of a document): The process of deriving one document from another of a different type by discarding structural information.

6 Abbreviations

The following abbreviations are used in parts 2 to 4 of ISO 8571:

ACSE	association control service element
ASE	application service element
CCR	commitment, concurrency and recovery
DU	data unit
EFS	external file service
FADU	file access data unit
FERPM	file error recovery protocol machine
FPDU	file protocol data unit
FPM	file protocol machine
FTAM	file transfer, access and management
Id	identifier
IFS	internal file service
OSI	open systems interconnection
OSIE	open systems interconnection environment
PCI	protocol control information
PDU	protocol data unit
PDV	presentation data value
PSDU	presentation service data unit
RSE	real system environment

Section one: FTAM general concepts

7 OSI architectural background

The aim in the standardization of a file service is to allow the Open Systems Interconnection of file users, who wish to transfer, access or manage information held by systems which behave as if they store files. Anything which appears as an open system, and conforms to the specified file protocols in the role of responder, is considered to provide a virtual filestore.

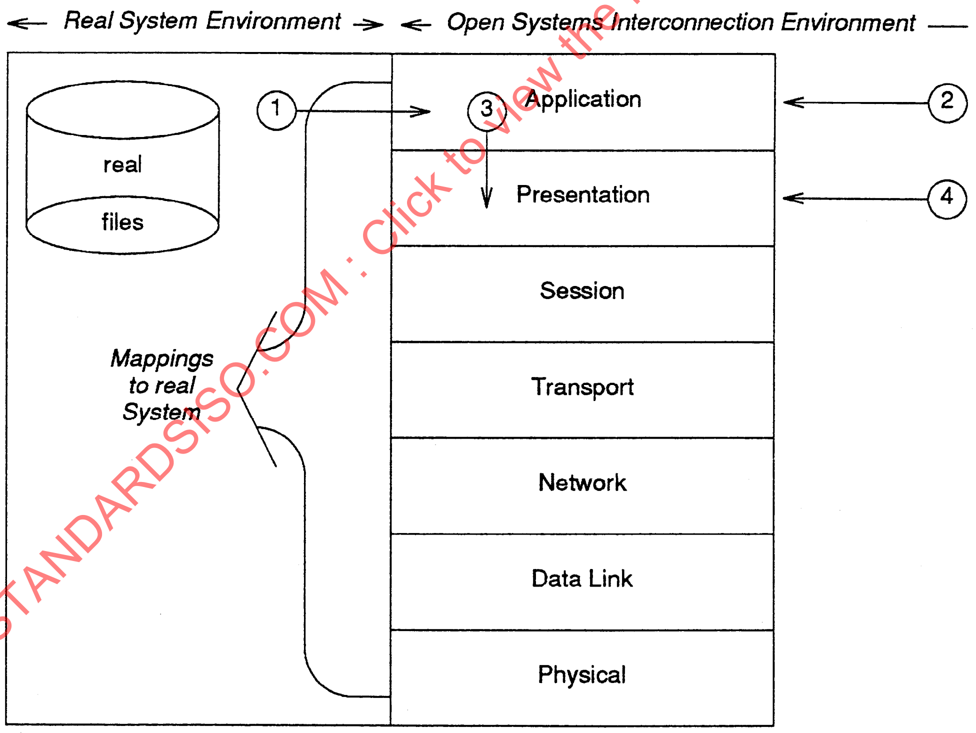
As in all the Open Systems Interconnection standards, the architectural divisions used here reflect classifications of protocol. They do not in any way constrain the way that possible underlying implementations are to be structured. The only point at which conformance can be tested is the point of interconnection.

The various parts of ISO 8571 are part of a larger set of Open Systems Interconnection standards, which are interrelated by the OSI Basic Reference Model (ISO 7498). The aspects of interconnection protocols common to many different fields of use are defined in separate standards, and the logical relation of the various elements is provided by supporting definitions of the services available.

In understanding the intent of OSI, it is important to distinguish carefully between the interrelated set of standards and the implementation, in hardware and software, of a real open system using the protocols specified in the standards. This distinction leads to the identification of two environments:

a) the implementation aspects, in terms of the real facilities and resources which support an application process within a real system, constitute the Real System Environment (RSE);

b) the set of definitions of the standardized services and protocols and data structures which enable the interconnection of systems constitute the Open Systems Interconnection Environment (OSIE). The stylized aspects of the behaviour of an application process visible in the Open Systems Interconnection Environment constitute the application entity; the activities of the application entities are supported by the layered communication functions defined in the OSI reference model.



NOTES

1 The flow of activations between the RSE and the OSIE represents the mapping of the intention to communicate onto the application entity user element.

2 The flow between application entities (both the application PCI and any user data) is a logical flow of information in an agreed abstract syntax, which is independent of any particular transfer syntax.

3 The flow between the application and presentation entities is a logical flow of information, including both user data and application protocol control information in one or more agreed presentation contexts.

4 The flow between the presentation entities is of data encoded in the negotiated transfer syntaxes, and of presentation protocol control information.

Figure 1 — Flow of information between RSE and OSIE

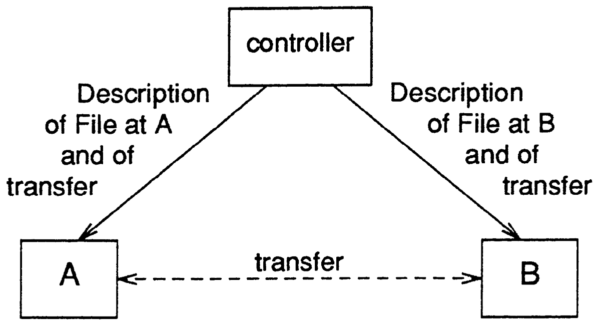

controller
Description of File at A and of transfer
Description of File at B and of transfer
A
transfer
B

Figure 2a - Logical flows of information in file transfer

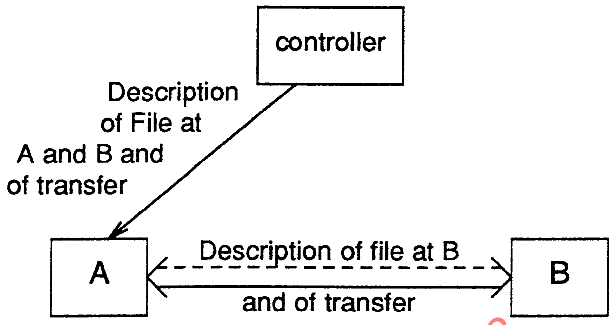

controller
Description of File at A and B and of transfer
A
Description of file at B and of transfer
B

Figure 2b - Actual flows of information in FTAM

Figure 2 — Logical and actual flows of information in file transfer

Part of the work to be done in designing an implementation is the choice of a mapping between terms in all levels of the set of definitions and aspects of the implementation. This process is shown in figure 1.

8 Nature of the file service

8.1 Control of file activity

The way file activities are controlled needs to be explained in order to clarify the aims of ISO 8571. Consider a file transfer or file access; for any transfer or access, there are three entities involved: an entity which takes the controlling initiative, an entity which accesses the source virtual file and an entity which accesses the destination virtual file. From the controller, there are two flows of information:

a) information, concerned with the specification of the source virtual file and with constraints on the way the transfer is to be performed, sent to the entity accessing the source file, and

b) information, concerned with the specification of the destination virtual file and with constraints on the way the transfer is to be performed, sent to the entity accessing the destination file.

These flows and their relation to the transfer are indicated in figure 2a.

To simplify the co-ordination and control of the transfer, it is assumed that the controller will channel these flows via one of the two file protocol entities, which will act as its agent in performing the transfer (see figure 2b). In many cases this will arise naturally, because the controller and the initiating file entity will be within the same system.

The service defined here is the one which supports the interconnection of the elements at A and B in figure 2b, and A and B are the users of the service referred to in the service definition.

8.2 Asymmetry of the dialogue

The actions to be supported by the file protocol show some important asymmetries which are reflected in the service and protocol structures. The asymmetries take the form of master to slave relationships.

Firstly, each activity is started by one file service user (the initiator, A in figure 2b), who has some established aim to achieve. The entity associated with the filestore (the responder, B in figure 2b) merely reacts to this initiative in a passive role. This applies even when a file is being transferred from one filestore to another, because the protocol does not need to carry information about the filestore at the initiator. The file protocol carries only information about the filestore at the responder. The act of transferring file access data units to the filestore at the responder can be considered as being performed by a copying application which has local access to one file and remote access to the other (see figure 3).

