ISO 8571 PT*2 88 ■ 4851903 0545700 414 ■

INTERNATIONAL STANDARD

ISO 8571-2

First edition
1988-10-01

AMENDMENT 2
1993-08-15

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

Part 2 :
Virtual Filestore Definition

AMENDMENT 2 : Overlapped access

Systèmes de traitement de l'information – Interconnexion de systèmes ouverts – Transfert, accès et gestion de fichiers –

Partie 2: Définition du système de fichiers virtuel

AMENDEMENT 2 : Chevauchement d'accès

Reference number
ISO 8571-2:1988/Amd.2:1993 (E)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Amendment 2 to International Standard ISO 8571-2:1988 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology.*

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*
- *Part 5 : Protocol Implementation Conformance Statement Proforma*

© ISO/IEC 1993

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

ISO 8571-2:1988/Amd.2:1993 (E)

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

Part 2 :

Virtual Filestore Definition

AMENDMENT 2 : Overlapped access

0 Introduction

Clause 0 provides an introduction to this amendment. The text in this clause is not intended for inclusion in ISO 8571 part 2.

0.1 General

ISO 8571 part 2 defines an abstract model of the virtual filestore for describing files and filestores. Sets of actions are defined that may be used to manipulate the elements of this model, and the properties of individual files and associations are defined in terms of attributes.

This amendment extends these definitions to incorporate the concepts of overlapped access.

0.2 Rationale

The objective in introducing overlapped access is to allow more efficient access to structured files when a single initiator has a need to perform many reading and updating operations; the serial nature of the current FTAM data transfer services introduces a significant control overhead if the FADUs are small. In this context, an FADU is small if its transmission time is comparable with the time to complete a confirmed service on the association (the association's round trip delay).

0.3 Summary

The current design envelope that there should be at most one file selection per association and one file open per file selection is maintained. If access to more than one file is to be overlapped, more than one association is necessary, The overlapped access takes place within a constant set of presentation contexts established as at present when the file is opened, or previously.

Two different degrees of overlap have been identified. Firstly, requests for future accesses may be issued whilst a previously requested BDT action is in progress, allowing the creation of a queue of read and write requests. In general, PCI relating to a given BDT action may be overlapped with other BDT actions, subject to restrictions; this is called consecutive overlap. Secondly, read and write actions can be performed in parallel, so that both directions of data transfer are exploited at any one time. Requests are then taken from the queue whenever either direction of transfer becomes free. This is called concurrent access.

The transfer of a single FADU, specified in a single F-READ request has the same interpretation as in ISO 8571. The resultant effect on the virtual filestore of a set of overlapped requests using consecutive access shall be the same as that of the equivalent set of requests issued in series; the service provided is serializable. If concurrent access is used then the resultant effect of a set of write actions on the virtual filestore, is also serializable. However, due to the non-determinism introduced by the use of concurrent access, it is also possible that in some uses of the service, the data transferred as a result of a read action is not consistent with the current state of the file.

1 Scope and field of application

This amendment makes no additions to clause 1.

2 References

This amendment makes no additions to clause 2.

3 Reference model definitions

This amendment makes no additions to clause 3.

4 Service conventions definitions

This amendment makes no additions to clause 4.

5 FTAM Definitions

This amendment makes no additions to clause 5.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Section one: The filestore model

6 File selection

This amendment makes no additions to clause 6.

7 File structures

7.6 Identification structure

Add last paragraph:

If concurrent access is in use then the current locations for read and write actions are independent. The semantics of the FADU identifiers previous, current and next, are dependent on the action to be performed.

8 Actions on files

Add last sentence to note 4:

If concurrent access is in use then the effects of a set of read actions, and of a set of write actions, are independently serializable. However, the data transferred as a result of a read action may not be consistent with the current state of the file in some situations.

9 Attributes

This amendment makes no additions to clause 9.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Section two: Actions on the Filestore

10 Actions on complete files

This amendment makes no additions to clause 10.

11 Actions for file access

11.1 Locate

Insert second paragraph

If concurrent access is in use then separate FADUs may be located for read and write operations.

11.6 Erase

Insert second paragraph

If concurrent access is in use then the operation may be used to erase the FADU currently located for either read or write operations.

NOTE - In concurrent access, if either the read or write current location is not within the erased FADU then that location remains unaffected.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Section three: Attribute definitions

12 File attributes

This amendment makes no additions to clause 12.

13 Activity attributes

13.4 Current location

Amend first sentence of first paragraph:

The current location is a vector attribute that indicates the current position within the file.

Insert as second paragraph:

If concurrent access is not in use then the attribute only contains a single element. If concurrent access is in use then the attribute contains two elements corresponding to the locations that are maintained for the read and write actions.

Amend first sentence of third paragraph:

The values of the elements may be either, ...

Insert clause and re-number:

13.6 Current degree of overlap

The current degree of overlap attribute is a scalar attribute. It indicates the degree of overlapped access that is in use in the open regime. The value of the attribute is an integer.

The scope of the current degree of overlap attribute is the open regime.

Table 4 - Activity attributes

Attribute	Type	Minimum Attribute Range
Current access request	boolean vector	see note
Current initiator identity	GraphicString	1 to 8 characters
Current location	vector of Node-Descriptor-Data-Element	see note
Current processing mode	boolean vector	see note
Current degree of overlap	integer	see note
Current calling application entity title	application entity title	No minimum required
Current responding application entity title	application entity title	No minimum required
Current account	GraphicString	1 to 8 characters
Current concurrency control	vector of enumeration	see note
Current locking style	boolean	No minimum required
Current access passwords	GraphicString or OCTET STRING	0 to 8 characters or octets

NOTE - No minimum range can be specified for this activity attribute. The values which this attribute can take depends on the capabilities of the initiator and responder and may be further restricted by the constraint set in use, and/or the permitted actions file attribute, and/or the access control file attribute, and/or the degree of overlap.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14 Attribute groups

14.1 Kernel group

Insert in b) and re-number:

6) current degree of overlap

15 Minimum attribute ranges

Replace table 4.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ISO 8571-2:1988/Amd.2:1993 (E)

UDC 681.3 : 621.39

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

Price based on 5 pages

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

8571 PT*2 88 ■ 4851903 0536271 159 ■

INTERNATIONAL STANDARD

ISO 8571-2

First edition
1988-10-01

AMENDMENT 1
1992-12-15

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

Part 2 :
Virtual Filestore Definition

AMENDMENT 1 : Filestore Management

Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Transfert, accès et gestion de fichiers –

Partie 2: Définition du système de fichiers virtuel

AMENDEMENT 1 : Gestion du système de fichiers

Reference number
ISO 8571-2:1988/Amd.1:1992 (E)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Amendment 1 to International Standard ISO 8571-2:1988 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology.*

ISO 8571-2 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*
- *Part 5 : Protocol Implementation Conformance Statement Proforma*

© ISO/IEC 1992

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

8571 PT*2 88 ■ 4851903 0536273 T21 ■

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

Part 2 :
Virtual Filestore Definition

AMENDMENT 1 : Filestore Management

NOTE – This amendment has additional subclauses and tables to ISO 8571 which are indicated by the use of lower case Roman letters beginning with "a" and imply ordering alphabetically, following the clause with the same numerical value in ISO 8571. These and all subsequent subclauses, tables, and cross references will be renumbered in subsequent editions.

Introduction

(amend 3rd paragraph, page 1)

ISO 8571 defines services for file transfer, access and management. It also specifies a 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, stored and managed within open systems, or passed between application processes.

(amend 4th paragraph, page 1)

ISO 8571 defines a basic file service. It provides sufficient facilities to support file transfer, file access, and management of files stored on open systems. ISO 8571 does not specify the interfaces to a file transfer, access or management facility within the local system.

1 Scope and field of application

(amend 1st paragraph)

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 objects 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).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Section one: The filestore model

5 Basic concepts

(amend 3rd paragraph (after note), page 2)

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

(amend 4th paragraph, page 2)

The properties of each object are defined by the values of a set of object attributes. These attributes are global; at any one time, a single attribute value is available to all initiators. Different object types may have distinct types of attributes, as well as types of attributes in common.

(add following paragraph 5, page 2)

Each file-directory maintains a parenthood relationship with zero or more subordinate objects. Some of the file-directory attributes may identify access control information to subordinate objects.

Each reference maintains a link to exactly one other object. The referent is either a file or a file-directory. The identity of the referent is available as an attribute of the reference, in the form of a (possibly incomplete) primary pathname. This attribute can not be changed. Other reference attributes may identify the object type and access control information to the linked object. If the identity of the referent changes, the corresponding reference ceases to exist.

(amend 7th paragraph, page 3)

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

(amend 9th paragraph, pages 3 and 4)

An arbitrary number (greater than or equal to zero) of initiators may have initialized FTAM regimes at any one time. Exchanges between the initiator and the responder lead to the selection of at most one object in the responder's virtual filestore to be bound to a particular FTAM regime at any one time. Note that multiple file objects may be identified for later selection via the generalized selection service. However only one object may be selected at a time. Further, no guarantees are placed on the availability of any file object in this group if it is eventually selected.

(add after clause 5, page 4)

5a The virtual filestore model

5a.1 Filestore Objects

A virtual filestore is comprised of one or more of three kinds of objects:

a) files;

b) file-directories;

c) references.

5a.1.1 Files

File objects contain data, and provide structuring information to access the data within them (see clause 7).

5a.1.2 File-directories

File-directory objects maintain a set of relationships to zero or more other objects within the filestore, whether those objects are files, references, or other file-directories. This relationship is parenthood. A file-directory is said to be the parent of an object if it maintains the relationship of parenthood for that object. Similarly, an object is said to be the child of a specified directory if that directory is the object's parent. In this way, file-directories provide a means of grouping objects within the virtual filestore. These groups can then be used to provide a structural order (the filestore tree) to the data files within the filestore.

An object is 'in' a file-directory if either

a) that file-directory is the parent of the object

b) there is a reference who's parent is the file-directory, linking to the object.

An object is 'under' a file-directory if either

a) the object is in the file-directory

b) the object is in another file-directory that is under the file-directory. (Note this is a recursive definition.)

5a.1.3 References

Reference objects maintain exactly one relationship to exactly one other object within the filestore. That relationship is linkage. The object linked by the reference must be either a file or a file-directory. The structure defined by the parent and linkage relations is called the filestore structure.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5a.2 Filestore structure

Every virtual filestore has a root object. The root is the only object in the filestore that has no parent. This root is either a file or a file-directory. It cannot be a reference. In the case where it is a file, that file will be the only object within that filestore.

The relationship of parenthood results in a hierarchical model of the filestore, where the root node is represented by the filestore root object, intermediate nodes are represented by file-directories maintaining at least one parenthood relationship, and leaf nodes are represented by files, references, and file-directories maintaining no parenthood relationships.

References may be used for convenience of access in special situations, or for special security needs. References provide a simple means of allowing an object to appear in more than one place in the filestore hierarchy without having to duplicate the object, or worry about maintaining consistency between duplicate objects. In normal use a user will not observe any difference in behavior whether an object is accessed via parenthood or reference.

