

INTERNATIONAL STANDARD

ISO
8571-2

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 2 : Virtual Filestore Definition

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

Partie 2 : Fichier virtuel

ISO 8571-2 : 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 8571-2 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 form an integral part of this International Standard. Annexes C, D and E are for information only.

Contents

	Page
0 Introduction	1
1 Scope and field of application	1
2 References	1
3 Definitions	2
4 Abbreviations	2
Section one: The filestore model	
5 Basic concepts.....	3
6 File selection	4
7 File structures	4
7.1 File access structure	4
7.2 Abstract structure definition.....	5
7.3 Abstract syntax definition	5
7.4 File transfer structure	5
7.5 Access context	5
7.6 Identification structure	7
7.7 Constraint sets	7
8 Actions on files.....	7
8.1 Relation to bulk data transfer	8
8.2 Read bulk data transfer.....	8
8.3 Write bulk data transfer.....	8
9 Attributes.....	8
9.1 Attribute scope	8
9.2 Scalar, vector and set attributes.....	9
9.3 Attribute values	9
9.4 Support of file attributes	9
Section two: Actions on the Filestore	
10 Actions on complete files	10
10.1 Create file	10
10.2 Select file.....	10
10.3 Change attribute.....	10
10.4 Read attribute.....	10
10.5 Open file.....	10
10.6 Close file	10

10.7	Delete file	10
10.8	Deselect file.....	10
11	Actions for file access	10
11.1	Locate	10
11.2	Read.....	10
11.3	Insert	10
11.4	Replace	10
11.5	Extend	11
11.6	Erase	11
11.7	File actions and current location	11
Section three: Attribute definitions		
12	File Attributes.....	12
12.1	Filename	12
12.2	Permitted actions	12
12.3	Contents type	12
12.4	Storage account	12
12.5	Date and time of creation	13
12.6	Date and time of last modification	13
12.7	Date and time of last read access.....	13
12.8	Date and time of last attribute modification	13
12.9	Identity of creator	13
12.10	Identity of last modifier	13
12.11	Identity of last reader.....	13
12.12	Identity of last attribute modifier	13
12.13	File availability.....	13
12.14	Filesize	14
12.15	Future filesize	14
12.16	Access control	14
12.17	Legal qualifications.....	15
12.18	Private use	15
13	Activity attributes.....	15
13.1	Active contents type	15
13.2	Current access request.....	15
13.3	Current initiator identity	15

13.4	Current location	15
13.5	Current processing mode	15
13.6	Current calling application entity title	15
13.7	Current responding application entity title	15
13.8	Current account	16
13.9	Current concurrency control	16
13.10	Current locking style	16
13.11	Current access passwords	16
13.12	Active legal qualification	16
14	Attribute groups	16
14.1	Kernel group	16
14.2	Storage group	16
14.3	Security group	17
14.4	Private Group	17
15	Minimum attribute ranges	17
Annexes		
A	File access structure constraint sets	19
B	Document types	28
C	Reading of structured files	41
D	Insertion in a structured file	43
E	ASN.1 cross reference	47

Figures

1 Relationship of files, attributes and associations	3
2 The Access Structure as a Tree Structure	4
3 ASN.1 definition of file contents	5
4 ASN.1 definition of file structure	6
5 Unstructured file transfer	41
6 Flat file transfer	41
7 Hierarchical file transfer	42
8 Source of transferred data	43
9 Initial state of destination file	43
10 Final state of insert as sisters to A	43
11 Final state of insert as children of C (normal)	44
12 Final state of insert as children of C (variant)	45
13 Final state of insert subtree as sister	45
14 Final state of insert subtree as child	46
15 Initial state of ordered flat file	46
16 Final state of merge of ordered flat files	46

Tables

1	Result of reading in an access context	7
2	Effect of actions on location	11
3	Concurrency control options	16
4	Activity attributes	17
5	File attributes	18
6	Basic constraints in the unstructured constraint set	20
7	Basic constraints in the sequential flat constraint set	21
8	Identity constraints in the sequential flat constraint set	21
9	Basic constraints in the ordered flat constraint set	22
10	Identity constraints in the ordered flat constraint set	22
11	Basic constraints in the ordered flat constraint set with unique names	24
12	Identity constraints in the ordered flat constraint set with unique names	24
13	Basic constraints in the ordered hierarchical constraint set	25
14	Identity constraints in the ordered hierarchical constraint set	25
15	Basic constraints in the general hierarchical constraint set	26
16	Identity constraints in the general hierarchical constraint set	26
17	Basic constraints in the general hierarchical constraint set with unique names	27
18	Identity constraints in the general hierarchical constraint set with unique names	27
19	Information objects in the unstructured text document type	29
20	Information objects in the sequential text document type	32
21	Information objects in the unstructured binary document type	35
22	Information objects in the sequential binary document type	37
23	Information objects in the hierarchical document type	40

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

Part 2 : Virtual Filestore 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, and may be stored within open systems or passed between application processes.