The second asymmetry is the more basic one that, when transferring file access data units, one particular entity is the sender, and the other is the receiver; at any instant during data transfer there is a preferred direction of data flow.

8.3 External file service and internal file service

The communication identified between the initiator and the responder in 8.2 can itself be divided in a modular fashion by separating out the error recovery mechanisms, yielding two distinct service levels (see figure 4).

These are

a) an external file service, in which the user states its 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) an internal file service used by the file 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

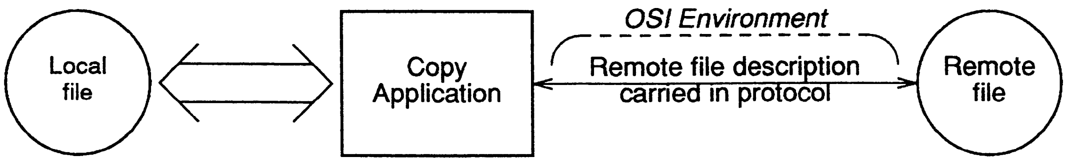


Figure 3 — Example of the dialogue between file entities

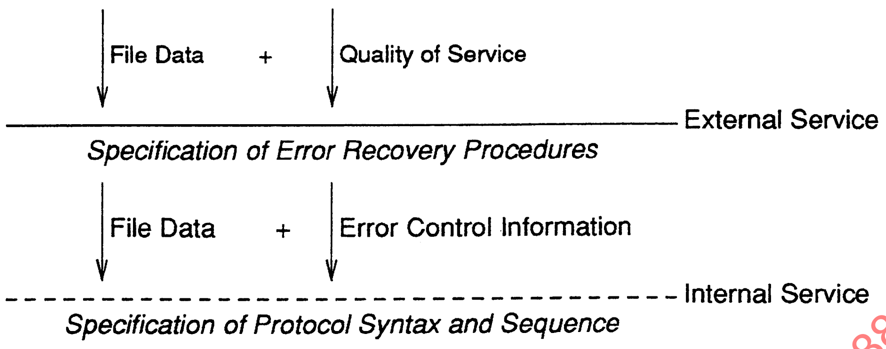

File Data
+
Quality of Service
External Service
Specification of Error Recovery Procedures
File Data
+
Error Control Information
Internal Service
Specification of Protocol Syntax and Sequence

Figure 4 — Structuring a file protocol entity

these procedures is the user of the internal file service. The choice of the error recovery procedures to be used is based on analysis of cost from the FTAM and Communication Qualities of Service requested in the external service and on local management information.

8.4 Service classes and functional units

The FTAM protocol and service are very rich in function. A wide range of types of application can be supported, but the cost of implementing all parts of the protocol on all occasions would be correspondingly high.

If implementors were given arbitrary choice as to which of the functions were to be provided, there would, on the other hand, be little chance that similar choices would be made, and so little probability of communication.

To avoid these problems, the FTAM service defines two types of functional selection. At the most basic level, the functions defined are grouped into functional units. An implementation must support a functional unit either completely or not at all; the number of choices to be made are thus limited. Mechanisms are defined which allow negotiation of functional units when the FTAM regime is initialized, and these allow the two communicating entities to reach a common understanding of the set of functional units available.

This in itself reduces variety, but still leaves considerable freedom. Further convergence is achieved by defining service classes, each of which support broad categories of use. These classes are

a) the transfer class, which allows for the movement of files or parts of files between systems, placing emphasis on simple operation with a minimum of protocol overhead before and after the data transfer;

b) the management class, which allows control of the virtual filestore by a series of independent confirmed service exchanges, but does not include file transfer mechanisms.

c) the transfer and management class, which combines the features of the transfer class and the management class;

d) the access class, which allows the initiating entity to perform a sequence of operations on the file access data units, providing for the manipulation of remote data.

e) the unconstrained class, which leaves the selection of functional units to the designer of the distributed application, giving full flexibility for optimization, but no guarantee of a common functional kernel.

The classes of service which are needed are negotiated when the FTAM regime is initialized.

9 Functions associated with the file service

9.1 Control of actions

The virtual filestore defines the actions which can be performed on a file. In any particular situation, only a subset of these actions are available. This subset is determined by

a) the file attributes (see section two); these indicate the actions appropriate to the file contents (in terms of the constraint set applied) and the local storage mechanisms (by means of the permitted actions file attribute) and express any access control constraints affecting file access;

b) the current state of the filestore, particularly the constraints implied by any concurrent access in progress to the same file;

c) the values of the activity attributes established by parameters of the file service when the data transfer regime was being negotiated.

Each of these can restrict the set of actions available. The process of refinement, leading to the set of actions which can actually be performed, is illustrated by the Venn diagrams in figure 5.

The first diagram shows that the actions appropriate to a file are those which are allowed by the file structure as expressed by the structural constraint set, the permitted actions attribute and the access control attribute. The result is a set of actions which could in principle be performed on the file.

The resultant set of actions is then further reduced to take account of any system constraints applying to the filestore; the most important of these for file access are the constraints imposed by the concurrent activities of other users.

Finally, the initiator requests a set of actions while building up the data transfer regime, and at each stage the negotiation of parameters may result in a restriction either of the set of actions requested, or of the set the system would allow, or both. When the association is initialized, actions are limited by the service class and the set of functional units negotiated. When the file is selected, a subset of these actions may result from the permitted actions agreed. Lastly, when the file is opened, the actions to be used are declared in the stated processing mode.

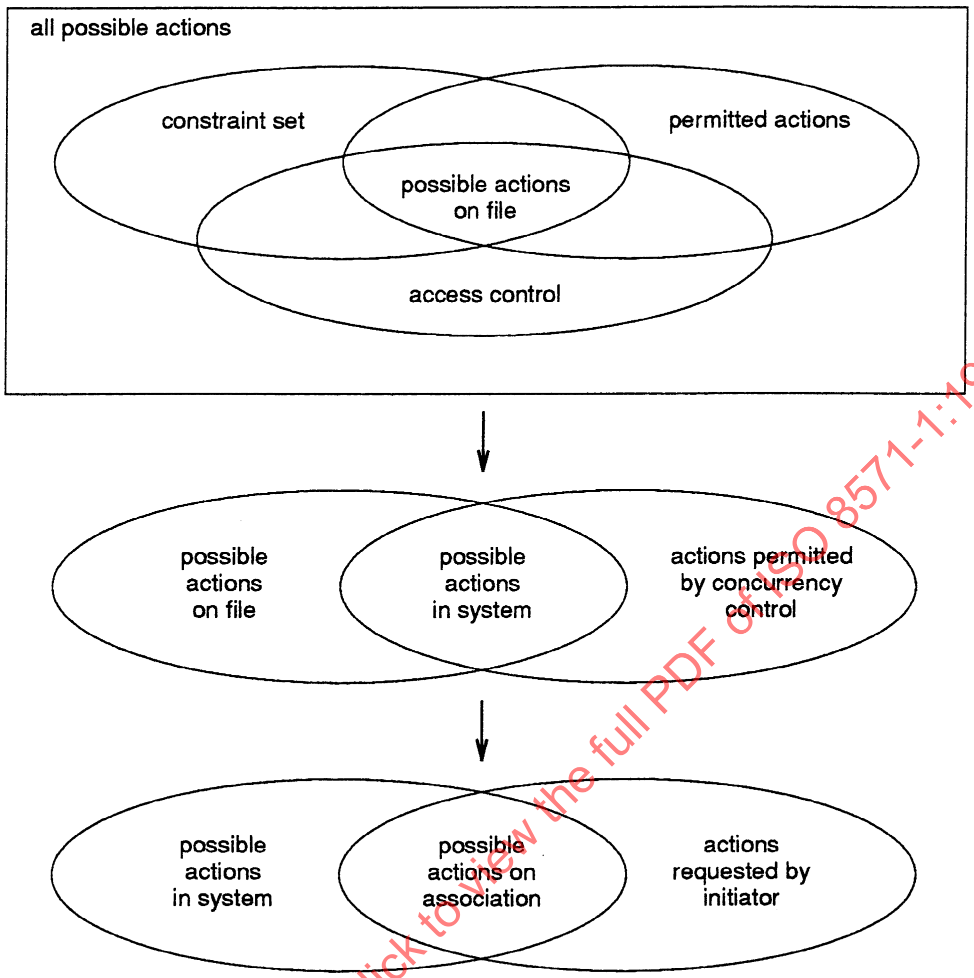

all possible actions
constraint set
permitted actions
possible actions on file
access control
possible actions on file
possible actions in system
actions permitted by concurrency control
possible actions in system
possible actions on association
actions requested by initiator

Figure 5 — Restrictions on the possible actions

9.2 Accounting

The FTAM service defines a basic mechanism for the carriage of accounting and charging information. Account names may be associated with a file to cover the costs arising from its storage, and accounts may be associated with file regimes to cover the costs of access to the information in the file.