5a.3 Name resolution

An object is identified within the virtual filestore by a pathname. A pathname is comprised of a series of object names. Each object name in the series identifies the next child object in the virtual filestore. The last object name in the series identifies the target object. The root object in a filestore is identified by a pathname comprised of zero object names. The exact algorithm is described in 5a.3.2.

5a.3.1 The current name prefix

When the pathname of an object begins its series of object names at the root of the filestore, it is called a complete pathname. Otherwise, to uniquely identify an object within the virtual filestore, the incomplete pathname must be resolved to a complete pathname. This is done with the current name prefix activity attribute. The current name prefix is assigned to the association by the responder. The current name prefix is a complete pathname of a file-directory object. The actual mechanisms for this assignment are outside the scope of FTAM, but possible uses could be for providing default file-directories to users, protecting filestore users from potential filestore organizational changes, or for enhanced security control.

An incomplete pathname is resolved to a complete pathname by prepending the series of object names within the current name prefix to the incomplete pathname.

Objects within a virtual filestore may be referenced by complete pathname, or by an incomplete pathname. In the latter case, the responder resolves the incomplete pathname to a complete pathname using the current name prefix. The file protocol is designed such that the responder need not reveal the current name prefix to the initiator, should it be desirable to conceal the filestore structure above this file-directory for security or other reasons.

5a.3.2 Resolving a pathname

A complete pathname is resolved to an object by a series of steps using the object names of the pathname to locate the intermediate objects along the path in turn.

Initially, the root node is located.

For each step, while object names of the pathname remain to be resolved:

a) if the object located is a reference, and the filestore user has passthrough access to this reference, then the object which it references is located (if the user does not have passthrough access to this reference, or if the referenced object is not found, an error is reported);

b) if the object located is a file-directory, and the filestore user has passthrough access to this file-directory, then the child object named by the next object name of the pathname is located (if the user does not have passthrough access to this directory, or the next object name does not correspond to any child of this directory, an error is reported);

c) if the object located is a file, then an error is reported.

If the object located when all object names of the pathname have been exhausted is a reference, then the final action taken depends on the operation being performed:

d) if the operation is specific to reference objects, then the operation is performed on the reference object located;

e) if the operation is not specific to reference objects, then the object to which the reference refers is located, and the operation is performed on the referenced object.

5a.4 Object type checking

If the object located when a pathname is resolved is not of the type required for the operation to be performed then an error is reported.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

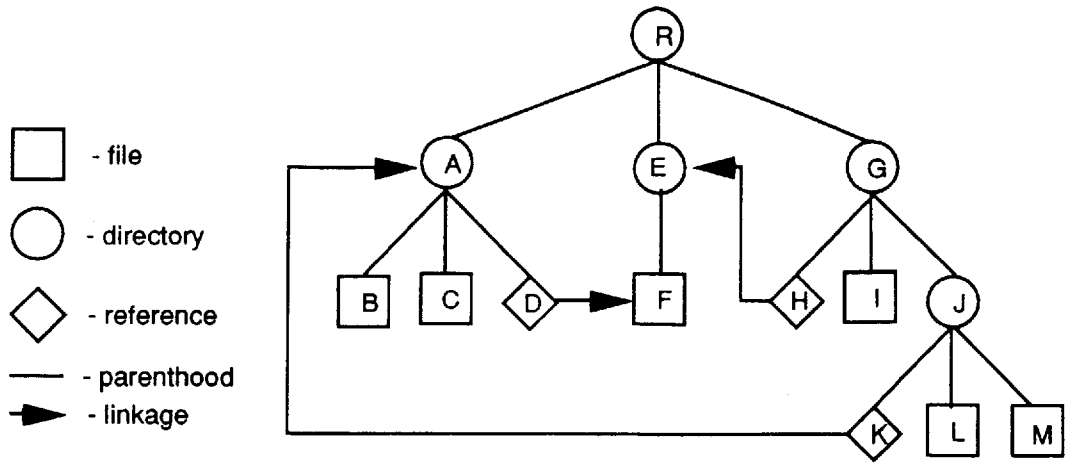


Figure 1a – An example tree structure of a VFS

5a.5 Example

Figure 1a shows an example of a filestore containing references.

The file F has primary pathname E,F. However, it may also be accessed by the following names involving references:

a) A,D

b) G,H,F

c) G,J,K,D

Thus for file selection, the filestore in this example appears as if duplications of data took place as in figure 1b.

NOTES

1) In normal use, except when explicit manipulation of the reference object is carried out, a user will not observe any difference in behaviour whether an object is accessed via parenthood or reference.

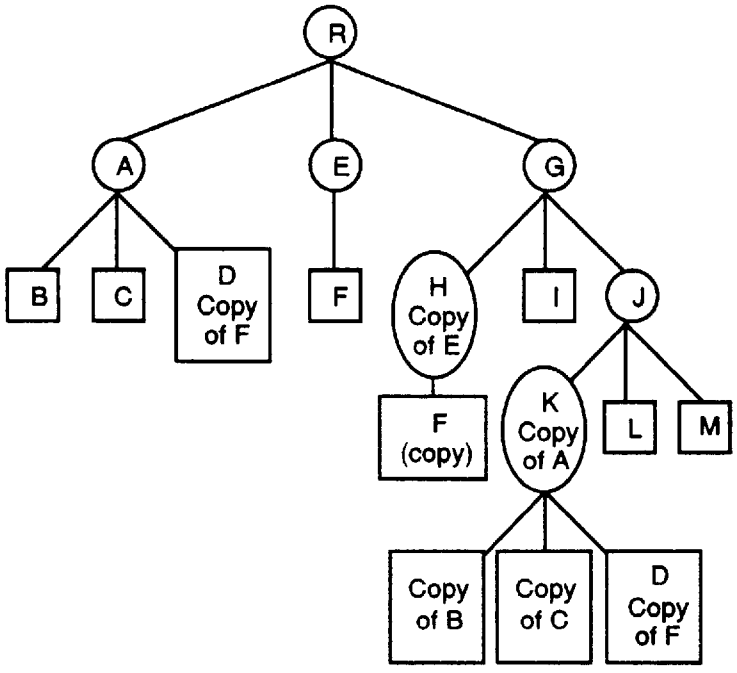


Figure 1b – An example of the apparent structure of a VFS

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) References may be used for convenience of access in special situations. References may have, in conjunction with the path access control attribute, applications to security and secure views of the filestore structure.

(amend title of clause 6, page 4)

6 Object selection

(amend 1st paragraph, page 4)

From outside the filestore, selection of an object is always made using the pathname of the object. Even in the case of generalized selection services, the actual selection of a single object from within the group of file objects maintained in the generalized selection group activity attribute is made by implicit (i.e., internal to the responder controlling the filestore) reference to the pathname of the object. The reference to an object 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 pathname of an object is defined in clause 13.19.

(insert after 1st paragraph, page 4)

6.1 Methods of object selection

Two methods of object selection are provided.

6.1.1 Simple object selection

(amend 2nd paragraph of clause 6, page 4)

Simple object selection 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 object unambiguously from among all the objects in the filestore. This information is the pathname of the object. The current pathname activity attribute is set to the pathname used to identify the selected object.

(replace 3rd paragraph of clause 6, page 4)

6.1.2 Generalized object selection

The second form of object selection works only with file objects and references to file objects. It is the generalized selection mechanism. First, an FTAM regime is initialized with the application entity handling the virtual filestore. Assertions regarding the file objects' attributes are provided by the initiator to the responder. This group of complete pathnames is created and maintained by the responder in the generalized selection group activity attribute. Inclusion in this group is based on the attribute assertions provided by the initiator. Access permission by the initiator to the files based on requested actions and access authorization provided by the initiator is implicitly an assertion in identifying the group of pathnames.

NOTE – The generalized file selection mechanism does not imply formal selection of the objects identified by pathname within the generalized selection group. It merely collects the pathnames for later use in other operations.

Within the FTAM regime, multiple sequential select regimes can be established. These sequential select regimes are created either by selecting another pathname in the generalized selection group, or by the simple object selection mechanism, described above.

Selecting another object in the generalized selection group operates by requesting the responder to choose a previously unselected pathname from the pathname group, and attempting to select it. If it cannot be selected (for example, it has been renamed or deleted since it became a member of the group, or some access control attributes controlling access to the object have been changed to exclude the initiator), then that pathname is removed from the pathname group, a new previously unselected pathname is chosen, and the responder tries again. The current pathname activity attribute is the pathname chosen. No status codes are provided to the initiator to identify this condition. A pathname is considered previously unselected until it is chosen by the responder during a select another action. Selecting an object explicitly by pathname does not affect it's status as previously selected or not within the pathname group.

The initiator deselects an object, which was selected in this manner, by deselecting or deleting the file or its reference. The initiator is notified that no more unselected pathnames exist in the group through a last member indicator. If additional pathnames exist within the group, but upon attempting to select the next one, none are found that can be selected by this initiator, (for example, a concurrency control lock is now in place), then an error is returned.

If, after either of these notifications, another request to select another object is received from the initiator, the responder then considers all remaining pathnames in the group as previously unselected, and begins again. If no pathnames remain in the list, a permanent error is returned. No access guarantees are made regarding the objects listed in the generalized selection group.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Pathnames may be removed from the group by the responder if the responder determines that the object cannot be selected by the initiator. The initiator will not be notified of any deletions from the group. It is possible that, because of references, the same file object will appear in the group multiple times, under different pathnames.

All generalized actions are considered to be specific to a file object. Any references to file objects that may be included in the generalized selection group are treated transparently, so that users are not necessarily aware that they are dealing with a reference.

6.2 Selection of references

After selecting or creating a reference object, actions appropriate to a reference object and actions appropriate to the type of object referenced are allowed (see 5a.3.2). Actions specific to a reference object operate on the selected reference object and its attributes. Actions not specific to reference objects operate on the referenced object and its attributes, with the exception of the object name attribute; in this case, the referenced object "inherits" the object name attribute of the reference object for this specific association for the duration of the select regime. The current pathname activity attribute is set from the pathname used to identify the reference object.

A change attribute action not specific to a reference object results in changes to the referenced object's attributes, with the exception of the object name attribute; in this case, the object name (and possibly the primary pathname) of the reference object is changed.

Actions specific to reference objects are:

a) F-LINK

b) F-UNLINK

c) F-READ-LINK-ATTRIB

d) F-CHANGE-LINK-ATTRIB

Actions involving attribute value assertion lists may operate directly on references, depending on the settings of the object type attribute value assertions, if any.

When invoking an attribute value assertion list, the reference object's attributes and the referenced object's attributes (with the inherited object name attribute) are considered independently. The reference object successfully matches the attribute value assertion list if either of the reference object's or referenced object's attributes match the assertions.

(add after clause 7, page 7)

7a Actions on objects

The virtual filestore defines actions which manipulate the objects within the filestore. The definition of the individual actions (see section two) states the objects to which actions apply, and the effects on those objects. Some actions also establish filestore state, such as the current name prefix, or generalized selection group.