ISO 8571 defines a basic file service. It provides basic 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 definition in this part of ISO 8571 is used in the subsequent parts of ISO 8571 which specify services and protocols. They reference the filestore definition in order to assign meaning to the various descriptive data items which they manipulate. This definition will also be used by protocol implementors when choosing a mapping from the protocol items onto their real storage mechanism.

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

- Annex A - File access structure constraint sets;
- Annex B - Document types;

and the following annexes which do not form part of the standard:

- Annex C - Reading of structured files;
- Annex D - Insertion in a structured file;
- Annex E - ASN.1 cross reference.

1 Scope and field of application

This part of ISO 8571

- a) defines an abstract model of the virtual filestore for describing files and filestores (see section one);
- b) defines the set of actions available to manipulate the elements of the model (see section two);
- c) defines the properties of individual files and associations in terms of attributes (see section three);
- d) defines the form of representations of files with hierarchical structures (see clause 7 in section one).

This part of ISO 8571 does not specify

- e) requirements relating to the mapping from real to virtual filestores;
- f) requirements for implementations of the real filestore.

The Virtual Filestore definition is provided for reference by the other parts of ISO 8571 defining the file service (ISO 8571-3) and specifying the file protocol (ISO 8571-4).

2 References

ISO 6429, *Information processing - ISO 7-bit and 8-bit coded character sets - Additional control functions for character imaging devices.*

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

ISO 8571, *Information Processing Systems - Open systems Interconnection - File transfer, access and management.*

- Part 1: General introduction.
- Part 3: File Service definition.
- Part 4: File Protocol specification.

ISO 8601, *Data elements and interchange formats - Information interchange - Representation of dates and times.*

ISO 8650, *Information Processing Systems - Open Systems Interconnection - Protocol specification for the Association Control Service Element.*

ISO 8571-2 : 1988 (E)

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 8832, *Information Processing Systems - Open Systems Interconnection - Specification of the basic class protocol for Job Transfer and Manipulation.*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 Definitions

Terms are defined in ISO 8571-1.

4 Abbreviations

Abbreviations are defined in ISO 8571-1.

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

Section one: The filestore model

5 Basic concepts

A virtual filestore is handled by one or more application entities; an initiator may communicate with these by initializing an FTAM regime, in which the file service is provided. Each application entity that is capable of operating as a responder in an FTAM regime supports a single virtual filestore. Thus an application entity title serves to identify a virtual filestore.

NOTE - Mechanisms for allocation of unambiguous application entity titles and for the corresponding directory services to provide addresses for communication with the entity are outside the scope of ISO 8571.

A filestore may contain an arbitrary number (greater than or equal to zero) of files (see figure 1).

The properties of each file are defined by the values of a set of file attributes. These attributes are global; at any one time, a single attribute value is available to all initiators.

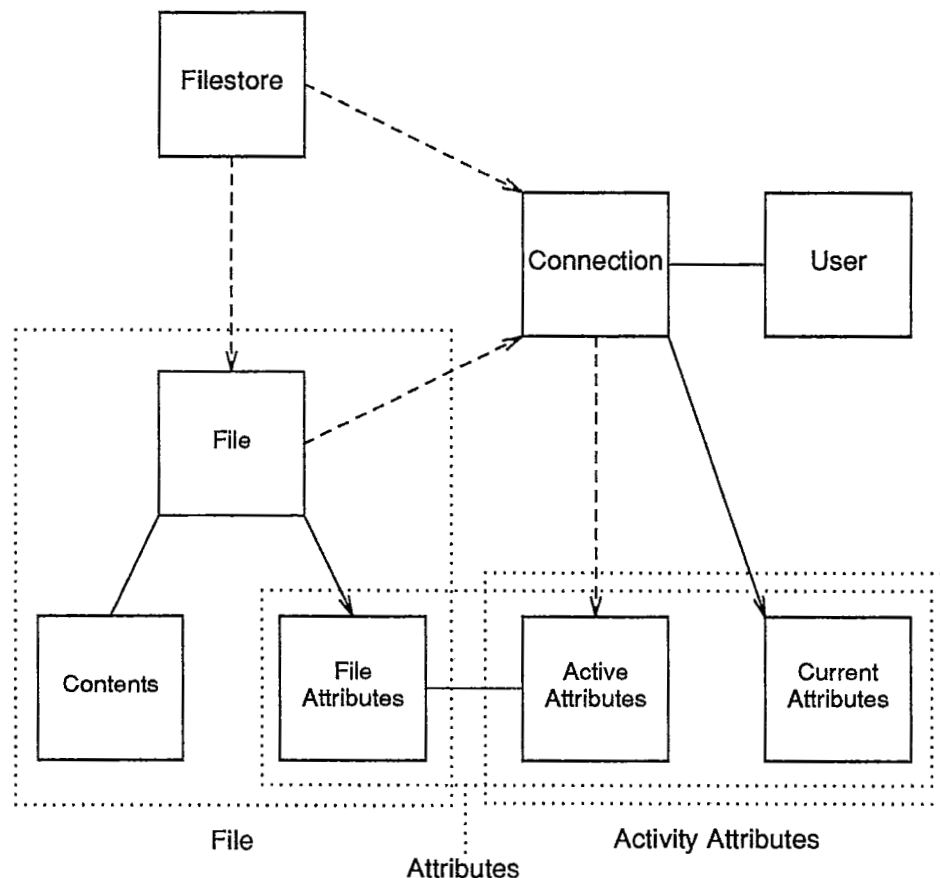
Each file is either empty or has some contents and a structure. Some of the file attributes identify structural aspects of the contents.