Corresponding charging parameters allow the costs incurred against these accounts to be reported when a regime is terminated. An account may be set up when an FTAM regime is initialized, but this may be over-ridden when a particular file is selected to allow its usage to be charged against a separate account if necessary.

These mechanisms allow for the exchange of accounting and charging information, but the accounting model to be used, and the mechanisms for management of budgets and allocations are left outside the scope of ISO 8571; the issues involved cover many activities, and FTAM is but one example of their use.

9.3 Concurrency control

The objective of the concurrency control mechanisms is to ensure that an initiator has a consistent view of the file by restricting shared access. These mechanisms are designed to provide a way for a user to perform a coordinated series of actions without interference from concurrent accesses.

Individual filestore actions are implemented in such a way that any single action, such as a read or a write data transfer, appears atomic and serializable. This means that although the sequence of actions on a set of associations is not in general predictable, actions on one association are seen from other associations only when they are complete.

Concurrency control is provided solely to control the correct parallel execution of multiple tasks. Its effects must be distinguished from those of access control, which provides security mechanisms, and specification of permitted actions, which indicates the capabilities of the system containing the filestore. Any particular action must be consistent with all three of these control mechanisms before it can be performed.

Two levels of concurrency control are provided. The outer level controls access to the whole file, while the inner level can be used to allow separate control of the access to individual file access data units. The outer, or file, level can be applied when the file is selected or created and released when the file is deselected or deleted. It can also be modified for the duration of the file open regime if, for example, the file is first to be read and then re-opened for updating. File concurrency controls are evaluated at the time the file is selected or opened, and persist until the file is deselected or closed.

File concurrency is applied separately to each type of action which can be performed on the virtual filestore. There is therefore independent control of the read, insert, replace, extend, erase, read attribute, change attribute and delete file actions. This provides a very powerful set of tools for the construction of application oriented concurrency schemes, including both the familiar ones and schemes tailored to particular applications. However, precisely because of their power, not every configuration expressible with these tools will find practical use, and intelligent application wide design is needed. For every file selection or file open regime, each action is either defaulted or specified in one of the following categories:

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

b) shared: I can perform the operation - so can others;

c) exclusive: I can perform the operation - others cannot;

d) no access: no one may perform the action.

Thus, for example, a distributed application may specify shared read access, and no access to other actions to protect overall data integrity. Any number of users can then read data in normal operation, but if a management entity claims the file requesting exclusive replace and insert access to modify its structure, the activity cannot begin until all other access has ceased, and prevents further access by other users until it has itself completed.

However, if all users requested shared read and exclusive replace for the whole file, only one user at a time could access the file because of the replace restriction, and this would defeat concurrency.

To allow a finer level of granularity, the second type of control, file access data unit (FADU) locking, is introduced. Use of this mechanism for locking individual FADUs is negotiated by selection of the corresponding functional unit when the FTAM regime is initialized; it is then requested for a particular file when that file is opened.

If the FADU locking is requested, concurrency controls specified at the time the file is opened are not applied immediately (although any concurrency control requested at file selection remains in force). Instead, the control requested is applied for the duration of each file access action requested. Thus, for example, a request to read a FADU might bring into effect for the duration of the read action a shared concurrency control requested when the file was opened.

In addition, individual file access data units can have a lock applied for some period within the file open regime, bounded by a pair of file access actions which are suitably marked by a parameter.

The lock on a file access data unit can have two states - on or off. While the lock is off, the actions specified as shared for the file as a whole can be performed concurrently with other similar accesses. However, when the lock is on as a result of an action by a user, that user takes exclusive control (or no access in place of actions not previously required) of the whole of the file access data unit concerned. A user is unable to take control of a FADU if some smaller FADU within it is locked by another user. A user is also unable to take control of a FADU if it forms part of a larger FADU which is locked by another user.

Locks may be turned on or off in association with the locate or data transfer actions; requests to turn a lock on are honoured before the associated action is performed, while requests to turn them off are honoured afterwards. Actions specified at file selection as requiring exclusive control (or no access) for the file as a whole are not affected by file access data unit locking.

Some systems may find difficulty in implementing the full range of concurrency control possibilities. Such systems may implement more restrictive controls than requested, such as deciding as a local matter to force all actions to have the same level of restriction as the most restrictive request, but their users will observe a corresponding degradation in the quality of service.

9.4 Access control

The access control mechanisms provided by ISO 8571 are based on the concept of an access control list. Each of the entries in this list gives a set of actions and concurrency constraints, and a set of tests which an initiator needs to satisfy before these filestore actions can be performed; the actions are allowed if the conditions given by any one of the entries in the list are satisfied. Thus a list might contain entries each allowing a number of named initiators to read a file, and a separate entry allowing any entity quoting a particular password to read from and write to it.

In addition to the actions given in the entry, allowed concurrency control combinations can also be included (see 9.3). If they are not included, the performance of concurrency control is determined locally by the filestore.

The concurrency control of actions on the file is at a finer level than a simple yes or no; this is reflected by a finer level of access control. An initiator may wish to perform an action while sharing the file with other accessors, or it may require exclusive access. The initiator may even request that no other entity is allowed to perform an action which it is not itself authorized to perform. This is supported by the access control file attribute, which records the ability to exercise each of the concurrency control options separately for each action. A concurrency control is allowed only if so indicated for the particular form of access requested. However, an initiator is always allowed to indicate that, for it, access is not required.

For example, an initiator may be allowed to request shared read access to a file and be given the ability to specify no access for anybody to the delete action for the duration of the read, even if the delete action is not itself accessible with either shared or exclusive access.

The list entries each specify an allowed set of operations, and may specify the identity and location (in terms of its application entity title) of the initiator, and any necessary access passwords.

In establishing the FTAM regime and the file selection and open regimes, values are established for various activity attributes corresponding to the possible items in the list. In particular, the initiator asks to perform a certain set of actions by setting the current access request activity attribute when establishing a file selection regime. Before allowing the requested actions, the responding entity scans the access control list to determine if the activity attribute values match any of the entries. If a match for the set of actions is found, and the associated tests are satisfied, the actions can be performed; if no match is found, the request is rejected.

Thus in summary, the access control list is a permanent property of the file, and is stored for as long as the file exists; an access check is made against this list whenever a regime implying access is set up; the access granted remains valid for the duration of that regime.

9.5 Commitment

ISO 8571 contains, in the error recovery protocol, mechanisms for ensuring that the requested data transfer is performed successfully, even following a wide range of communication or systems failure. However, there are requirements for the performance of groups or series of actions in a distributed application such that the whole group is either completed, and known to be completed, or rolled back so that no effects remain.

ISO 9804, Commitment, Concurrency and Recovery (CCR) defines mechanisms to achieve these effects under certain conditions. Applications can be constructed combining CCR and FTAM, and rules are defined to do this in the case of file transfer. This is a special case of the more general symbiotic sharing of associations which the FTAM protocol allows.

10 Service providers supporting FTAM

10.1 ACSE - application contexts and the FTAM environment

The application association and related application context needed to support the file protocol are established by the association control service element, defined in ISO 8649. The necessary properties of the association and context are expressed in terms of the service primitives initializing and terminating the instance of the FTAM regime.

The file protocol specifies that these requirements be met by the establishment of a new association. Indeed, in the initial ACSE standard, only association establishment and release are defined. However, when future ACSE functions allow an existing application association to be refurbished in order to provide the FTAM environment, such mechanisms will be an equally valid way of starting and ending the FTAM regime.

The properties of an application association constitute an application context which has a name. In general this name will identify many different aspects of the application, use of FTAM being only one. However, a specific application context name is defined in ISO 8571-4 for use when the primary intent is to transfer files as an activity in its own right.