The actions are invoked by service primitives. Their semantics are defined in conjunction with the filestore management primitives defined in ISO 8571-3.

Use of each action is subject to access control by the responder (see 12.16).

9.1 Attribute scope

(amend 1st paragraph, page 8)

Two classes of attributes are defined:

a) object attributes; each object is described by one set of object attribute values. The scope of the object attributes is the virtual filestore, and if an object attribute value is changed by the actions of one initiator, the new value is seen by any other initiators subsequently reading that attribute. Some attributes are specific to the type of object.

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 object attributes.

      NOTE – In most cases the mapping is trivial, since many file attributes are fixed at object 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 object attribute values.

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

NOTE – The current attributes are not exactly equivalent to static object 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.

(add after clause 9.4, page 9)

9.5 Extension attribute sets

The file protocol provides a mechanism for access to object attribute sets which are defined externally to this standard. This is done through extension attribute sets. An extension attribute set consists of an object identifier to identify the attribute set definition, and some number of attributes belonging to the identified attribute set. Each attribute is identified by it's own Object Identifier, and maintains a value specific to that attribute.

If the initiator sends or requests the value of attribute sets not understood by the responder, the responder merely ignores those attribute sets it does not understand, completing the action as though they had not been present.

The responder must never send information about an attribute set not specifically requested by the initiator.

Requesting or recognizing an attribute extension set implies support for all attributes defined within the attribute extension set, and their mechanics.

9a Attribute value assertion lists

The file protocol provides a means for identifying a set of object pathnames based on the attributes of the objects they identify. This mechanism is by attribute value assertion list.

An attribute value assertion consists of an identification of an attribute, a target attribute value, and a relationship. An attribute value assertion is true for a specified object pathname if, for the object identified by the pathname,

a) the identified attribute exists for this object type, and

b) the identified attribute for that object has the specified relationship to the supplied target attribute value.

An attribute value assertion list consists of a set of attribute value assertion sublists, each sublist consisting of a set of attribute value assertions. An attribute value assertion list describes a subset of all pathnames of objects within the virtual filestore.

An object pathname is described by an attribute value assertion list if all of the following are true:

a) the initiator has read-attribute access to the object via that pathname;

b) the initiator has read access to each file-directory specified implicitly in the pathname pattern (see the note in 9a.1.2);

c) the initiator has passthrough access to each file-directory specified implicitly in the pathname pattern;

d) there exists at least one attribute value assertion sublist in the attribute value assertion list for which every attribute value assertion within it has the value 'true' for the object identified by that pathname.

When performing actions on objects using attribute descriptions to identify the set of objects, the initiator provides an attribute value assertion list to describe the desired objects.

The responder then creates a list of pathnames based on the above criteria.. The objects in this list of pathnames may then be operated upon singly, or as a group.

9a.1 Assertion types and components

9a.1.1 Relations for GraphicStrings

Assertions regarding GraphicStrings are made in terms of the logical relation "equality" in comparison to string patterns. A string pattern consists of a sequence of substring patterns. A substring pattern can be any of three types:

a) a specific sequence of characters;

b) a specification for an exact number of characters (those characters which are unimportant);

c) a specification for zero or more characters.

A GraphicString is equal to a string pattern if

1) every character in the GraphicString can be sequentially matched with a character in a pattern of type 'a', a position in a substring pattern of type 'b', or a substring pattern of type 'c' (pattern types correspond to the numbering of the list, above);

2) there are no characters in any string pattern of type 'a', or positions within string pattern type 'b' which do not have a corresponding character from the GraphicString.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 1a – Bitstring and bitstring pattern relationships

Significance	Value of assertion	Meaning
Not significant	any	bit is not significant for matching
Significant	0	bit is significant and matches 0
Significant	1	bit is significant and matches 1

9a.1.2 Relations for Pathnames

Assertions regarding Pathnames are made in terms of the logical relation "equality" in comparison to pathname patterns. Pathname patterns can be either complete pathname patterns, specifying pathname searches are to be made from the filestore root; or incomplete pathnames, specifying pathname searches are to be made from the file-directory identified by the current name prefix.

Either pathname pattern consists of a sequence of component patterns. A component pattern can take any of two forms:

a) a string pattern;

b) a specification for zero or more object names.

A Pathname is equal to a pathname pattern if

1) every object name in the Pathname can be sequentially matched with a string pattern in a component pattern of type 'a', or a component pattern of type 'b' (pattern types correspond to the numbering of the list, above);

2) there are no component patterns of type 'a' which do not have a corresponding object name from the Pathname.

NOTES

1) An object name is said to be "explicit" if the component pattern is of type "a", and that string pattern consists of a single substring pattern of type "a" (see 9a.1.1). Otherwise, the object name is said to be "implicit".

2) In each attribute value assertion sublist, there is always a pathname attribute assertion in effect, even if one is not supplied by the initiator. In that case, a pathname pattern resolving to all objects in the file-directory specified by the current name prefix is implied.

Access control passwords are only used with explicit pathnames.

9a.1.3 Relations for dates and times

Assertions regarding dates and times can be made in terms of

a) "less than" (i.e. before, or older than);

b) "greater than" (i.e. after, or younger than); or

c) "equality" (i.e. concurrent, or same age).

Assertions are evaluated according to both the precision given and the precision available on the local system.

9a.1.4 Relations for integers

Assertions regarding integers can be made in terms of the logical relations

a) "less than",

b) "greater than", or

c) "equality",

Where all are taken with their standard mathematical meanings.

9a.1.5 Relations for bitstrings

Assertions regarding bitstrings can be made in terms of "equality" with a pattern. A bitstring pattern consists of a significance mask and a value assertion. Matches against bitstring patterns are done on significant bits as shown in Table 1a.

A bitstring is equal to a pattern if each significant bit in the bitstring matches the corresponding assertion value in the pattern. Any bits appearing in the bitstring type attribute beyond the length of the significance mask are assumed to be not significant. Any bits set to significant appearing in the significance mask beyond the length of the bitstring type attribute are assumed to not match, making the attribute value not equal to the pattern.

9a.1.6 Relations for object identifiers

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 1b – Attribute relationship combinations

	less than	equality	greater than
less than	–	less than or equal	not equal
equality	less than or equal	–	greater than or equal
greater than	not equal	greater than or equal	–

Assertions regarding object identifiers can be made in terms of "equality". An object identifier pattern consists of an object identifier. An object identifier attribute is equal to an object identifier pattern if they are identical object identifiers.

9a.1.7 Relations for externally defined attributes

Relations for externally defined attributes must be defined within the specification of the external attribute.

9a.1.8 Relations for boolean

Assertions regarding boolean attributes are made in terms of "equality", taken with its standard mathematical meanings.

9a.1.9 Relations for enumerated values

Assertions regarding enumerated attributes are made in terms of "equality", taken with its standard mathematical meanings.

9a.1.10 Relations for octetstring values

Assertions regarding octetstring attributes are made in terms of "equality". An octetstring is equal to an octetstring provided in an attribute value assertion if the two octetstrings are the same length, and each octet within one octetstring is equal in numerical value to the corresponding octet of the other octetstring.

9a.2 Attribute value assertion structure

An attribute value assertion consists of an identification of an attribute, a target attribute value, and a relationship.

9a.2.1 Attribute identification

The way an attribute value assertion identifies the attribute against which it is to be compared depends on the specific attribute. Attributes can be either internal to the files protocol, or else outside the files protocol, using attribute extensions.

An attribute value assertion identifies itself as pertaining to an attribute specified within the files protocol implicitly by position within a list.

An attribute value assertion identifies itself as pertaining to an attribute within an attribute extension set by identifying the attribute by its object identifier. Mapping to specific attributes within an extension set will be defined by the extension set definition, and are outside the scope of this part of ISO 8571.

9a.2.2 Attribute value

The value of an attribute value assertion is either a pattern describing one or more possible values of the attribute (see 9a.1), or the indication "no value available".

9a.2.3 Attribute relationship

The attribute value assertion relationships defined in 9a.1 are defined in some subset of the terms "equality", "greater than", and "less than".

Where only the "equality" relationship is provided, the files protocol provides means for the negation of the attribute value assertion, resulting in the ability to identify an object pathname based on it's "inequality" to a specified attribute pattern.

Where the "greater than" and "less than" relationships are also provided, combinations of the relationships may be expressed to form new relationships by taking the logical "or" result of the truth value of each of the relationships individually. Table 1b shows the allowed combinations of the relationships, and the resulting relationships.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Section two: Actions on the Filestore

(add before clause 10, page 10)

9b Actions on the virtual filestore

9b.1 Change current name prefix

The action modifies the current name prefix activity attribute maintained by the responder. The new value must specify an existing file-directory, by any one of the following methods:

a) a complete pathname, in which case the current name prefix activity attribute is changed to the specified file-directory pathname;

b) an incomplete pathname, in which case it is resolved to a complete pathname using the existing current name prefix activity attribute, then applied as in (a);

c) specifying that the responder should set the current name prefix back to the default established at the time the FTAM Regime was established.

The responder may or may not return the newly established current name prefix, at the responder's option. If the new current name prefix is returned, it may be either a complete pathname, or an incomplete pathname relative to the previous current name prefix, again at the responder's option.

9b.2 List file-directory

The action interrogates the values of requested attributes of all objects in the filestore conforming to a specified attribute value assertion list.

Due to the actions of references, it is possible that the filestore structure could contain loops. For example, if a reference referred to its parent file-directory, it would appear as though it was under itself.

To avoid getting stuck in a loop, a responder reporting the contents of multiple file-directories will never report the contents of any given file-directory more than once. This exclusion applies only to actual file-directory objects encountered multiple times while traversing the filestore structure. It does not prevent a given object from appearing within the list of objects more than once due to one or more references to the object satisfying the attribute value assertion.

9b.3 Generalized selection

The action causes the responder to prepare a set of complete pathnames for the generalized selection group. Pathnames in the group comprise all files and/or references to files in the filestore conforming to a specified attribute value assertion list.

Due to the actions of references, it is possible that the filestore structure could contain loops. For example, if a reference referred to its parent file-directory, it would appear as though it was under itself.

To avoid getting stuck in a loop, a responder setting the generalized selection group activity attribute will never include the contents of any given file-directory more than once. This exclusion applies only to actual file-directory objects encountered multiple times while traversing the filestore structure. It does not prevent a given object from appearing within the generalized selection group more than once due to one or more references to the object satisfying the attribute value assertion.

9b.4 List generalized selection

This action causes the object pathnames currently maintained within the responder's generalized selection group to be enumerated, along with a requested set of attributes.

9b.5 Generalized deletion

This action deletes the files selected in the generalized selection group and thus clears the responder's generalized selection group activity attribute.

9b.6 Move object

The action moves an object to a new location in the filestore under the parenthood relation. The object's primary pathname attribute changes to reflect the move; no other attributes change unless so specified.