There is a set of activity attributes associated with each FTAM regime. The activity attributes are of two types.

The first are in one to one correspondence with the file attributes, and indicate the active value of those attributes as perceived by the initiator.

The second type of activity attributes are current values of state information concerning the exchanges taking place within an FTAM regime, and the facts established about the initiator. These are usually derived from corresponding parameters in protocol exchanges. An initiator has available the description of only the FTAM regimes in which it is participating.

An arbitrary number (greater than or equal to zero) of initiators may have initialized FTAM regimes at any one



Key:

—— 1:1 association

——> 1:n association ($n > 0$)

- - - -> 1:n association ($n \geq 0$)

Figure 1 — Relationship of files, attributes and associations



Documents

Document Sharing Hub

ISO 8571-2 : 1988 (E)

time. Exchanges between the initiator and the responder lead to the selection of at most one file in the responder's virtual filestore to be bound to a particular FTAM regime at any one time.

6 File selection

From outside the filestore, selection of a file is always made by reference to the name of a file. The reference to a file is within the context of a particular filestore identified by the application entity title. The application entity title refers to the location of file storage, and is known to the file service users, but lies outside the scope of FTAM. The filename is defined in clause 12.

Selection of a file takes place in two stages. First, an FTAM regime is initialized with the application entity handling the virtual filestore, and then information is given to this entity, to identify the file unambiguously from amongst all the files in the filestore.

In general, selection could be made by the statement of a number of relations between given values and file attributes. However, in ISO 8571 reference to the file is always made in terms of the filename.

7 File structures

7.1 File access structure

This clause defines the properties of a hierarchical structure. The abstract structure of hierarchical files, including the simple cases of flat and unstructured files, is defined in this part using ASN.1 in the ASN.1 module ISO8571-FADU (see 7.2). A hierarchical structure has the following properties:

- The file access structure is an ordered tree.
- Zero or one Data Units are assigned to a node.

c) Each node within the structure gives access to its subtree. The access unit (that is, the subtree) is known as a File Access Data Unit (FADU) and it is comprised of the structural nodes of the subtree and the Data Units contained within the subtree. The root node of the tree gives access to the entire file.

d) Optionally a node has a name assigned to it.

e) The number of levels, the arc lengths and the number of arcs originating from each node are not restricted.

NOTES

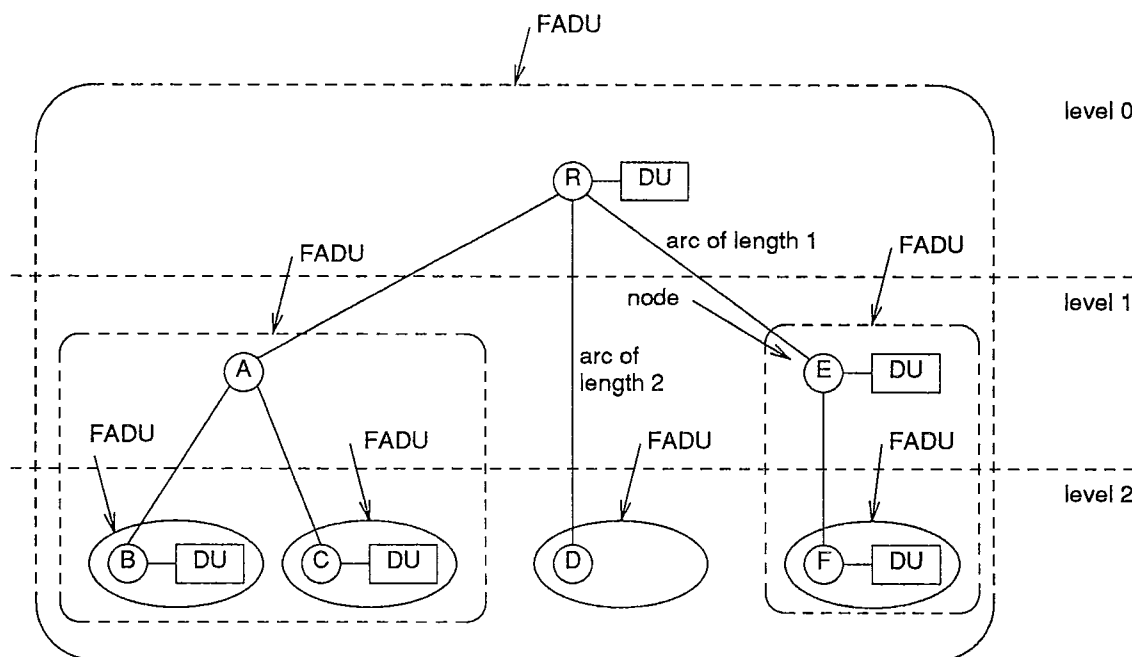
1 Applicable constraint sets may limit the maximum level, and the arc lengths allowed.