At any instant, the file protocol operates so that there is one file activity in progress over a particular association; if more than one file activity is necessary, more than one association is established. When the file activity is complete, the decision whether to start another, or to release the association and its supporting connections, or to modify the application context to provide some other kind of service is a local management decision.

10.2 Presentation Service

FTAM protocol control information and file data are communicated by use of the Presentation Service, defined in ISO 8822.

The aspects of the information meaningful to an application are independent of any particular encoding, and are expressed by using the concept of abstract syntaxes. These essential aspects are described by the definition of data types as part of the specification of the abstract syntax; they are the only aspects which need to be referred to in the specification of the application.

The Presentation Service manages the representation of information meaningful to the application entities. Considering all the elements involved in the representation of information communicated, three representations can be identified. Firstly, there is a transfer syntax, which is a representation of the information communicated between the open systems, and then for each real system there is a representation of the information used within that real system. All three representations of the information represent a single common abstract syntax, corresponding to an established presentation context. However, any presentation process needs to be aware of only two of these representations - the transfer syntax and its own local representation. The protocol specified between presentation entities is only concerned with the transfer syntax.

For any particular application, certain aspects of the information transferred are significant; the others are only conventions adopted to allow these significant aspects to be carried. The distinction between significant and insignificant aspects is a property of the application and different applications may make different distinctions.

For example, an application which transfers text messages may be concerned with the sequence of words that forms the message, but not with their precise layout on the printed page. On the other hand, an application which transfers text for typesetting will be concerned with every detail of the layout of the text. Similar considerations apply to non-text transfers.

The aspects of the transfer which are significant to the applications are expressed by the abstract syntax; from the application point of view, each abstract syntax corresponds to a single presentation context, and one or more presentation contexts may be in use at any one time. When a presentation context is established, a suitable transfer syntax is negotiated between the presentation entities. Any representational detail not explicit in the definition of the abstract syntax will be negotiated by the presentation service provider. The transformations performed by a presentation entity are restricted so as to preserve all aspects of the data types defined in the abstract syntax.

A set of presentation contexts is established when the FTAM regime is initialized, and this may be sufficient for activities to be undertaken. If a wide range of file types is to be handled, context management facilities in the presentation service can be requested to allow the defined context set to be altered to provide the necessary presentation context for each file when it is opened.

When communicating individual data elements, the application provides the presentation service with the presentation data values to be transferred, the data types to which they belong and the abstract syntax from which these are drawn; the negotiated transfer syntax for that presentation context is then used. The selection of the FTAM application protocol implies use of the abstract data type required to express this particular protocol's control information (the FTAM PCI abstract syntax). ISO 8825 specifies rules for establishing a transfer syntax for this protocol control information from the abstract syntax which all FTAM implementations are required to support. They may, in addition, support other standard or enterprise specific transfer syntaxes.

10.3 Session Service

The entities in the presentation layer which support the FTAM activity will themselves communicate via the Session Service, defined in ISO 8326. The Session Service provides means of structuring the communication dialogue, which are passed on to the application entities by the presentation entities.

The Presentation Service can provide, by means of the underlying Session Service, synchronization point insertion and resynchronization services to support file checkpointing and recovery. These services allow the insertion of checkpoints into the flow of file user data, the purging of the session connection after an error and the resynchronization of the session synchronization point mechanisms before the data transfer is resumed.

The specification of the Session Layer also requires the FTAM protocol implementation to keep track of a session token controlling which entity is able to issue synchronization points.

Section two: Virtual Filestore — General Concepts

11 Virtual filestore

11.1 Need for a filestore model

The ways in which real filestores are implemented vary considerably between existing real systems. Different real systems have a wide range of styles for describing the storage of data and the means by which it can be accessed. A common model for use in describing files and their attributes is needed before services, protocols and procedures for file transfer, access and management can be used in the Open Systems Interconnection Environment. This model is called the "Virtual Filestore".

Such a filestore definition is very powerful, because it allows the differences in style and specification to be absorbed into a mapping function from the open system onto the real open system, and any particular real open system can then interwork with other different real open systems in terms which can be mutually understood. By screening the details of real systems from the user of the external connection, the need to modify the existing real systems and hence the initial cost of Open Systems Interconnection is reduced.

Similarly, a great variety in the way the file service is used needs to be encompassed. The file protocol implementation which acts in the role of the initiator may be invoked directly by a human user, by a subsystem processing a queue of submitted file requests, or by a user written application program. The file protocol implementation acting on behalf of the responder may directly access a real filestore or interface to a user written application program. In all these cases the same file protocol will be used.

Expressing the dialogue in terms of the Virtual Filestore allows interconnection of a wide range of real systems of different complexity. The definition of a number of optional functional units within the file service definition and a number of optional attribute groups within the virtual filestore definition make possible styles of working in which simpler real open systems interwork with more sophisticated systems; for instance, a complex computer system could communicate with the auxiliary storage of an intelligent terminal, or with a unit-record peripheral. The method serves not only to conceal differences of style between similar kinds of data storage, but also to resolve differences of type or sophistication. However, conformance statements are required which state which of the attributes defined in ISO 8571 are actually supported, so that a potential user can confirm that any necessary properties, such as security or concurrency control are supported by the real filestore.

11.2 Mapping the virtual filestore definition

To use the file service and file protocol, an implementation has to relate the elements of the virtual filestore definition to the real storage system available.

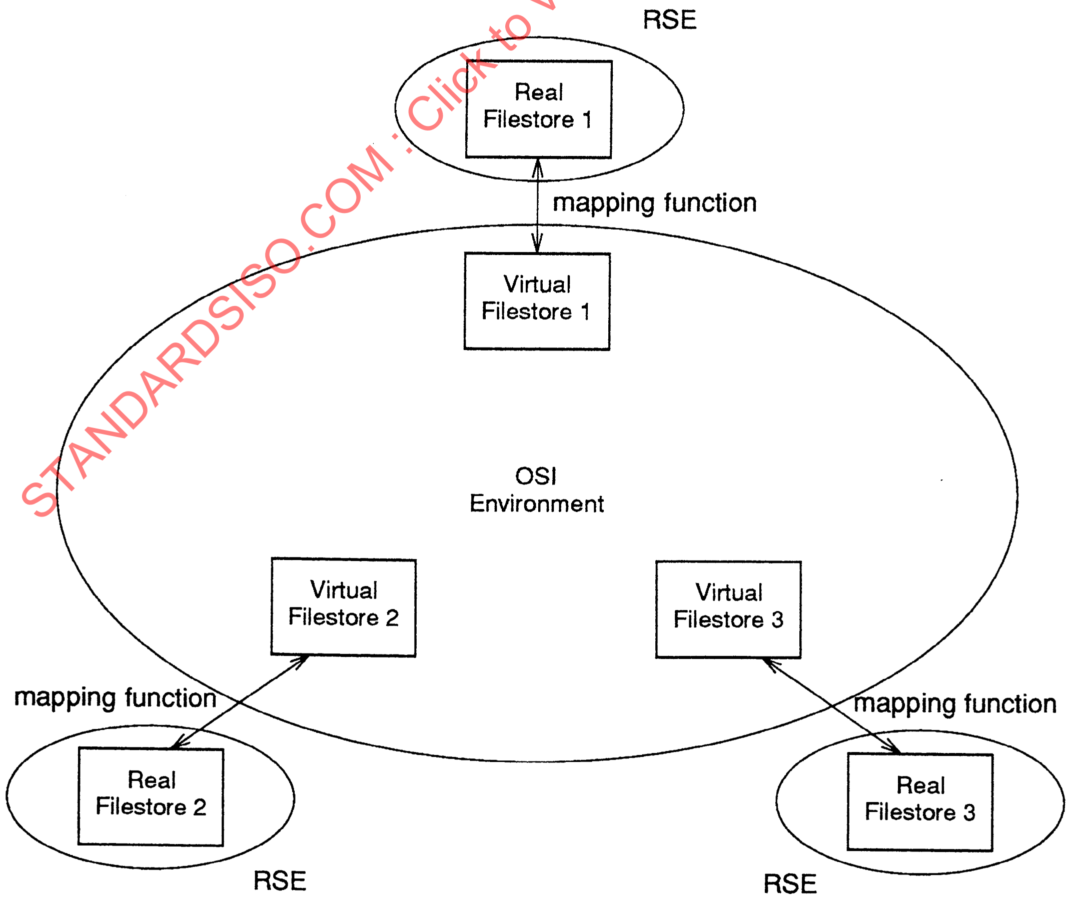


Figure 6 — Mapping between real systems and open systems

An implementor using ISO 8571 sets up a mapping (see figure 6) between

a) the actions, file access data units and file and activity attributes defined in the virtual filestore in the Open Systems Interconnection Environment and

b) resources in the Real System Environment.