9b.7 Generalized move

The action moves the objects identified within the generalized select group activity attribute to a new location in the filestore tree under the parenthood relation. The objects' primary pathname attributes change to reflect the move; no other attributes change unless so specified. When references to files exist in the generalized selection group, only the references are moved within the filestore tree.

9b.8 Copy file

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The action creates a duplicate of an existing object into a specified location in the filestore structure.

9b.9 Generalized copy

The action creates duplicates of the group of file objects identified by pathname in the responder's generalized selection group into a specified location in the filestore tree. When references to files exist in the generalized selection group, the file object is duplicated into the new location, not the reference.

9b.10 Generalized change-attribute

The action changes attributes of the file objects identified by pathnames in the generalized selection group.

a) For a scalar attribute, the action replaces existing value of the attribute.

b) For a vector attribute, the action replaces the complete list of elements with a supplied list.

c) For a set attribute, the action:

1) adds a given element of elements to the attribute: and/or

2) removes an element or elements equal to a supplied value or values from the attributes.

NOTES

1) Successful addition of an element to a set attribute requires the element be distinct from all elements in that set.

2) Successful removal of an element from a set attribute requires the existence of the element in the set.

9c Actions on file-directories

9c.1 Create file-directory

The action creates a new file-directory and establishes the attributes of the new directory. At the time of creation, there are no objects under a new file-directory. This action creates a select regime, forming a relationship between the initiator and the newly created directory.

9c.2 Delete file-directory

The action deletes a specified empty file-directory. A file-directory cannot be deleted if it still maintains parenthood relationships to any objects.

9c.3 Read file-directory attribute

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

9c.4 Change file-directory attribute

This action changes the existing file-directory 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 specified list.

c) for a set attribute, the action:

1) adds a specified element or elements to the attribute; and/or

2) removes an element or elements equal to a specified 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.

9d Actions on references

9d.1 Create reference

The action creates a new reference and establishes attributes of the new reference, including the linked object attribute. A select relationship between the initiator and the linked object is created, forming a select regime.

9d.2 Delete reference

The action removes a specified reference object from the filestore. The referenced object is not deleted. This action terminates the current object selection regime.

9d.3 Read reference attribute

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

9d.4 Change reference attribute

This action changes the existing reference attributes.

a) For a scalar attribute, the action replaces the existing value of the attribute.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) For a vector attribute, the action replaces the complete list of elements with a specified list

c) for a set attribute, the action:

1) adds a specified element or elements to the attribute; and/or

2) removes an element or elements equal to a specified 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.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Section three: Attribute definitions

(insert prior to clause 12, page 12)

11a Generic object attributes

Files, file-directories, and references all have the generic object attributes defined in this clause. All object attributes are global, in that each one has one value, or one set of values, at any particular time. All initiators of file actions will see the same value, set of values or obtain an indication of "no value available" for a file attribute

11a.1 Object name

Each object in a filestore has an object name, with the exception of the root object. The object name is a scalar attribute, of type Graphicstring (see clause 15). The value of the object name attribute is set at object creation, but can be altered by a change object attribute action specifying a new pathname activity attribute. The name attribute of an object identifies the object unambiguously with respect to it's parent directory.

A pathname identifying an object is made up of a sequence of object names (see Section 1, 5a.3.2).

11a.2 Object type

The object type attribute gives the type of object. It is an enumerated scalar attribute. The attribute value is set at object creation and cannot be altered by any change attribute action. The possible values are:

a) File

b) File-directory

c) Reference

11a.3 Primary pathname

The primary pathname attribute is a vector attribute of type GraphicString. It identifies the single complete pathname of an object where each adjacent pair of objects whose names appear in the pathname are related solely by parenthood. The primary pathname attribute may be reported in terms of the current name prefix activity attribute.

NOTE – The filestore root has a primary pathname consisting of a vector with zero elements.

11a.4 Storage account

The storage account attribute identifies the account authority responsible for the accumulated charges for maintenance of the object. The attribute is a scalar attribute, of type GraphicString. The attribute can be altered by the appropriate change attribute action.

11a.5 Date and time of creation

The date and time of creation object attribute gives the date and time that the object was created. The attribute refers to the local date and time of the responder. The attribute is a scalar attribute. The value of the attribute is of type GeneralizedType (see ISO 8571-2 subclause 9.3). The attribute is set by the responder when the object is created and may not be altered by any change attribute action.

11a.6 Identity of creator

The identity of creator attribute gives the identity of the initiating user that created the object. The attribute is a scalar attribute. The attribute value is of type GraphicString. The attribute is set by the responder when the object is created to the value of the identity of the initiator activity attribute when the object is created. The attribute may not be altered by any change attribute actions.

11a.7 Legal qualifications

The legal qualifications attribute conveys information about the legal status of the object and its use. The attribute is a scalar attribute. The attribute value is of type GraphicString. The attribute value is set at object creation and can be altered by the appropriate change attribute action.

11a.8 Date and time of last attribute modification

The date and time of last attribute modification attribute indicates when the attributes of the object were last modified. The attribute is modified by the responder whenever a change attribute action is successfully performed on one or more attributes. The attribute is a scalar attribute. The value of the attribute is of type GeneralizedTime.

11a.9 Identity of last attribute modifier

The identity of last attribute modifier attribute gives the identity of the last initiating user that changed attributes of the object by a change attribute action. The identity of last attribute modifier is a scalar attribute. The attribute is altered by the responder

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

whenever the attributes of the object are changed by a change attribute action. The value is set to the value of the identity of initiator activity attribute. The value is of type GraphicString. For a newly created object, the value is equal to the value of the identity of creator attribute.

11a.10 Access control

The access control attribute gives conditions under which access to the object is allowed. The access control attribute is a set attribute.

Each element of the set gives one condition under which access to the object is allowed. Access to the object is allowed if at least one of these conditions is satisfied. However, the access must be based on a single condition, not on the union of a number of separate conditions. Some elements do not apply to all object types (see table 2a).

The attribute value is set at object creation, but may be altered by the appropriate change attribute action.

A condition consists of one or two terms giving a statement of the access it allows (see (a) below), together with a set of from zero to three other terms giving tests based on the values of various activity attributes (see (b) below). The whole condition is satisfied only if all the terms in it are satisfied.

In detail these are:

a) the statement of access, consisting of

   1) an action list term, in the form of a boolean vector; an action is present in the list if the corresponding boolean is true; the booleans correspond to the actions "read", "insert", "replace", "extend", "erase", "read attribute", "change attribute", "delete", "passthrough", and "link".

      The action list term is used whenever an attempt is made to set the current access request attribute. The action list term is satisfied only if all the actions in the proposed value for the current access request attribute are present in the action list.

      If the action list term is satisfied, the "select" and "deselect" actions are also possible. If the satisfied action list term contains file access actions (see clause 11), the actions "open" and "close" are also possible. The action "locate" is then also allowed, if it is available in the FTAM regime established.

   2) optionally, a concurrency access term, in the form of a vector of concurrency keys; the concurrency keys correspond to the actions "read", "insert", "replace", "extend", "erase", "read attribute", "change attribute", and "delete". A concurrency key is a boolean vector whose elements correspond to the possible values of concurrency locks (see 13.9); the elements in the concurrency key correspond to the lock values "not required", shared", "exclusive" and "no access".

      The concurrency access term is used whenever an attempt is made to set the current concurrency control attribute. The concurrency access term is satisfied only if the boolean corresponding to the lock requested in the proposed value for the current concurrency control attribute is true on the concurrency key.

      If this term is omitted from the access control element, the concurrency control parameter shall not be present in access allowed by this access control element.

b) and the tests are terms consisting of:

   1) optionally, an identity; the value is a GraphicString. The term is satisfied if it matches the value of the current initiator identity attribute for the association (see 13.3).

   2) optionally, passwords in the form of a vector of strings (one or more of which may be empty); each element of the vector corresponds to one of the actions "read", "insert", "replace", "extend", "erase", "read attribute", "change attribute", "delete", "passthrough", and "link". The term is satisfied if each non-null string in the vector matches the corresponding password, held in the current access passwords attribute.

   3) optionally, a location; the value is an application entity title. The term is satisfied if it matches the value of the current calling application entity title attribute for the FTAM regime (see 13.6).

NOTES

1) The initial value of the access control attribute when the object is created is not defined in this standard.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) The attribute can be supported by implementations that allow only one access control condition at a time (a list of length one).

3) Some implementations may impose restrictions on the way terms may be combined in conditions, or on the number of conditions of various forms that exist. For example, an implementation may allow only one password per permitted access type.

4) Matching of identity, password or application entity title is not necessarily on the basis of textual identity. For example, the name of an individual may match the name of a group if the individual is a member of that group.

5) The value of any authentication information in a condition should not be returned in response to any read object attributes action.

The elements for object access actions are (note not every element is appropriate for all object types – see table 2a):

a) "read", allowing the file access action read or the file-directory action list;

b) "insert", allowing the file access action insert or, for file-directories, any action resulting in the creation of an object in the given file-directory;

c) "replace", allowing the file access action replace;

d) "extend", allowing the file access action extend;

e) "erase", allowing the file access action erase, or for file-directories, any action resulting in the deletion of an object in the given file-directory;

f) "read attribute", allowing a read attribute action on an object;

g) "change attribute", allowing a change attribute action on an object;

h) "delete", allowing a delete action on an object.

i) "passthrough", allowing the use of a file-directory or reference object name within a pathname;

j) "link", allowing the creation of references to the given object.

11a.11 Permitted actions

The permitted actions attribute is a vector attribute and indicates the set of actions that could actually be performed on the object, and the set of FADU-identity styles that can be used in actions on file objects (see 7.6 and clause 11). Not all elements of the attribute correspond to all object types – see table 2a. The responder implements this set of permitted actions in any way which is capable of mapping them onto the underlying real system.

The different styles of FADU-identities are categorized into three FADU-identity groups as follows:

Traversal:	begin, first, next, last, end;
Reverse Traversal:	begin, first, previous, last, end;
Random order:	current, single Node-Name, sequence of Node-Names, node number.

The attribute is a vector whose elements take boolean values. Each of the elements indicates the availability of an action or a FADU-identity group. The elements are:

a) actions available:
   1) read
   2) insert
   3) replace
   4) extend
   5) erase
   6) read attribute
   7) change attribute
   8) delete
   9) passthrough
   10) link

b) FADU-identity groups available:
   1) traversal
   2) reverse traversal
   3) random order

The value of the permitted actions attribute is set when the object is created, and cannot be changed by any change attribute action.

11a.12 Object size

The object size attribute is a scalar attribute. It is altered by the responder whenever the object's storage space requirements change. The attribute value is set to the nominal size in octets of the object when it is not selected (see note 2).

The attribute cannot be altered by use of any change attribute action, or set using the initial attributes parameter on a create action.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 2a – access control and permitted actions by object type

	file	file directory	reference
read	X	X	
insert	X	X	
replace	X		
extend	X		
erase	X	X	
read attributes	X	X	X
change attributes	X	X	X
delete	X	X	X
passthrough		X	X
link	X	X	
concurrency	X		
FADU-ids	X		

Legend:

"X" The access control or permitted actions element applies to the object.