2 These terms are illustrated in the figure 2, where, for tutorial purposes, each node has been uniquely identified by the letters A to F.

A preorder traversal sequence is a specific sequence of nodes in the tree; it is established by traversing the subtree corresponding to the whole file. A subtree is traversed by appending the root node of the subtree to the end of the traversal sequence established so far, and then, for each of the children of the subtree in their order of appearance, traversing that subtree.

NOTE - The algorithm describes the abstract construction of a sequence, which is used for establishing FADU identity, defining the location operations and determining the order of transmission. This part of ISO 8571 does not define how the algorithm is to be implemented.

The File Access Data Units (FADUs), being in one-to-one correspondence with the subtrees, can be identified in the same manner as subtrees (that is, by their root nodes). Likewise, data units are always associated with a node and may be identified by the identification of the node. The general hierarchical structure can represent a wide range of different practical file structures. However, real systems can



NOTE - the following features of the tree are identified (subtrees are identified by their root node):

Root node of tree : node R

Preorder traversal sequence : nodes R, A, B, C, D, E, F

Order of appearance : left to right

Figure 2 — The Access Structure as a Tree Structure

only support a limited range of structures, and there are restrictions on the way files can be modified. To express this, the concept of a constraint set is introduced. A constraint set defines limitations on the range of structures allowed, and expresses the way in which the basic access actions can modify the structure. Constraint sets reflecting certain common file types are defined in this part of ISO 8571 (see annex A), but other constraint sets may subsequently be defined and registered.

NOTE - It is expected that a registration authority will be established to maintain a register of constraint sets.

7.2 Abstract structure definition

The access structure of the hierarchical file model is specified in figure 4, using ASN.1. The data unit contents may be expressed in ASN.1, as in figure 3, or in some other abstract syntax notation.

Files are accessed in terms of File Access Data Units (FADU), which are equivalent to the subtrees in the general hierarchical model. The smallest amount of data to be specified for access is one Data Unit (DU). For the purpose of transfer with checkpointing, the Data Units may be divided into smaller parts called data elements (DE). It is not possible to access the individual data elements of a Data Unit by means of the filestore actions defined in ISO 8571; the data unit is accessed as a whole.

7.3 Abstract syntax definition

For the purpose of providing access to the structure of files in the FTAM hierarchical file model this International Standard assigns the ASN.1 object identifier value

```
{ iso standard 8571 abstract-syntax (2) ftam-fadu (2) }
```

as an abstract syntax name for the set of presentation data values, each of which is a value of the ASN.1 type ISO8571-FADU.Structuring-Data-Element.

The corresponding object descriptor value shall be

"FTAM FADU"

The ASN.1 object identifier and object descriptor values

```
{ joint-iso-ccitt asn1 (1) basic-encoding (1) }
```

and

"Basic Encoding of a single ASN.1 type"

(assigned to an information object in ISO 8825) can be used as a transfer syntax name with this abstract syntax.

7.4 File transfer structure

The ASN.1 module ISO8571-FADU defines the access structure of the general hierarchical model. The syntax for transfer of files' contents is derived from this and consists of values of type Data-Element. Only values of type Data-

Element are transferred from the FTAM sender to the FTAM receiver by means of the Presentation Service, ISO 8822. However these values shall appear in their syntactical order.

NOTE - The syntax is constructed so that the syntactical order is equivalent to the preorder traversal sequence.

Structuring information (i.e. values of type Structuring-Data-Element) is communicated in FTAM FADU context (the one corresponding to the FTAM FADU abstract syntax, see 7.3). Node names are either in the same context or in a different, but embedded context. Values of type File-Contents-Data-Element are communicated in the file contents presentation context.

The abstract syntaxes used to transfer part or all of the contents of a file are indicated by the contents type file attribute. Each abstract syntax used shall be associated with a different presentation context. There are two possibilities:

a) The contents type file attribute value specifies an abstract syntax and a constraint set; a presentation context is required corresponding to the specified abstract syntax. This presentation context is used to transfer both Node-Names and the actual contents of the file. A distinct presentation context, which corresponds to the file structuring abstract syntax (see 7.3), is required if the constraint set in use supports file structuring information.

b) The contents type file attribute value specifies a document type; a presentation context is required corresponding to each abstract syntax specified in the document type register entry. The document type also defines which presentation context is to be used for transfer of Node-Names, if present. They can be

1) in a presentation context corresponding to one of the abstract syntaxes defined by the document type (user-coded);

2) in the same presentation context as the file structuring information (ftam-coded).