When the protocol messages are received by the application entity, they are interpreted in terms of the correspondences established by that entity between the virtual filestore components and actions and those aspects of the real system environment relating to information storage. The mapping is therefore a set of correspondences established by the designer of the real open system.

11.3 Form of the virtual filestore

The definition of the Virtual Filestore forms a schema for the description of filed information. In this description, a file is an entity having

a) a single filename, that allows it to be referenced without ambiguity;

b) other descriptive file attributes which express properties of the file such as accounting information, history, etc.;

c) file attributes expressing the actions capable of being performed on the file;

d) file attributes describing the logical structure and dimensions of the data stored in the file;

e) any file access data units forming the contents of this file.

These are all aspects of the file which can be observed by any authorized initiator. If two observers make the same enquiry about these aspects of a single file, they will obtain the same information about its properties, provided that no modification took place between the enquiries. These properties are called file attributes.

There are also activity attributes describing the relation between the file and a particular initiator, concerned with things like authentication, data transfer options, accumulated cost, etc.; there is an independent set of values for these activity attributes for each activity in progress. This set is created after the FTAM regime involving the filestore is initialized, maintained while it persists, and destroyed at latest when it is finally released.

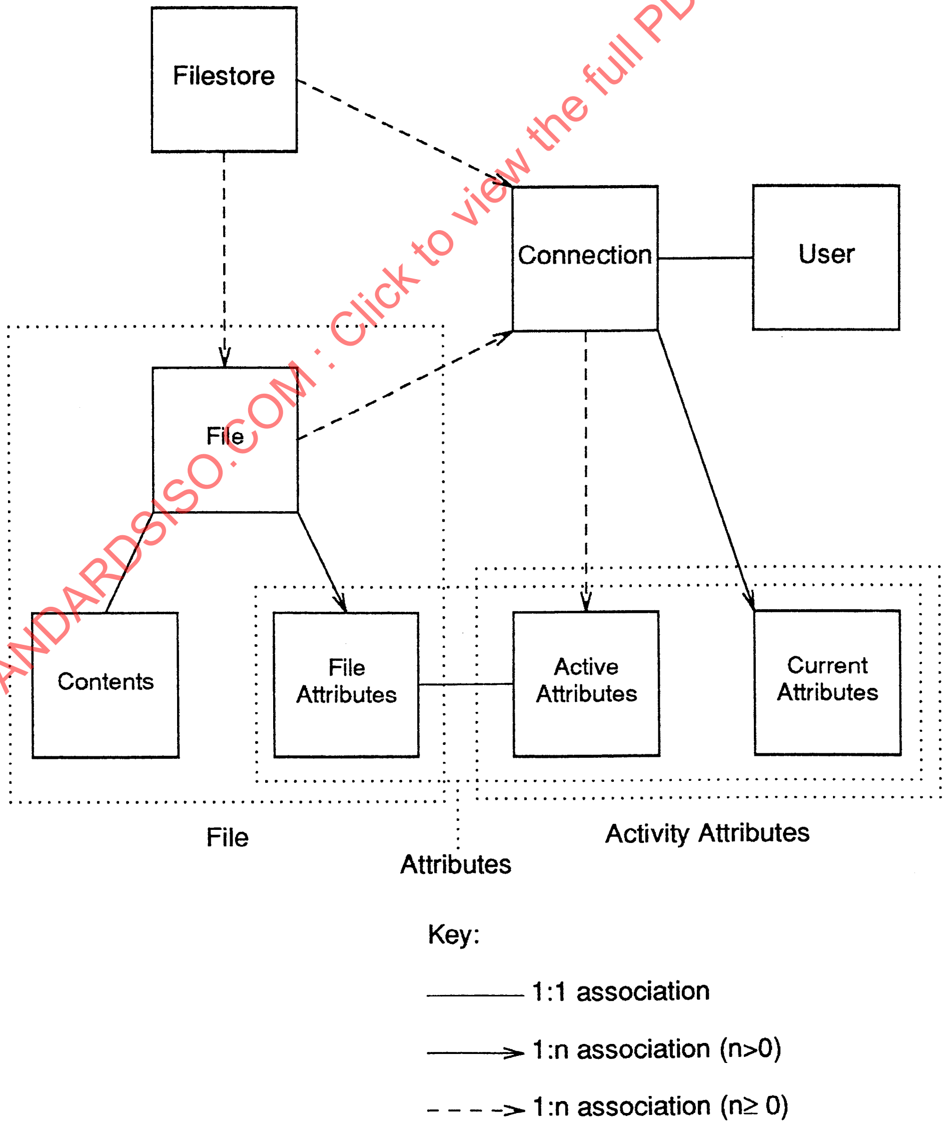


Figure 7 — Virtual filestore schema

Some file attributes place constraints on the structure of the file's content. This structure is preserved during the lifetime of the file. However, not all users accessing the file are concerned with its full generality. For instance, there may be a need to access a complex hierarchical file as if it were flat in order to construct summary reports, or it may not be necessary to access the smallest structural units of a file independently on all occasions. In addition to the file attributes used in file creation and file management for describing the permanent file access structure, there is a specified access context indicating, when a read data transfer is requested, the subset of the file structuring information and user data from the file access data unit to be transferred.

11.4 Attribute dynamics

The file attributes reflect the state of the file as actually stored. Communication between the initiator and the responder builds up an incomplete shared picture of these file attributes, reflecting the knowledge about them which the initiator has gained from this particular communication. Thus in principle, for each file attribute, there exists a corresponding active attribute expressing the information about it which has been passed to the initiator via the parameters of the FTAM service primitives.

Independently of these are the current attributes which describe the established FTAM regime itself; these include the statement of the identity and location of the initiator and the results of negotiations at the start of the various regimes which have been established.

The active and current attributes together are called the activity attributes. It is these activity attributes which determine the operation of the file protocol machines.

11.5 Filestore schema

The preceding description of the form of the virtual filestore may be expressed in the form of a schema relating the various concepts and the nature of their interrelationships. This schema is expressed graphically in figure 7.

12 File structures

12.1 Categories of Structure

A file exists to contain zero, one or more identifiable Data Units. These Data Units are related in a logical fashion. The filestore model defined in ISO 8571 provides a tree structure to represent the relation between the Data Units, for access and identification purposes. This tree structure is called the file access structure. Each node of the file access structure has associated with it zero or one Data Units. The structure of the information from the file service user's point of view may be different from the file access structure and the file service user must map from its semantics to the file access structure.

NOTE - In general these relations may be sequential, hierarchical, network or relational. The need for models other than the hierarchical one is recognised. Such models are not included in ISO 8571 , but future addenda or revisions may include additional models (for example, for network or relational databases) or refer to models defined in other standards.

The unit in terms of which operations on the file's content are performed is a File Access Data Unit (FADU), which may itself be structured, and contains Data Units with a defined abstract syntax. The subset of the file access structuring information and user data transferred in a particular read action is determined by the access context specified for the action.

There are four aspects of the file structure, each conveying different information about the file:

a) the file access structure - describes the composition of the file from file access data units;

b) the presentation structure - describes the abstract structure of the data units which are defined within the file access structure;

c) the transfer structure - describes the serialization of the file access data units for communication purposes;

d) the identification structure - describes the naming of the nodes in the file access structure, and the identification of file access data units to be transferred.