" " The access control or permitted actions element does not apply to the object, and is ignored on operations involving this kind of object.

The value of the attribute is an integer.

NOTES

1) The object size is based on the real filestore's particular representation of the information involved. The size of an object may vary when it is moved or copied.

2) If a real filestore allocates space for objects in multiples of a basic unit (e.g., a block or bucket), it may be unable to determine the accurate number of octets utilized by an object. The object size may then assume only multiples of the basic unit (in octets).

11a.13 Future object size

The future object size is a scalar attribute. It indicates the nominal size in octets to which an object may grow as a result of modifications or extensions.

The attribute value is set at object creation, but can be altered by an appropriate change attribute action.

The value of the attribute is an integer.

NOTE – When the value of the current object size attribute attains the value of the future object size attribute, the responder may:

a) increase the future object size;

b) increase the future object size and give a warning;

c) not increase the future object size, but indicate an error.

11b File-directory attributes

11b.1 Identity of last reader

The identity of last reader attribute gives the identity of the last initiating user that listed the file-directory. The attribute is altered by the responder whenever the contents of the file-directory are listed, and set to the value of the identity of initiator activity attribute. The identity of last reader attribute is a scalar attribute. The value is of type GraphicString. For a newly created file-directory, the value is equal to the value of the identity of creator attribute. The attribute cannot be altered by use of any change attribute action.

11b.2 Date and time of last read access

The date and time of last read access attribute indicates when the parenthood relationships

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

maintained by the file-directory were last listed. The attribute is altered by the responder whenever the contents of the file-directory are listed, and set to the value of the date and time as known by the responder. The date and time of last read access attribute is a scalar attribute. The value is of type GeneralizedTime. For a newly created file-directory, the value is equal to the value of the date and time of creation attribute. The attribute cannot be altered by use of any change attribute action.

11b.3 Date and time of last modification

The date and time of last modification attribute indicates when the list of parenthood relations maintained by the file-directory was last modified, and refers to the local date and time of the responder. The attribute is a scalar attribute. The attribute is altered by the responder whenever an object is added or removed from the file-directory via any create object or delete object actions. The attribute is not altered when the contents or attributes of an object in the file-directory are modified. The attribute cannot be altered by any change attribute action.

11b.4 Identity of last modifier

The identity of last modifier attribute is a scalar attribute. The identity of last modifier attribute is modified by the responder whenever an object or reference is added to or removed from the list of parenthood relations maintained by the file-directory. The value of the attribute is set to the value of the identity of initiator activity attribute when the file-directory is modified. The attribute cannot be changed by any change attribute action.

11b.6 Child objects

This attribute is a set of the object names of all child objects in the file directory. It is initially empty when the file directory is created. It is altered by the responder whenever child objects are created or deleted.

11c Reference attributes

11c.1 Path access control

The path access control reference attribute gives conditions under which objects may be accessed when the pathname by which they are accessed includes this reference. The attribute is a vector attribute. The form of the attribute is identical to that of the access control attribute. Each element of the set gives one condition under which objects may be accessed via the reference.

The attribute value is set at reference creation, but may be altered by the change reference attribute action.

11c.2 Linked object

The linked object attribute names the object to which a reference links. The attribute is set at the time the reference is created. The value of the attribute always names the complete primary pathname of an existing object. A responder may return the value of this attribute as an incomplete pathname if desired. The value of the current name prefix activity attribute has no effect on the resolution of the referenced object. This attribute is a scalar attribute. The type of the attribute is pathname. This attribute can not be altered by use of any change attribute action.

11c.3 Linked object type

The linked object type attribute gives the type of object referred to by the reference. It is an enumerated scalar attribute. The attribute value is set at reference object creation or when changing the linked object attribute of the reference. It cannot be altered by any change attribute action. The possible values are:

a) File

b) File-directory

12. File Attributes

(delete subclauses 12.1, 12.2, and 12.4, page 12)

(delete subclauses 12.5, 12.8, 12.9, and 12.12, page 13)

(delete subclauses 12.14, 12.15, and 12.16, page 14)

(delete subclause 12.17, page 15)

12.18 Private use

(append after 3rd paragraph, before the note, page 15)

No assertions can be made about values of the private use attribute.

(append after clause 12.18, page 15)

12a Attribute extension sets

Attribute extension sets are sets of attributes defined externally to this standard. An attribute extension set is identified by Object Identifier, and consists of one or more attributes that may be associated with one or more filestore object types.

Each object attribute within an attribute extension set is also identified by an Object Identifier. The attribute describes some aspect of the associated filestore object. These attributes are not limited to scalar

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

values, but may take on other types of values, such as a vector or a set.

The definition of an attribute within an attribute extension set may require that specific actions be performed by an entity managing a virtual filestore during the course of maintaining the attribute. For example, an attribute could require that it be updated upon some other action within the virtual filestore, as is done with the existing date and time of last modification object attribute. Alternatively, an attribute could modify the behavior of the virtual filestore, as is the case of the existing access control object attribute.

Attributes within an attribute extension set may also have patterns defined for use in attribute value assertions. Any attribute without a defined pattern may not be used within an attribute value assertion.

An implementation supporting a given attribute extension set must support the syntax and semantics of all attributes and their patterns defined by the attribute extension set, including any requirements placed on the filestore by the attribute.

13 Activity attributes

13.2 Current access request

(amend 1st paragraph, page 15)

The current access request attribute is a vector attribute whose elements take boolean values. There is one element for each action which may or may not be possible in a particular select regime, and thus be requested during the FTAM regime.

(amend 2nd paragraph, page 15)

If any access actions on a given file object are allowed, the actions "open" and "close" are also possible. The action "locate" is then also allowed, if it is available in the FTAM regime established.

(delete 3rd paragraph, page 15)

(amend 4th paragraph, page 15)

If any action on a filestore object is allowed, the "select" and "deselect" actions are also possible.

13.9 Current concurrency control

(append the following paragraph, page 16)

Concurrency control only applies to file objects.

13.11 Current access passwords

(amend 1st paragraph, page 16)

The current access passwords attribute consists of a structure containing passwords. The passwords correspond to access controls and path access controls of the target object and its current pathname. Each password gives a value associated with one of the elements of the current access request attribute, plus a vector of passwords for the "passthrough" activity, and a password for the "link" activity. These values are set from the parameters of the select or create object service primitives, or from the generalized access passwords activity attribute on a select another service primitive.

13.12 Active legal qualification

(amend 1st paragraph, page 16)

The active legal qualification is a scalar attribute. Its meaning is not defined in ISO 8571, nor is its relationship to the object attribute Legal qualifications. The value is established when the object is selected, and is derived from the legal qualifications attribute.

(amend 3rd paragraph, page 16)

The scope of the active legal qualification attribute is the select regime.

(append following subclause 13.12, page 16)

13.13 Current name prefix

The current name prefix activity attribute gives the complete pathname relative to which incomplete pathnames are resolved. Its scope is the FTAM regime. The attribute is a vector attribute. The value is set

a) at filestore initialization to a default pathname of the home file-directory based on the information agreed for filestore initialization or

b) by changing or resetting the current name prefix via the change current name prefix service primitive.

The value of the attribute is a complete pathname.

13.14 Generalized access passwords

The generalized access passwords activity attribute is a structure identical to that of the current access passwords activity attribute. Its scope is the FTAM regime. Each password gives a value associated with one of the elements of the generalized access request attribute, plus a vector of passwords for the "passthrough" activity, and a password for the "link" activity. The values of this attribute are set from the parameters of the generalized select file service primitive.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

When a file accessed using the select another file service primitive, the values are verified against values in the current access control activity attribute (see 13.15) to determine whether access is to be permitted.

The value of each element of the attribute is of type GraphicString or Octetstring.

13.15 Current access control

The current access control activity attribute gives the access control that applies to the currently selected object. The attribute is a vector attribute. Each element of the vector gives the most restrictive condition from the corresponding element of the access control attributes of

a) the object being accessed;

b) the path access control of all file-directories and references listed in the complete pathname that is derived for the object by the method described in 5a.3.

The scope of the current access control activity attribute is the select regime.

13.16 Generalized selection group

The generalized selection group activity attribute is a group of complete pathnames maintained by the responder, identifying one or more files existing in the filestore. This group of files is determined by the generalized selection file service primitive, as described in 6.1.2. The scope of the attribute is the FTAM regime. The initial value upon FTAM regime establishment is an empty group.

NOTE – The F-GROUP primitive operations will only be effective if a previous F-GROUP-SELECT operation has produced a non-trivial list of pathnames.

13.17 Generalized access request

The generalized access request attribute's structure is identical to the current access request activity attribute. It is taken from the parameter on the generalized selection service, and is used when building the generalized selection group (see clause 6.2).

The current access request attribute takes on this value within a select regime established by the select another action.

13.18 Current object type

The current object type attribute is an enumerated scalar attribute. The possible values are:

a) file

b) file-directory

c) reference

The value of this attribute is set from the object type attribute of the object selected when establishing a select regime. The scope of this attribute is the select regime.

13.19 Current pathname

The current pathname activity attribute is a pathname maintained by the responder, identifying the complete pathname used within the filestore to identify the currently selected object. The scope of the attribute is the selection regime. It is set either explicitly by a select object, create object, or link object services, or implicitly from a pathname in the generalized selection group by the select another service.

13.20 Generalized concurrency control

The generalized concurrency control attribute's structure is identical to the current concurrency control activity attribute. It is taken from the parameter on the generalized selection service, and is used when building the generalized selection group (see 6.2).

The current concurrency control attribute takes on this value within a select regime established by the select another action.

14 Attribute groups

(append after 1st paragraph, before the note, page 16)

Table 3a shows the relationships between object attributes, object types, and attribute groups. An attempt by the initiator to read an attribute from a negotiated attribute group that is not applicable to the object type will be ignored by the responder, resulting in a response containing only the attribute values that were requested by the initiator and applicable to the object.

An attempt by the initiator to change an attribute from a negotiated attribute group that is not applicable to the object type will produce an error from the responder.

14.1 Kernel group

(amend 1st paragraph, page 16)

Kernel attributes are always defined for a corresponding object, and are available for any such object in any specification which references the virtual filestore. For kernel attributes, the value "no value

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

available" cannot be provided.

In addition to the object attributes listed in table 3a, the following activity attributes are in the kernel group, and are supported by every conformant FTAM implementation:

FTAM regime:

a) Current initiator identity

b) Current calling application entity title

c) Current responding application entity title

d) Current name prefix

Selection regime:

e) Current access request

f) Current pathname

g) Current Object type

Open regime:

h) Active contents type

i) Current location

j) Current processing mode

14.2 Storage group

(amend 1st paragraph, page 16)

Attributes in the storage group are only meaningful if for fully supported attributes the responder guarantees the storage of information, and allows one activity to reference information established by some previous activity. Any of these attributes may take the value "no value available".