7.5 Access context

Use of the abstract structure defined in the ISO8571-FADU module to derive the corresponding sequence of information for transfer will yield the full hierarchical structure of the files, i.e. all the structuring information and all the data in the specified FADU will be transferred. However it is possible to access files for reading with a restricted view of their structure by use of different access contexts. In all cases, those data elements which are transferred are in the order defined in ISO8571-FADU, and nodes are transferred in the order in which they appear in the preorder traversal sequence.

1	ISO8571-CONTENTS DEFINITIONS ::=
2	
3	BEGIN
4	
5	File-Contents-Data-Element ::= ANY
6	-- Values of File-Contents-Data-Element are always transferred in a
7	-- presentation context which is different from the presentation context
8	-- used to transfer FTAM PCI. The actual presentation data values allowed
9	-- are found in the abstract syntax for the file contents, as specified
10	-- in the contents type file attribute for the file.
11	
12	END

Figure 3 — ASN.1 definition of file contents

ISO 8571-2 : 1988 (E)

```

1  ISO8571-FADU DEFINITIONS ::=
2
3  BEGIN
4
5      Subtree ::= SEQUENCE {
6          node          Node-Descriptor-Data-Element,
7          data          [0] IMPLICIT DU OPTIONAL,
8          -- present if and only if a DU is connected to the node.
9          children      [1] IMPLICIT Children OPTIONAL }
10         -- a leaf node is characterised by having no children
11
12      Children ::= SEQUENCE {
13          enter-subtree  Enter-Subtree-Data-Element,
14          SEQUENCE OF Subtree,
15          -- subtrees must appear in their proper order according to
16          -- their proper ordering as children of their parent node.
17          exit-subtree   Exit-Subtree-Data-Element }
18
19      DU ::= SEQUENCE OF ISO8571-CONTENTS.File-Contents-Data-Element
20
21      Node-Descriptor-Data-Element ::= [APPLICATION 0] IMPLICIT SEQUENCE {
22          name          Node-Name OPTIONAL,
23          -- present only if the root node of the subtree is a named node.
24          arc-length    [1] IMPLICIT INTEGER DEFAULT 1,
25          -- used to specify the length of the arc to the root node of the subtree
26          -- from its parent node.
27          data-exists    [2] IMPLICIT BOOLEAN DEFAULT TRUE }
28          -- data-exists = TRUE indicates that a DU is connected to the root node
29          -- of the subtree.
30
31      Node-Name ::= CHOICE {
32          ftam-coded     [0] IMPLICIT GraphicString,
33          -- when ftam-coded is used, the Node-Name belongs to the same abstract syntax
34          -- as the structuring information. Node-Names are then transferred in
35          -- the presentation context established to support the FTAM FADU abstract
36          -- syntax. This form of Node-Name is only allowed when the contents type
37          -- file attribute contains a document type name. To support this
38          -- alternative, at least the G0 character set registered in character
39          -- set register entry 2 shall be supported.
40          user-coded     EXTERNAL }
41          -- the actual types allowed are found in the abstract syntax for the files
42          -- contents, as specified in the contents type file attribute for the file.
43
44      Enter-Subtree-Data-Element ::= [APPLICATION 1] IMPLICIT NULL
45
46      Exit-Subtree-Data-Element ::= [APPLICATION 2] IMPLICIT NULL
47          -- the enter-subtree and exit-subtree data elements are used to bracket
48          -- the list of subtrees, which are children of the preceding node.
49
50      FADU ::= Subtree
51
52      Structuring-Data-Element ::= CHOICE {
53          Node-Descriptor-Data-Element,
54          Enter-Subtree-Data-Element,
55          Exit-Subtree-Data-Element }
56
57      -- Data-Element is defined to be a general data type whose values are
58      --
59      -- a) a value of the ASN.1 type Structuring-Data-Element in the abstract
60      --    syntax "FTAM FADU"; or
61      --
62      -- b) a value of the ASN.1 type ISO8571-CONTENTS.File-Contents-Data-Element
63      --    in the abstract syntax derived from the contents type file attribute.
64  END

```

Figure 4 — ASN.1 definition of file structure

7.5.1 HA - Hierarchical All Data Units Access Context

In the HA access context, all four types of data elements (Node-Descriptor-Data-Element, Enter-Subtree-Data-Element, Exit-Subtree-Data-Element and File-Contents-Data-Element) in the addressed FADU are transferred.

7.5.2 HN - Hierarchical No Data Units Access Context

In the HN access context, all data elements of the types Node-Descriptor-Data-Element, Enter-Subtree-Data-Element and Exit-Subtree-Data-Element from the addressed FADU are transferred.

7.5.3 FA - Flat All Data Units Access Context

In the FA access context, data elements of the types Node-Descriptor-Data-Element and File-Contents-Data-Element from the addressed FADU are transferred. Only those Node-Descriptor-Data-Elements in which data-exists has the value TRUE are transferred.