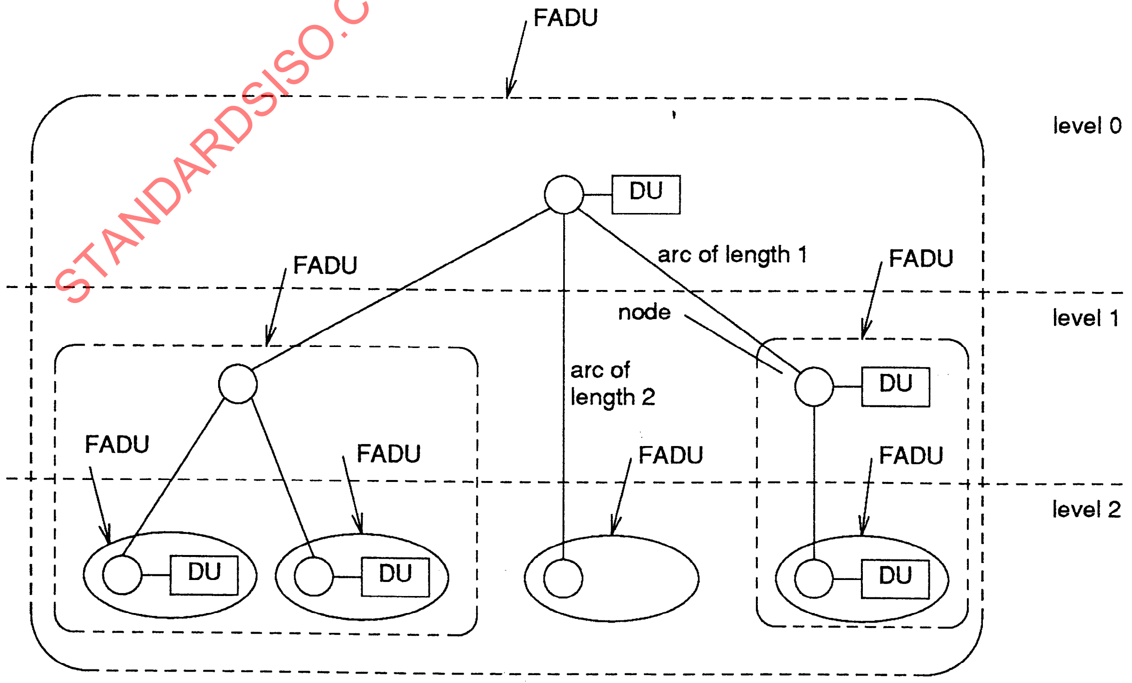


Figure 8 — An example of a file access structure

12.2 File access structure

The file access structure is primarily a static view of the file. Figure 8 shows an example for a file access structure of a file with a tree representation. The file access structure determines which parts of the file may be accessed independently using FTAM.

Use of a hierarchical structure for the file access boundaries does not prevent the application from representing non-hierarchical logical structures by use of references within the file access data unit contents.

12.3 Presentation structure

The presentation structure describes the relationship between data elements, data units and file contents. From it, the rules are derived for the transfer of the information contents of a file using the Presentation Service. The following description indicates how the rules for dividing the information contents of a file into data units and data elements operate. The detailed specification is given in ISO 8571-2 and in any document type definition applicable to the file. In other words, the description indicates how the record structure of the real files can be mapped onto the FTAM file structure, and how the presentation data values are mapped onto P-DATA primitives (PSDUs). It assumes a file defined using ASN.1.

a) The file contents and structuring information are made up of a series of Data-Elements.

b) There is a one to one correspondence between the Data-Elements defined in the ASN.1 Module ISO8571-FADU (defined in ISO 8571-2) and the data values passed to and from the FTAM service provider via the F-DATA service primitive.

c) In these Data-Elements, the File-Contents-Data-Elements (the user data) are normally defined as some ASN.1 type.

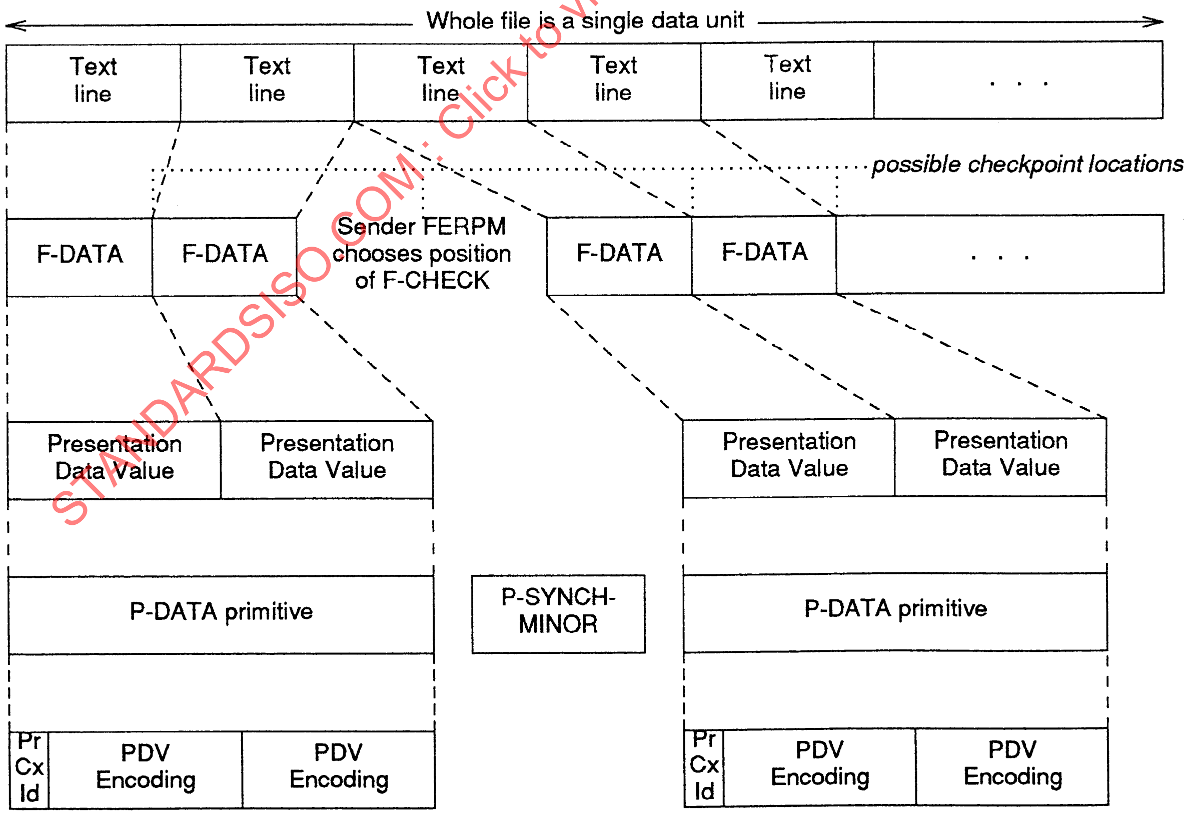
d) If a real file with a record structure is to be transferred as an unstructured file, the base type will typically be an ASN.1 CHOICE of the different record types in the file (if the file contains only one record type, the CHOICE will, of course, disappear). If access to individual records is to be provided by FTAM, it will be specified that only one value of the base type shall be in each Data-Unit.

NOTE - For example, for the document type FTAM-1, an unstructured text file, the type of the File-Contents-Data-Elements is GraphicString and it is understood that each GraphicString corresponds to one line of text.

e) The presentation protocol allows the assembly of the presentation data values into suitable groups for transfer. The degree of this grouping is an implementation option of the sender; it is a local matter of optimizing transfer overhead. The receiver does not know in advance how many values of the base type will be put into each P-DATA primitive.

f) Checkpoints can only be inserted between P-DATA primitives, so the amount of grouping mentioned under (e) is determined by the position of checkpoints inserted during the transfer of a file.

g) Note that, in detail, a structured file is transferred as a "series of values of type Data-Element" (see ISO 8571-2). Each Data-Element corresponds to one data value of F-DATA, and several of these may be mapped onto one P-



FERPM	File Error Recovery Protocol Machine
PDV	Presentation Data Value
Pr Cx Id	Presentation Context Identifier

Figure 9 — Transmission of an unstructured text file

DATA primitive (see ISO 8571-4). The series of Data Elements is described using ASN.1, but there is no encoding generated in the transfer stream for the SEQUENCE and SEQUENCE OF occurring in the expansion rules for "Subtree", "Children" and "DU", since these structural definitions form a secondary entry point and are not referenced from the file protocol.

The relationships between data elements, file services, presentation data values and the primitives of the presentation services and their encodings are indicated in figure 9, which shows the example of the transfer of an unstructured text file.