Specifying the storage group allows reference to the object attributes so marked in table 3a, and the following activity attributes:

FTAM regime:

a) Current account

b) Generalized selection group

c) Generalized access request

d) Generalized concurrency control

Selection regime:

e) Current account

f) Current concurrency control

Open regime:

g) Current concurrency control

h) Current locking style

14.3 Security group

(amend 1st paragraph, page 17)

Attributes in the security group are concerned with security and access control. Object attributes so marked in table 3a and the following activity attributes form the security group. Any of these attributes may take the value "no value available".

FTAM regime:

a) Generalized access passwords

Selection regime:

b) Current access passwords

c) Current access control

d) Active legal qualification

14.4 Private group

(amend 1st paragraph, page 17)

Attributes in the private group are outside the scope of OSI standardization. Object attributes so marked in table 3a are free standing, and may only be specified if the private use group is specified. Attributes in this group may take on the values "no value available" and "abstract syntax not supported".

14.5 Extension group

Attributes in the extension group are only meaningful if the responder supports extension attributes defined externally to this part of ISO 8571. If a responder supports a given attribute set in this group, then it must support the syntax and semantics of all the attributes in the set including any requirements in their patterns for use in attribute value assertions and any requirements placed in the filestore by the attribute.

15 Minimum attribute ranges

(amend 2nd paragraph, page 17)

For activity attributes which correspond directly to object attributes the same minimum range shall apply to the activity attribute as to the corresponding object attribute.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 3a – Object attribute groups

	File	File-directory	Reference
Access control	L	L	L
Availability	S		
Contents type	K		
Date and time of creation	S	S	S
Date and time of last attribute modification	S	S	S
Date and time of last modification	S	S	
Date and time of last read access	S	S	
Future object size	S	S	S
Identify of last modifier	S	S	
Identity of creator	S	S	S
Identity of last attribute modifier	S	S	S
Identity of last reader	S	S	
Legal qualifications	L	L	L
Linked object			K
Linked object type			K
Object name	K[1]	K[1]	K
Object size	S	S	S
Object type	K	K	K
Path access control		L	L
Permitted actions	K	K	K
Primary pathname	S	S	S
Storage account	S	S	S
Private use	P		
Attribute Extension Set	E	E	E
Child Objects		K	

Legend:

K – kernel group
S – storage group
L – security group
P – private group
E – extension group
– Not applicable to this object type

NOTE – Excluding the root of the filestore, which does not have an object name.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 4 – Activity Attributes

(append entries to table 4, page 17)

Active contents type	Object Identifier	no minimum required
Active legal qualification	Graphicstring	no minimum required
Current access control	boolean vector	see note
Current name prefix	vector of Graphicstrings, each element	zero elements 1 to 8 upper case letters or numbers starting with a letter
Current object type	enumeration	any non-empty subset of the defined values
Current pathname	vector of Graphicstring	single vector element 1 to 8 upper case letters or numbers starting with a letter
Generalized access passwords	Graphicstring or OCTET STRING	0 to 8 characters or octets
Generalized access request	boolean vector	see note
Generalized concurrency control	vector of enumeration	see note
Generalized selection group	set of vectors of Graphicstring	single vector element 1 to 8 upper case letters or numbers starting with a letter

(amend note on table 4, page 17)

NOTE – No minimum range can be specified for this activity attribute. The values which this attribute can take depend on the capabilities of the initiator and responder and may be further restricted by the constraint set in use, and/or the permitted actions object attribute, and/or the access control and path access control object attributes.

Table 5 – Object Attributes

(amend entry "filename", page 18)

Attribute	Type	Minimum Attribute Range
Pathname[1]	vector of Graphicstrings, each element	single element 1 to 8 upper case letters or numbers starting with a letter

(amend entry "File availability", page 18)

Availability	enumeration	any non-empty subset of the defined values

(amend entry "Filesize", page 18)

Object size	Integer	no minimum required

(amend entry "Future filesize", page 18)

Future object size	Integer	no minimum required

(amend title to Table 5, page 18)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(append entries to table 5, page 18)

Linked object type	enumeration	any non-empty subset of the defined values
Linked object	vector of Graphicstrings, each element	single element 1 to 8 upper case letters or numbers starting with a letter
Object name	Graphicstring	1 to 8 upper case letters or numbers starting with a letter
Object type	enumeration	any non-empty subset of the defined values
Path access control	set of conditions	single condition
action list	boolean vector	any non-empty subset of the defined values
concurrency access	vector of boolean vectors	any non-empty subset of the defined values
identity	Graphicstring	1 to 8 characters
location	application entity title	all application entity titles
passwords	Graphicstring or OCTET STRING	0 to 8 characters or octets
Primary pathname	vector of Graphicstrings, each element	single element 1 to 8 upper case letters or numbers starting with a letter
Child objects	set of Graphicstrings, each element	single element 1 to 8 upper case letters or numbers starting with a letter

NOTE – Pathname is not actually an object attribute, but an activity attribute. However, changing this activity attribute via a change attribute action can possibly result in changes to the object attribute "promary pathname".

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ISO 8571-2:1988/Amd.1:1992 (E)

UDC 681.3 : 621.39

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

Price based on 23 pages

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8571 PT*2 88 ■ 4851903 0105930 2 ■

INTERNATIONAL STANDARD ISO 8571-2 : 1988
TECHNICAL CORRIGENDUM 1

Published 1991-06-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

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

Part 2:
Virtual Filestore Definition

TECHNICAL CORRIGENDUM 1

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

Partie 2: Fichier virtuel

RECTIFICATIF TECHNIQUE 1

Technical corrigendum 1 to International Standard ISO 8571-2 : 1988 was prepared by ISO/IEC JTC 1, *Information technology.*

Page 14

Subclause 12.16

Delete note 1 and insert the following:

"1 The initial value of the access control attribute is assigned when the file is created. If the access control attribute is not supplied when the file is created, then it is a local responder matter to determine the value assigned."

UDC 681.3.621.39 **Ref. No. ISO 8571-2 : 1988/Cor.1 : 1991 (E)**

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

© ISO/IEC 1991

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Page 21

Annex A, table 8

Delete the column headings and insert the following:

Action	Begin	End	First	Last	Current	Next	Previous	Node number

Page 22

Annex A, table 10

Delete the column headings and insert the following:

Action	Begin	End	First	Last	Current	Next	Previous	NodeSeq	Node number

Page 24

Annex A, table 12

Delete the column headings and insert the following:

Action	Begin	End	Current	Next	Previous	NodeSeq	Node number

Page 32

Annex B, table 20

Replace Datatype2 with the following:

.
.
.

Datatype2: : = Node-Descriptor-Data-Element

Page 34

Annex B.2, subclause 11.1.2

Second paragraph, delete the first line and insert "In access context FA the resultant document"

Page 37

Annex B, table 22

Replace Datatype2 with the following:

.
.
.

Datatype2: : = Node-Descriptor-Data-Element

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Page 38

Annex B.4, subclause 11.1.2

Second paragraph, delete the first line and insert "In access context FA the resultant document"

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8571 PT 2-88 4851903 0063505 6

Q-60-04

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

Reference number
ISO 8571-2 : 1988 (E)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© International Organization for Standardization, 1988 ●

Printed in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

INTERNATIONAL STANDARD ISO 8571-2 : 1988 (E)

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.*

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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

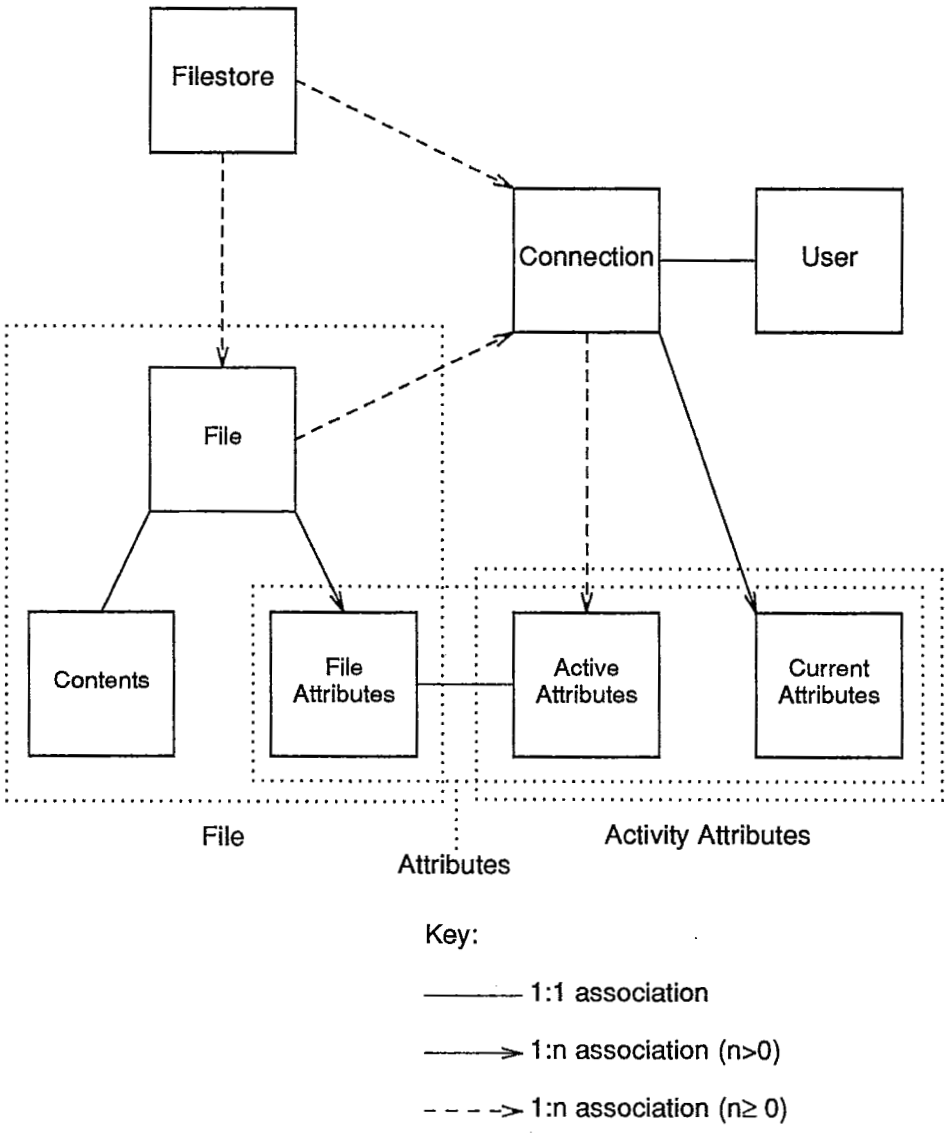


Figure 1 — Relationship of files, attributes and associations

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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:

a) The file access structure is an ordered tree.