7.5.4 FL - Flat One Level Data Units Access Context

In the FL access context, data elements of the types Node-Descriptor-Data-Element and File-Contents-Data-Element from those nodes in the addressed FADU which belongs to the specified level are transferred. Only those Node-Descriptor-Data-Elements in which data-exists has the value TRUE are transferred.

7.5.5 FS - Flat Single Data Unit Access Context

In the FS access context, the single Node-Descriptor-Data-Element and all the File-Contents-Data-Elements of the single DU belonging to the root node of the addressed FADU are transferred.

7.5.6 UA - Unstructured All Data Units Access Context

In the UA access context, only the data elements of type File-Contents-Data-Element from the addressed FADU are transferred.

7.5.7 US - Unstructured Single Data Unit Access Context

In the US access context, all the data elements of type File-Contents-Data-Element of the single DU belonging to the root node of the addressed FADU are transferred.

Table 1 — Result of reading in an access context

Access Context	Result
HA	Single Subtree
HN	undefined type - because Node-Descriptor-Data-Elements with data exists = TRUE are transferred without the data elements constituting the DU.
FA	series of Subtree each with one node
FL	series of Subtree each with one node
FS	Single Subtree with one node
UA	Single DU
US	Single DU

7.5.8 Summary of access contexts

Reading a non empty hierarchical subtree in these access contexts will have the results shown in table 1, in terms of the data types defined in ISO8571-FADU.

7.6 Identification structure

A FADU is identified by referencing the root node of the corresponding subtree. A FADU can be identified by any of the following mechanisms:

- "first": the first FADU in the preorder traversal sequence for the file structure for which data exists is identified;
- "last": the last FADU in the preorder traversal sequence for the file structure is identified;
- "previous": the FADU preceding the currently identified FADU in the preorder traversal sequence of the file access structure is identified;
- "current": the current location (see clause 8) in the file remains unchanged;
- "next": the FADU following the currently identified FADU in the preorder traversal sequence of the file access structure is identified;
- "begin": the exact meaning of begin depends on the constraint set in use, but is such that "locate next" will identify the "first" FADU.
- "end": establishes a state of the file where there is no current location, but use of "previous" will identify the "last" FADU in the preorder traversal sequence for the file access structure;
- Node-Name: specifies the identifier of the FADU which is to be identified. The search for the specific Node-Name is restricted to the children of the currently located node;
- Sequence of Node-Names: specifies a path of FADU identifiers from the root node of the file to the node to be located. The first Node-Name is a child of the root node of the file, so that the root node itself is identified by an empty sequence of Node-Names;
- Node number: specifies the node to be selected by its number in the preorder traversal sequence for the file access structure. The root node of the file has node number zero;

7.7 Constraint sets

The general hierarchical structure is constrained and its dynamics further qualified by the definition of a number of constraint sets, each identifying a particular structure in common use. These are specified in annex A. Constraint sets are referenced in protocol exchanges by names, which are of ASN.1 type "object identifier".

The set of actions that are ultimately allowable in the open regime is actually further constrained beyond the actions specified in the constraint set. This further restriction is imposed by the permitted actions file attribute, the processing mode activity attribute, the access control file attribute, the concurrency control activity attribute and the functional units selected for the current FTAM regime.

8 Actions on files

The virtual filestore defines actions which manipulate the data units in a file. The definition of the individual actions (see section two) states the data units to which the actions apply, and the effects on those data units. Some actions also establish filestore state, such as the state "file selected" or FADU location.

The actions are invoked by service primitives. Their semantics are defined in conjunction with those of the file service primitives defined in ISO 8571-3 (see clause 11).

ISO 8571-2 : 1988 (E)

Use of each action is subject to access control by the responder (see 12.16). The applicability of an action is subject to concurrency control governing the parallel activities during and after that action (see note 2 and 3).

NOTES

1 Some of the primitives of the service definition are thus closely related to particular actions, but the two are always logically distinct.

2 When the use of commitment, concurrency and recovery has been agreed (see ISO 9804), concurrency controls should be applied as specified in that standard.

3 The concurrency control permits an accessor to request either action not required, shared access, exclusive access or no access for various actions on a resource. If no access or exclusive access is requested and granted, actions by other associations do not occur. If shared access is requested and granted, exclusive access by other associations does not occur.

4 The responder performs actions (as defined in section two) so that they are serializable. Serializable execution is defined to be execution of the operations of concurrently executing activities in a way that produces the same effect as a serial execution of the same activities. A serial execution is one in which each action is completed before the next action begins.

8.1 Relation to bulk data transfer

The hierarchical model of the contents of a file is defined in ASN.1 module ISO8571-FADU. This model defines three data types that are called file structural items. These are

- a) Subtree : the main entry point data type of the module, representing a complete FADU;
- b) Children : a data type which consists of a sequence of an enter subtree, a sequence of one or more subtrees, and an exit subtree. Use of this data type at the outer level does not represent a single FADU but delimits a series of FADUs requiring special action (see 8.3);
- c) DU : a data type which consists only of a single Data Unit.