13 Constraint sets

The general hierarchical structure described in clause 12 can represent a very wide range of different practical file structures. However, any single application will only need a specific set of structures and any real system is only likely to support a limited range of file types, with restrictions on the way files can be modified. To express these limitations, the concept of a constraint set is introduced. A constraint set states limitations on the range of structures allowed, and defines how the basic file access actions can modify the structure without changing its essential nature. Constraint sets reflecting certain common file types are included within ISO 8571, but other constraint sets may subsequently be defined and registered.

Each constraint set is identified by an object identifier value, established either in ISO 8571, or in some other International Standard, or by any of the other mechanisms defined in ISO 8824.

ISO 8571 defines constraint sets to model a number of widely used structures which are subsets of the general hierarchical structures. For instance, different constraints on node naming reflect different classes of indexed structure. The constraint sets defined are:

a) unstructured, in which there is a single data unit, without a name;

b) sequential flat, in which there is a series of unnamed data units; this constraint can be used, for example, to model Fortran sequential input-output;

c) ordered flat, in which there is a series of named data units; the handling of data units with duplicate names is defined;

d) ordered flat with unique names, in which there is a series of uniquely named data units;

e) ordered hierarchical, in which the general hierarchy is allowed, in order to model multiple indexes; insertion is based on the position in these indexes;

f) general hierarchical, in which the general hierarchy is allowed, with full control on the placement of new nodes when the structure is modified;

g) general hierarchical with unique names, in which the requirement for unique naming is added to the general hierarchical constraint set.

14 Document types

A file's contents can be of many types. It is possible to specify the file model, the constraint set and the abstract syntax of the data units separately. However, this is a cumbersome procedure, and a single way of specifying the semantics, abstract syntaxes, transfer syntaxes and structural dynamics is preferable; this is done by the definition of document types.

Document type definitions may be qualified by a group of one or more parameters. These parameters effectively allow the definition of a family of closely related document types at one time. For example, a single definition may cover a number of types of text document, with the precise character set or maximum line length being indicated by parameters.

For some applications, a restricted view of a structured file may be sufficient while reading data from it. A document type definition can include a statement of other document types which are derivable from it with some loss of information; this process is known as simplification of the original document type. Similarly, the definition may allow variation of some parameters to less restrictive values; this process is known as relaxation of the original document type.

A document type is registered with either the international or a national registration authority (see ISO 9834-2 for the definition of the registration procedures) because of either its general interest or because of its importance to a specific group of applications. The register entry is named with an object identifier value.

A document type consists, in general, of a collection of related statements, some of which are specific to its intended use. This collection includes

a) the identification of the document type and of any information objects it references;

b) its intended scope; document types referenced in the virtual filestore definition and used for the communication of files are of concern in ISO 8571;

c) any parameters which apply to the document type;

d) constraints on the semantics to be applied in interpreting the document;

e) the file access structure of the document, in terms of the file model and constraint set to be applied;

f) the abstract syntax or syntaxes of the structuring information and of the data units within the structure;

g) the way the information is formed into a series of presentation data values and the procedures for their transfer;

h) the transfer syntax or syntaxes in which the defined abstract syntaxes can be communicated;

i) the details of the way operations defined in particular ASEs manipulate the document type; for example, this may include the result of concatenating two instances of the document type or the ways in which the file access structure can be simplified by viewing it as an instance of a simpler document type.

The concept of document type can be applied recursively, to allow stepwise refinement to successively smaller classes of possible documents. Thus, for example, a document type can be defined which leaves considerable freedom in the range of abstract syntaxes allowed in the data units. This document type can then be referenced in a series of document types which share the same general properties but each add different restrictions to the abstract syntaxes to be used.

Section three: Overview of the file service and file protocol

15 File service

The file service and its supporting protocol are concerned with creating, in a series of stages, a working environment in which the initiator's desired activities can take place. The dialogue, in turn

a) allows the initiator and the responder (filestore) to establish information about each other, including their respective identities in terms of their application entity titles;

b) identifies the file which is needed;

c) establishes the attributes describing the file and the bulk data transfer which is to take place in this activity;

d) engages in file management;

e) locates the position in the file access structure of the FADUs to be accessed;

f) inserts, replaces, extends or erases one or more complete FADUs.

These steps build up various parts of a set of file contexts. The period for which some part of the common state held by the service users is valid is called a regime. As progressively more shared state is established, a nest of corresponding regimes is built up. However, there will, in general, be a time lag between the establishment of a regime at the two ends of the association.

A period of time in which protocol exchanges have a particular purpose, such as establishing or releasing an application context is called a phase. For each phase, a set of valid messages is defined in terms of state transitions. At any time, each entity participating in the FTAM regime is in precisely one phase; phases cannot be nested one within another; the activity progresses through a series of phases.

The phases are introduced in the following sub-clauses, in the order in which they would occur during a typical use of the file service.

The nesting of regimes is shown in figure 10.

15.1 FTAM regime initialization phase

The FTAM regime initialization phase is the first in any instance of the file service. It establishes the application context and the initial state of the FTAM regime, ensuring that the application association which underlies it links the correct addresses and negotiating the service class and functional units which are to be available. It also establishes FTAM specific information such as the authorization and accounting information necessary for the ensuing operations on the filestore.

15.2 Filestore management phase

Operations in the filestore management phase are for study in preparing future addenda to ISO 8571.

15.3 File selection phase

The file selection phase establishes the file selection regime; it identifies or creates a unique file to which operations in subsequent phases will apply. The selection continues until explicitly unset by a deselection phase or a termination phase. The operations performed in the following phases therefore refer to the selected file and do not contain explicit file identification. The selection phase may involve the creation of a new file with specified properties.

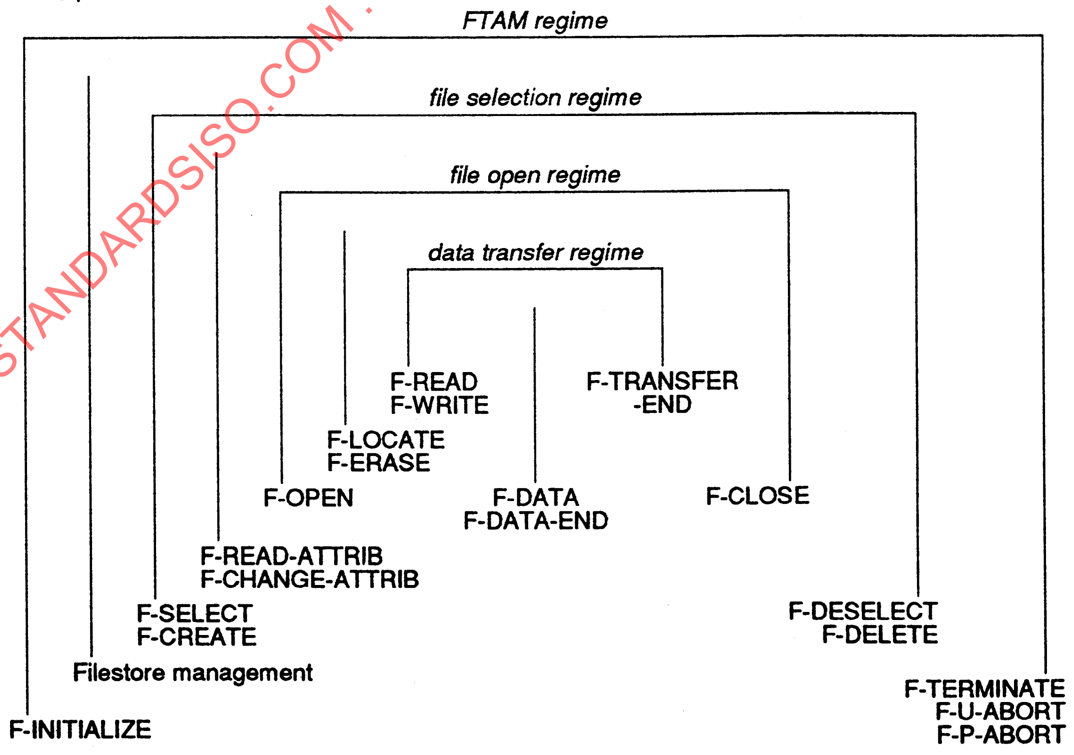


Figure 10 — File service regimes and related primitives

File selection in ISO 8571 is in terms of a filename. A file selection request may include file specific access control information.

NOTE - In some future addendum to ISO 8571 file selection may be expressed as a series of constraints on file attributes other than the filename (see ISO 8571-2) whose values serve to identify the file required.