b) 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

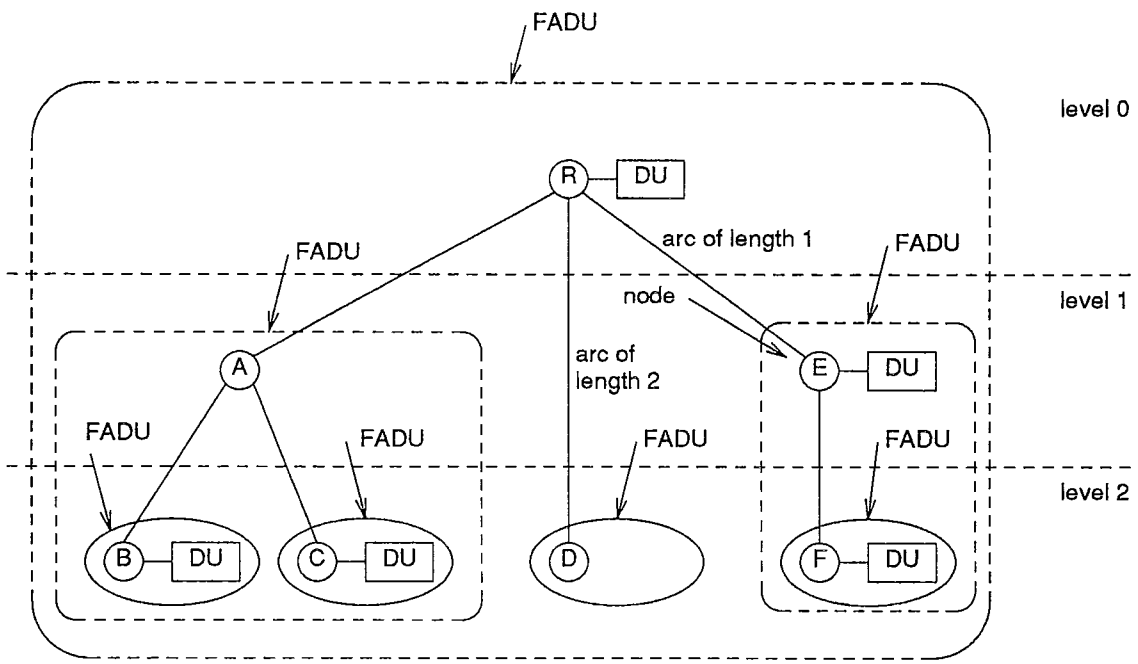
1 Applicable constraint sets may limit the maximum level, and the arc lengths allowed.

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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 valuesallowed
 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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
ISO8571-FADU DEFINITIONS ::=

BEGIN

  Subtree ::= SEQUENCE {
    node          Node-Descriptor-Data-Element,
    data          [0] IMPLICIT DU OPTIONAL,
      -- present if and only if a DU is connected to the node.
    children      [1] IMPLICIT Children OPTIONAL }
      -- a leaf node is characterised by having no children

  Children ::= SEQUENCE {
    enter-subtree    Enter-Subtree-Data-Element,
    SEQUENCE OF Subtree,
      -- subtrees must appear in their proper order according to
      -- their proper ordering as children of their parent node.
    exit-subtree     Exit-Subtree-Data-Element }

  DU ::= SEQUENCE OF ISO8571-CONTENTS.File-Contents-Data-Element

  Node-Descriptor-Data-Element ::= [APPLICATION 0] IMPLICIT SEQUENCE {
    name             Node-Name OPTIONAL,
      -- present only if the root node of the subtree is a named node.
    arc-length       [1] IMPLICIT INTEGER DEFAULT 1,
      -- used to specify the length of the arc to the root node of the subtree
      -- from its parent node.
    data-exists      [2] IMPLICIT BOOLEAN DEFAULT TRUE }
      -- data-exists = TRUE indicates that a DU is connected to the root node
      -- of the subtree.

  Node-Name ::= CHOICE {
    ftam-coded       [0] IMPLICIT GraphicString,
      -- when ftam-coded is used, the Node-Name belongs to the same abstract syntax
      -- as the structuring information. Node-Names are then transferred in
      -- the presentation context established to support the FTAM FADU abstract
      -- syntax. This form of Node-Name is only allowed when the contents type
      -- file attribute contains a document type name. To support this
      -- alternative, at least the G0 character set registered in character
      -- set register entry 2 shall be supported.
    user-coded       EXTERNAL }
      -- the actual types allowed are found in the abstract syntax for the files
      -- contents, as specified in the contents type file attribute for the file.

    Enter-Subtree-Data-Element ::= [APPLICATION 1] IMPLICIT NULL

    Exit-Subtree-Data-Element ::= [APPLICATION 2] IMPLICIT NULL
      -- the enter-subtree and exit-subtree data elements are used to bracket
      -- the list of subtrees, which are children of the preceding node.

    FADU ::= Subtree

    Structuring-Data-Element ::= CHOICE {
      Node-Descriptor-Data-Element,
      Enter-Subtree-Data-Element,
      Exit-Subtree-Data-Element }