The receiver of transferred bulk data is said to recognise a file structural item when it receives the corresponding series of data types Data-Element. That is, the series of data types Data-Element in a value of that file structural item data type.

Certain constraint sets distinguish two variants of some actions, defining different semantics for them. These are called either

- d) a normal action, resulting from transfer of a complete subtree at the outer level;
- e) a qualified action resulting from the transfer of a Children file structural item. The qualified action is applied to each subtree in the sequence within the Children data value.

NOTE - This mechanism is used to distinguish between the "insert as sister" and "insert as child" actions in the general hierarchical constraint set.

8.2 Read bulk data transfer

On performing an FADU read action (see 11.2), the responder shall generate one or more Subtree file structural items, or a single DU file structural item, depending on the access context used.

8.3 Write bulk data transfer

Before performing a write action, the responder shall perform a locate action using the FADU-Identity provided.

While performing an insert, replace or extend action (see 11.3 to 11.5), a responder shall parse the series of data

values received, in order to recognise a file structural item. The file structural items acceptable will depend on the constraint set in use, but in general the responder will recognise

- a) the series of data values which can be recognised as a Subtree; this file structural item represents an FADU.
- b) the series of data values which can be recognised as Children; this represents a series of FADU with which qualified actions are to be performed. The meaning of the qualified action and its effect is specified in each constraint set for which the file structural item is valid.
- c) the series of data values which can be recognised as a DU, and which is not recognisable as part of any other file structural item which consists of a longer series of data values; this file structural item represents a Data Unit. In certain constraint sets, this file structural item is used to extend or replace the data unit of the currently located FADU.

NOTES

1 When the transfer is performed with FTAM, the data values are passed to the responder via the F-DATA indication primitive.

2 The file structural item Children consists of an Enter Subtree, a series of Subtree, and Exit Subtree (see 7.2).

3 The action qualifier mechanism is used, for example, to distinguish between the insert actions in a general hierarchical file; the normal action is insert as sister, and the qualified action is insert as child.

4 A DU file structural item is not self delimiting. It is terminated by the end of the transferred data, or by a following Subtree or Children file structural item.

5 The recognition of a file structural item will in general require a look-ahead of one data value.

For each file structural item received, the responder shall

- d) perform the action requested for the transfer as a whole, on the received Subtree or DU, or on each member of the qualified series in Children.
- e) after performing each action, update the current location as specified in the constraint set.

This process is terminated immediately if an error occurs, but if no failure has occurred when the actions specified for a recognised file structural item have been performed, the responder shall attempt to recognise a further file structural item and repeat the above process. This continues until the data transfer terminates or an error is detected. It is an error to receive a series of data values which cannot be recognised as a complete file structural item. However a single data transfer may contain many separate file structural items.

9 Attributes

9.1 Attribute scope

Two classes of attributes are defined:

- a) file attributes; each file is described by one set of file attribute values. The scope of the file attributes is the virtual filestore, and if a file attribute value is changed by the actions of one initiator, the new value is seen by any other initiators subsequently reading that attribute.
- b) activity attributes; each activity takes place within an FTAM regime and is described by one set of activity attribute values. The scope of the activity attributes is at most the FTAM regime, and a distinct and independent set of activity attribute values is bound to each FTAM regime. There are two distinct subdivisions of the activity attributes.

1) The active attributes are in one to one correspondence with the file attributes.

NOTE - In most cases the mapping is trivial, since many file attributes are fixed at file creation time. However several of the active attributes such as active contents type and active legal qualifications have distinct values which are subsets of the file attribute values.

2) The current attributes concern the initiator and, are in general derived from the parameters on the protocol exchanges.

NOTE - The current attributes are not exactly equivalent to static file attributes, but in some cases are closely related. For example the current access passwords must be members of the access passwords term in the access control attribute.

9.2 Scalar, vector and set attributes

Three types of attributes are defined:

- a) scalar attributes; each scalar attribute has one value at a given time;
- b) vector attributes; each vector attribute has a value which is a list of zero or more elements, each of which has an independent value. The elements are ordered;
- c) set attributes; each set attribute has a value which is itself an unordered set of zero or more elements, each of which has a distinct value.

The effect of the change attribute action on each of these different types of attribute is defined in 10.3.

9.3 Attribute values

For each attribute, the type of its value (or of the value of its elements for a vector or set attribute) is defined. The value type is one of the following:

- a) a sequence of characters (excluding space) from the character sets defined by GraphicString in ISO 8824 (ASN.1);
- b) a sequence of octets, as defined by OCTET STRING in ISO 8824 (ASN.1);
- c) an integer, as defined by INTEGER in ISO 8824 (ASN.1);
- d) a boolean, as defined by BOOLEAN in ISO 8824 (ASN.1);

e) an application entity title, as defined by AE-title in ASN.1 module ACSE-1 in ISO 8650 (ACSE);

f) a date and time; date and time values are limited to those expressible in GeneralizedTime; as defined in ISO 8601 (see note 1);

g) an item from a named set of values defined by enumeration in this part of ISO 8571;

h) an OBJECT IDENTIFIER, as defined in ISO 8824;

i) an indication that the value cannot be determined (see note 2).