15.4 File management phase

Operations in the file management phase allow the file service user to perform file management actions on the selected file. For instance, some of the attributes of the file may be read or updated.

15.5 File open phase

The file open phase establishes the file open regime in which transfer of file access data units can take place. It establishes the data transfer facilities, including any special presentation contexts required for the transfer.

15.6 Data access phase

The whole file is a single file access data unit, but there can also exist smaller FADUs within it. Actions on the contents of a file act on FADUs. The data units consist, in general, of a series of data elements, but these cannot be requested or updated independently.

The data access phase consists of a period of time in which a number (for the transfer class only one) of operations are performed. These operations are either bulk data transfer operations or control operations. Each bulk data transfer operation consists of:

a) a statement specifying the operation to be performed, giving the type of operation and identifying the file access data unit to which it is to be applied;

b) any necessary data transfer;

c) a terminating exchange.

The control operations are simple commands and responses. The operations defined are

d) locate, which points to a node within the file structure; this node is the root of some file access data unit;

e) erase, which removes an FADU from the file structure.

15.7 File close phase

The file close phase ends the file open regime established by the open phase; it ends the period in which data transfer is possible. The close phase allows exchange of status information at the end of the file open regime. It provides confirmation that all requested data transfers have terminated, either successfully or in failure; if successful, the previous actions will not be revoked. Following a successful close, the file maintains the result of all actions performed during the data transfer phase, despite communication or application failures which occur subsequently. Failure after an open and prior to a successful close leaves the file in an undefined state; the state of the file may be re-established by error recovery procedures.

15.8 File deselection phase

The deselection phase releases the file selection established by the selection phase. The deselection phase may include standard accounting sequences. It leaves in effect any authentication or account identification established before the file selection phase. The deselection phase may involve the deletion of the selected file.

15.9 FTAM regime termination phase

This phase dissolves the regime shared by the initiator and the responder, releasing any authentication and accounting regime. There is then no further file-related regime remaining; the file service activity is over.

16 Mechanisms in the file protocol

16.1 Protocol state machine

The main actions of the basic protocol are concerned with the establishment and termination of a series of nested regimes. In the innermost regime, the data transfer is a simple unidirectional flow with possible insertion of checkpoints. The protocol implementation can therefore be described accurately by a straightforward state machine, closely modelled on the finite state descriptions given in ISO 8571-4.

The cost and complexity of an implementation correspond closely to the complexity of the state machine on which it is based. The service classes and functional units have therefore been chosen to give a simple state machine, with as little interdependence between functional units as possible.

16.2 Grouping of protocol data units

One way in which the state machine is simplified is by the introduction of mandatory grouping into the transfer service class. Conforming implementations are required to concatenate the protocol data units which establish the file selection and open regimes; a similar constraint applies separately to those which terminate them. A threshold parameter at the start of concatenation controls how much of the concatenation must succeed before any of the requested actions are taken. Implementations of the transfer service class are required to set the threshold parameter on the grouping so that either all the requested regimes are established or all the establishment requests fail (i.e. the actions are rolled back).

The result of this grouping is that the protocol data units concerned with selection, attribute management and file open can be considered, from the implementors point of view, as subfields in a single data transmission. The state machine for file regime initialization has only two stable states (initialized and open) and two pending states.

However, by making grouping optional in the access and unconstrained service classes, great flexibility to support a wide range of applications is introduced in an upwards compatible way.

16.3 Transparency

To the presentation service provider, both the file user data and the FTAM protocol control information are a series of presentation data values. In general, the ability to transfer arbitrary files without ambiguity between user data and PCI is required. This means that steps must be taken to prevent arbitrary series of presentation data values in the file user data, which might syntactically be taken as PCI being so interpreted.

The solution chosen in the FTAM protocol is to separate the two using the presentation context. The first presentation context used in the FTAM regime is deemed to be the FTAM PCI presentation context, and presentation data values in any other presentation context can clearly be identified as distinct from FTAM PCI by presentation protocol mechanisms, even if they are taken from the same abstract syntax and are syntactically well formed protocol data units.

Since the transparency mechanism is based on a presentation context, which is an instance of the abstract syntax in use, transparency is even maintained if the file user data is in the FTAM PCI abstract syntax; there would then be two distinct presentation contexts corresponding to this abstract syntax.

16.4 Checkpoint insertion

Checkpoints are inserted into the stream of file user data at points chosen by the sender to allow the operations of the restart and recovery functions. Both the sender and receiver must save information giving the position in the file of any active checkpoints (that is, checkpoints issued or received but not yet superseded by the acknowledgement of a later checkpoint).

In general, there is also a need to keep information concerning the point in the structure of the file at which the checkpoint is inserted. This falls into two categories:

a) semantic information implicit in the contents of the file, such as position in the file access structure or other relations between data units, such as pointers and identifier references; these need no extra information which is not already maintained with the file.

b) information on the state of the interpretation of the abstract and transfer syntaxes. If checkpointing were allowed anywhere, the state of interpretation of the corresponding transfer syntax would also need to be preserved. This is potentially additional information, not implicit in the stored file, because local syntactic transformations managed by the presentation service (such as re-ordering the fields in an application oriented record) may make it impossible to deduce the received state from the stored state, or indeed to store the information at all before the syntactic unit is complete.

To avoid the need for implementations to make special provision for this second type of information, the positioning of checkpoints is constrained so that they may only fall at the semantic boundaries of complete units defined in the abstract syntaxes.

For the FTAM file structuring information, this constraint is expressed in the ASN.1 notation by expressing the file structure and contents as a series of concatenated data elements, each of which is independently encodable. The protocol abstract syntax refers to these elements via a single symbol, whose expansion is a CHOICE of the available data elements. The semantic relationship from a structural point of view is defined using ASN.1 in terms of a data type representing the complete hierarchical file model, but this data type is never referenced in the abstract syntax of the protocol, and so generates no encoding.

Subject to these constraints, the checkpoints are placed at positions determined by the sender, which may or may not be related to storage boundaries in either system. The receiver must be prepared to accept checkpoints in any valid position in the file.

16.5 Diagnostics and results

The protocol provides two levels of information on the success or failure of the operations requested.

First, there is an indication at a very general level, intended to determine the behaviour of the protocol machines themselves; this is signalled by a pair of result parameters indicating:

a) the success of any requested protocol machine transitions, and

b) the success of any requested filestore actions.

For example, in a file deletion, the protocol transition of deselection always succeeds, but the filestore action of deletion may fail.

The value of the action result parameter may indicate: success, recoverable failure (which may initiate the error recovery protocol) or complete failure.

The second source of information is the diagnostic parameter, which contains more user oriented information, with detailed explanations and a more complex structure. Analysis of this parameter is not necessary for the progress of the protocol.

16.6 Docket handling and non-volatile storage

ISO 8571 defines error recovery procedures which allow a data transfer activity to be continued after a communication or system error which breaks the supporting presentation connection and application association. However, for recovery to take place, the body of information which describes the activity and the stage reached in the data transfer needs to be preserved intact. Errors which destroy the information in the system that states who it is communicating with and why cannot be recovered.

The body of information which must be preserved is called a docket in ISO 8571. The definition of this term does not imply that the implementation needs to concentrate the information in a single place, but merely indicates that the whole collection shares a certain level of permanence.

The type of storage used to hold the docket will depend on the reliability objectives set by the user of the file service. For any type of storage there is some small probability of error, even if only that associated with physical damage to the supporting equipment. The designer of the distributed application may choose any form of non-volatile memory technology which meets the system targets for overall reliability.

16.7 Error recovery mechanisms

The aim of the error recovery procedures specified in ISO 8571 is to provide an uninterrupted file service to the external file service users, correcting errors by exchanges between the protocol machines without the user being made aware that anything out of the ordinary has happened.

Errors may destroy the supporting communication, or one of the end systems may fail and be restarted; in either case the high level association between the communicating applications is maintained because both hold matching dockets. The error recovery protocol reconstructs the supporting connections and the state of the data transfer prior to the failure, based on the information held in the dockets, and the communication then continues.

The file service user gives the file error recovery protocol machine the information necessary for selecting the mechanisms just outlined by means of the quality of service parameter and by local means. The file error recovery protocol machine, acting as the user of the internal file service, may thus negotiate the use of the recover or restart functional units (or neither of them) for use in the specific association.