NOTES

1 The resolution and accuracy with which date and time values are maintained are determined by the responder, and are not defined in this part of ISO 8571. Optionally, the truncation of the representation from the right is used to indicate precision (see ISO 8601); truncation from the left is not used.

2 This response would be made, for instance, if the real system environment had not been able to preserve the information.

9.4 Support of file attributes

Three levels of support for a file attribute within a filestore are defined:

- a) the attribute is not supported; an initiator not supporting an attribute will never reference it. An attempt to reference an attribute which a responder does not support results in an error.
- b) the attribute is partially supported; an initiator shall not claim partial support for any attribute. Reference by an initiator to an attribute which is partially supported by a responder will yield the result "no value available"; attempts to change such an attribute will not succeed.
- c) the attribute is supported; an initiator claiming support for an attribute shall be capable of requesting filestore actions specifying attribute values with at least the minimum range, as defined in clause 15.

NOTE - This capability may be subject to conformance tests.

Reference to an attribute supported by a responder will yield meaningful values with at least the minimum range, as defined in clause 15. Responders claiming support of an attribute shall support the associated semantics of the attribute as defined in clauses 12 and 13.

Section two: Actions on the Filestore

10 Actions on complete files

10.1 Create file

The action creates a new file or optionally selects an existing file, and establishes the attributes of the new file. It establishes the file selection regime and selects the newly created file (see 10.2). The initial state of the file is defined by the creation state entry in the constraint set for that file (see annex A).

10.2 Select file

The action creates a relationship between the initiator and a particular file. The action establishes the file selection regime, which is a prerequisite for the actions in clauses 10.3 to 10.8. In order to select a file successfully, the parameters shall specify exactly one file.

10.3 Change attribute

The action changes the existing file attributes.

- a) For a scalar attribute, the action replaces the existing value of the attribute.
- b) For a vector attribute, the action replaces the complete list of elements with a given list.
- c) For a set attribute, the action:
 - 1) adds a given element or elements to the attribute; and/or
 - 2) removes an element or elements equal to a given value or values from the attribute.

NOTES

- 1 Successful additions of an element to a set attribute requires that element be distinct from all elements in that set.
- 2 Successful removal of an element from a set attribute requires the existence of that element in that set.

10.4 Read attribute

The action interrogates the values of the requested attributes. For a vector or set attribute, it returns the complete list of element values.

10.5 Open file

The action establishes the file open regime for the performance of the actions for file access (see clause 11), on the selected file. A file may be opened for read, when only the read and locate actions are valid, or opened for write, when all file access actions are valid.

10.6 Close file

The action terminates, in an orderly manner, the open regime previously established by the Open File action.

10.7 Delete file

The action deletes and deselects the selected file. It terminates the current file selection regime.

10.8 Deselect file

The action terminates, in an orderly manner, the current file selection regime.

11 Actions for file access

The access actions operate in the regime established by the Open File action. The actions available on a particular file will depend on, and may be modified by the constraint set which applies to the file (see annex A). In addition the permitted actions and access control file attributes may further constrain whether or not certain of the actions may be performed on this file.

The range of FADU identity styles that may be used may be modified by the constraint set. The structuring of the file contents for access purposes is the file access structure defined in clause 7, as modified when performing the read action by the access context in use.

11.1 Locate

The action locates the specified FADU. Parameter values are available to request location at the beginning or at the end of the file. When a file is opened, the location is as specified in the applied constraint set.

Parameter values are available to allow for different styles of identifying the FADU. The range of valid FADU-identity values is defined in 7.6.

If the locate operation fails, the current location remains unchanged.

11.2 Read

The action locates and reads an FADU. The location after the access is unchanged.

The data units and structuring information to be transferred are determined on the basis of the access context requested. The request may be for any of the defined access contexts specified in the constraint set.

11.3 Insert

The action creates a new FADU and inserts it into a position in the file specified in the constraint set (see annexes A and D). In hierarchical files the FADU may be inserted at any level relative to the node becoming the parent; the level is specified in the Node-Descriptor-Data-Element of the FADU (see 7.2). The default relative level is one.

11.4 Replace

This action replaces the contents in an existing DU or FADU. The previous contents are lost. Either the entire FADU currently located is replaced or only the content of the DU connected to its root node is replaced.

- a) Replacing the entire FADU is valid only if file structuring information is available with the replacement FADU. Neither the node name nor the arc length of the root node of the currently located FADU can be changed by means of the replace action. Replacement applies to both structuring information and DUs within the FADU, i.e. new structuring information may be introduced as part of the replacement.

- b) Replacing the DU connected to the root node of the currently located FADU is only allowed if no file structuring information is made available with the replacement DU.