    -- Data-Element is defined to be a general data type whose values are
    --
    --   a) a value of the ASN.1 type Structuring-Data-Element in the abstract
    --      syntax "FTAM FADU"; or
    --
    --   b) a value of the ASN.1 type ISO8571-CONTENTS.File-Contents-Data-Element
    --      in the abstract syntax derived from the contents type file attribute.
END
```

Figure 4 — ASN.1 definition of file structure

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.5.1 HA - Hierarchical All Data Units Access Context

In the HA access context, all four types of data elments (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:

a) "first": the first FADU in the preorder traversal sequence for the file structure for which data exists is identified;

b) "last": the last FADU in the preorder traversal sequence for the file structure is identified;

c) "previous": the FADU preceding the currently identified FADU in the preorder traversal sequence of the file access structure is identified;

d) "current": the current location (see clause 8) in the file remains unchanged;

e) "next": the FADU following the currently identified FADU in the preorder traversal sequence of the file access structure is identified;

f) "begin": the exact meaning of begin depends on the constraint set in use, but is such that "locate next" will identify the "first" FADU.

g) "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;

h) 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;

i) 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;

j) 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 note 1).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The current location is not changed by the replace action.

11.5 Extend

The action adds data to the end of the DU associated with the root node of the currently located FADU. The extend action only applies to existing DUs. The current location is not changed by the extend action.

11.6 Erase

The FADU is erased and the first remaining FADU in the preorder traversal sequence after the erased FADU is located. If the erase action is performed while the root node of the file is located, the file is restored to its state following creation.

11.7 File actions and current location

Table 2 defines the effect of the file access actions on the current location. This current location may have been established by an implicit locate if the service primitive which invokes the action carried an FADU identifier. If a locate action is not successful then the current location remains unchanged.

NOTE - Annex D contains examples of the effect of actions on the current location and the way structures can be built up.

Table 2 — Effect of actions on location

Access operation	Location after access
Locate	As specified by FADU Identity, defined in 11.1.
Read	As specified by FADU Identity
Insert	See annex A
Replace	Unchanged
Extend	Unchanged
Erase	FADU following the Erased FADU (by preorder traversal sequence)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Section three: Attribute definitions

12 File Attributes

Each file attribute is global, in that it has one value, or one set of values, at any particular time. All initiators of file actions will see the same value, set of values or obtain an indication of "no value available" for a file attribute.

12.1 Filename

Each file in a filestore has a filename. The filename is a vector attribute, the elements of which form a sequence of name components. Each component is a value of type GraphicString (see clause 15).

The value of the filename attribute is set at file creation, but can be altered by the change attribute action.

NOTES

1 ISO 8571 does not define any interpretation for the components of a filename; they provide a transparent naming mechanism to the initiator and the responder.

2 The relation between the components defined in the virtual filestore and any division into components in the real system environment is a local implementation choice. An implementation may map a local component structure onto the components of the filename attribute, or it may choose to map its existing filename syntax into a filename with only one component name in terms of the virtual filestore.

3 An implementation may reflect the virtual filestore filename components in selecting an access path to the file, but this choice is not in itself visible for interconnection purposes.

12.2 Permitted actions

The permitted actions attribute is a vector attribute and indicates the set of actions that can be performed on the file, and the set of FADU- identity styles that can be used in the actions for file access (see 7.6 and 11). The responder implements this set of permitted actions in any way which is capable of mapping them onto the underlying real system.

The different styles of FADU-identities are categorised into three FADU- identity groups as follows:

Traversal: begin, first, next, last, end;

Reverse Traversal: begin, first, previous, last, end;

Random order: current, single Node-Name, sequence of Node-Names, node number.

The attribute is a vector whose elements take boolean values. Each of the elements indicates the availability of an action or a FADU-identity group. The elements are:

a) actions available:
   1) read
   2) insert
   3) replace
   4) extend
   5) erase
   6) read attribute
   7) change attribute
   8) delete file

b) FADU-identity groups available:
   1) traversal
   2) reverse traversal
   3) random order

The value of the permitted actions attribute is set when the file is created, and cannot be changed by the change attribute action.

12.3 Contents type

The contents type attribute indicates the abstract data types of the contents of the file and the structuring information, which is necessary if the complete file structure and semantics are to be maintained during the transfer of the file. The contents type is a scalar attribute.

The value of the contents type attribute is set when the file is created, and cannot be altered by use of the change attribute action.

The value is either a document type name, possibly with parameters in a single value of any type, or a pair of abstract syntax name and constraint set name. Each of these names are values of type OBJECT IDENTIFIER.

An open action for writing will not succeed if contents type information is given with the open action which does not match this attribute. An open action for reading will only succeed if any information given with the open action is equal to, or a simplification or relaxation of, this attribute (see annex B). In more detail, the information given may be:

a) not provided, in which case the value of the attribute is used;

b) an abstract syntax name and constraint set name, in which case the match must be exact;

c) a document type name without parameters, in which case the match must be exact for writing but may be a defined simplification for reading;

d) a document type name with null parameters, in which case the name must match, or be a simplification for reading, but the parameter values are supplied from the attribute value;

e) a document type name with non-null parameters, in which case the name must match, or be a simplification for reading, and the parameters must match or be more restrictive for writing and match or be a relaxation for reading.

12.4 Storage account

The storage account attribute identifies the accountable authority responsible for accumulated file storage charges and is a scalar attribute.

The attribute value is set at file creation, but can be altered by the change attribute action.

The value of the storage account attribute is of type GraphicString.

NOTE - The mechanisms for administering the accounts, and the means by which charging rates are agreed, are outside the scope of this part of ISO 8571.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12.5 Date and time of creation

The date and time of creation attribute indicates when the file was created and is a scalar attribute.

It is set by the responder when the file is created and refers to the local date and time of the responder. The date and time attribute value becomes the date and time when the create action was performed.

The attribute cannot be altered by use of the change attribute action.

The value of the attribute is of type GeneralizedTime (see 9.3).

12.6 Date and time of last modification

The date and time of last modification attribute is a scalar attribute. it indicates when the contents of the file were last modified.

It is altered by the responder whenever the file has been opened for modification or extension and is closed (including closure following a connection failure). The attribute is not altered unless the file is opened to allow change of the contents; it is not altered when the file attributes are changed. The date and time attribute value becomes the date and time when the close action was performed.

The attribute cannot be altered by use of the change attribute action.

The value of the attribute is of type GeneralizedTime (see 9.3). For a newly created file, the value is equal to the value of the date and time of creation attribute.

12.7 Date and time of last read access

The date and time of last read access attribute is a scalar attribute. It indicates when the contents of the file were last read.

It is altered by the responder whenever the file has been opened for reading and is closed (including closure following a connection failure). The date and time attribute value becomes the date and time when the close action was performed. The attribute is not altered unless the file is opened; it is not altered if the file is selected but not opened, for example to read its attributes.

The attribute cannot be altered by use of the change attribute action.

The value of the attribute is of type GeneralizedTime (see 9.3). For a newly created file, the value is equal to the value of the date and time of creation attribute.

12.8 Date and time of last attribute modification

The date and time of last attribute modification attribute is a scalar attribute. It indicates when the value of a file attribute was last modified.

It is altered by the responder whenever the change attribute action is successfully performed on one or more attributes. The attribute is not modified by an implicit change to an attribute, such as Filesize. The date and time attribute value becomes the date and time when the change attribute action was performed.

The attribute cannot be altered by the change attribute action.

The value of the attribute is of type GeneralizedTime (see 9.3). For a newly created file, the value is equal to the date and time of creation attribute.

12.9 Identity of creator

The identity of creator attribute is a scalar attribute. It is set by the responder when the file is created. The identity is set to the value of the current initiator identity activity attribute when the file is created.

The attribute cannot be altered by use of the change attribute action.

The value is of type GraphicString.

12.10 Identity of last modifier

The identity of last modifier attribute is a scalar attribute. It is altered by the responder whenever the file has been opened for modification or extension and is closed (including closure following a connection failure). The identity is set to the value of the current initiator identity activity attribute when the file is closed.

The attribute cannot be altered by use of the change attribute action.

The value is of type GraphicString. For a newly created file, the value is equal to the value of the identity of creator attribute.

12.11 Identity of last reader

The identity of last reader attribute is a scalar attribute. It is altered by the responder whenever the file has been opened for reading and is closed (including closure following a connection failure). The identity is set to the value of the current initiator identity activity attribute when the file is closed.

The attribute cannot be altered by use of the change attribute action.

The value is of type GraphicString. For a newly created file, the value is equal to the value of the identity of creator attribute.

12.12 Identity of last attribute modifier

The identity of last attribute modifier attribute is a scalar attribute. It is altered by the responder whenever the change attribute action is successfully performed on one or more attributes. The identity is set to the value of the Current Initiator Identity.

The attribute cannot be altered by use of the change attribute action.

The value is of type Graphicstring. For a newly created file, the value is equal to the value of the identity of creator attribute.

12.13 File availability

The file availability attribute is a scalar attribute. It indicates whether delay should be expected before the file can be opened.

The attribute value is set at file creation, but can be altered by the change attribute action.

The value may be either "immediate availability" or "deferred availability".

NOTE - The attribute indicates a classification of files which is meaningful to the responder, and not a quantitative measure of delay. If the value is initially set to "immediate availability", it is assumed that the file is stored on a non-demountable device, and no significant delay will be encountered when accessing the file. If the initiator sets the value to "deferred availability", it may be an indication to the responder that the file can be stored on a demountable device. If the initiator resets the attribute to "immediate availability", it may indicate to a responder to copy the

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

file to an immediately available device in case the file has been off-loaded to a demountable device. The actual use of this attribute is, however, implementation dependent.

12.14 Filesize

The filesize attribute is a scalar attribute. It is altered by the responder whenever the file is closed after having been opened for modification and extension (including closure following a connection failure). The attribute value is set to the nominal size in octets of the complete file when the file is closed (see note 2).

The attribute cannot be altered by use of the change attribute action, or set using the initial attributes parameter on the create action.

The value of the attribute is an integer. For a newly created file, the value is set to zero.

NOTES

1 The filesize is based on the real filestore's particular representation of the data types involved. The size of a file may vary, depending on the transfer syntax negotiated, when the data is moved or stored in a different form.

2 If a real filestore allocates space for a file in multiples of a basic unit (e.g., block, bucket), it may be unable to determine the accurate number of octets in a file. The filesize may then assume only multiples of the basic unit (in octets).

12.15 Future filesize

The future filesize attribute is a scalar attribute. It indicates the nominal size in octets to which the file may grow as a result of modification and extension.

The attribute value is set at file creation, but can be altered by the change attribute action.

The value of the attribute is an integer.

NOTE - When the value of the current filesize attribute attains the value of the future filesize attribute, the responder may:

a) increase the future filesize;

b) increase the future filesize and give a warning;

c) not increase the future filesize, but indicate an error.

12.16 Access control

The access control attribute is a set attribute. It defines conditions under which access to the file is valid.

Each element of the set gives one condition under which access to the file is valid. Access to the file is allowed if at least one of these conditions is satisfied. However, the access must be based on a single condition, not on the union of a number of separate conditions.

The attribute value is set at file creation, but can be altered by the change attribute action.

A condition consists of one or two terms giving a statement of the access it allows (see (a) below), together with a set of from zero to three other terms giving tests based on the values of various activity attributes (see (b) below). The whole condition is satisfied only if all the terms in it are satisfied.

In detail these are:

a) the statement of access, consisting of

1) an action list term, in the form of a boolean vector; an action is present in the list if the corresponding boolean is true; the booleans correspond to the actions "read", "insert", "replace", "extend", "erase," "read attribute", "change attribute" and "delete file".

The action list term is used whenever an attempt is made to set the current access request attribute. The action list term is satisfied only if all the actions in the proposed value for the current access request attribute are present in the action list.

If the action list term is satisfied, the "select" and "deselect" actions are also possible. If the satisfied action list term contains file access actions (see clause 11), the actions "open" and "close" are also possible. The action "locate" is then also allowed, if it is available in the FTAM regime established.

2) optionally, a concurrency access term, in the form of a vector of concurrency keys; the concurrency keys correspond to the actions "read", "insert", "replace", "extend", "erase," "read attribute", "change attribute" and "delete file". A concurrency key is a boolean vector whose elements correspond to the possible values of concurrency locks (see 13.9); the elements in the concurrency key correspond to the lock values "not required", "shared", "exclusive" and "no access".

The concurrency access term is used whenever an attempt is made to set the current concurrency control attribute. The concurrency access term is satisfied only if the boolean corresponding to the lock requested in the proposed value for the current concurrency control attribute is true in the concurrency key.

If this term is omitted from the access control element, the concurrency control parameter shall not be present in access allowed by this access control element.

b) and the tests are terms consisting of:

1) optionally, an identity; the value is a GraphicString. The term is satisfied if it matches the value of the current initiator identity activity attribute for the association (see 13.3).

2) optionally, passwords in the form of a vector of strings (one or more of which may be empty); each element of the vector corresponds to one of the actions "read", "insert", "replace", "extend", "erase," "read attribute", "change attribute" and "delete file". The term is satisfied if each non-null string in the vector matches the corresponding password, held in the current access passwords attribute.

3) optionally, a location; the value is an application entity title. The term is satisfied if it matches the value of the current calling application entity title attribute for the FTAM regime (see 13.6).

NOTES

1 The initial value of the access control attribute when the file is created is not defined in this standard.

2 The attribute can be supported by implementations which allow only one access control condition at a time (a list of length one).

3 Some implementations may impose restrictions on the way terms may be combined in conditions, or on the number of conditions of various forms that may exist. For example, an implementation may allow only one password per permitted access type.

4 Matching of identity, password or application entity title is not necessarily on the basis of textual identity. For example, the name of an individual may match the name of a group if the individual is a member of that group.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12.17 Legal qualifications

The legal qualifications attribute is a scalar attribute. It conveys information about the legal status of the file and its use.

The attribute value is set at file creation. It can be altered by the change attribute action.

The value of the attribute is of type GraphicString.

NOTE - The implication of supporting this attribute will depend on the form of national data protection legislation.

12.18 Private use

The private use attribute is a scalar attribute. Its meaning is not defined in this part of ISO 8571.

The attribute value can be set at file creation. It can be altered by the change attribute action.

The value of the attribute can take any form.

NOTE - The use of this attribute is strongly discouraged. It is provided for use in circumstances where no combination of attributes within the standard can convey the intended semantics.

13 Activity attributes

The activity attributes are used as a descriptive tool to reflect the state of the FTAM regime in progress. There is a set of activity attribute values for each initialized FTAM regime. The attribute values reflect the state of the filestore access only, not the detailed state of the communication path to the filestore.

13.1 Active contents type

The active contents type attribute is a scalar attribute identifying either the document type name or the pair of abstract syntax name and constraint set name established for the transfer of the file. The value is established when the file is opened, and is derived from the contents type attribute, or from an overriding value provided by the initiator.

The value is either a document type name, possibly with parameters in a single value of any type, or an abstract syntax name/constraint set name pair. Each of these names are values of type OBJECT IDENTIFIER.

The scope of the active contents type attribute is the open regime.

13.2 Current access request

The current access request attribute is a vector attribute whose elements take boolean values. There is one element for each action which may or may not be possible in a particular select regime, and thus be requested during the FTAM regime.

The elements for file access actions are:

a) "read", allowing the file access action read;

b) "insert", allowing the file access action insert;

c) "replace", allowing the file access action replace;

d) "extend", allowing the file access action extend;

e) "erase", allowing the file access action erase;

If any file access action is allowed, the actions "open" and "close" are also possible. The action "locate" is then also allowed, if it is available in the FTAM regime established.

The elements for actions on complete files are:

f) "read attribute", allowing the file action read attribute;

g) "change attribute", allowing the file action change attribute;

h) "delete file", allowing the file action delete file.

If any file action is allowed, the "select" and "deselect" actions are also possible.

The scope of the current access request attribute is the select regime.

13.3 Current initiator identity

The current initiator identity attribute is a scalar attribute. It identifies the initiator to the responder when the FTAM regime is established.

The value of the attribute is of type GraphicString.

The scope of the current initiator identity attribute is the FTAM regime.

13.4 Current location

The current location attribute indicates the current position within the file, and is a scalar attribute. The value of the attribute may be modified by any action (locate, erase, read, insert) within the open regime. The initial value of current location at initialization is defined by the constraint set in operation (see annex A).

The values may be either an integer identifying the "n"th node in the preorder traversal sequence, or a list of FADU names identifying the path to that node (see 7.6).

The scope of the current location attribute is the open regime.

13.5 Current processing mode

The current processing mode attribute is a vector attribute whose elements take boolean values. There is one element for each file access action allowed in a particular open regime.

The elements are

a) "read", allowing the file access action read;

b) "insert", allowing the file access action insert;

c) "replace", allowing the file access action replace;

d) "extend", allowing the file access action extend;

e) "erase", allowing the file access action erase.

The file access action "locate" is always valid, if it is available in the FTAM regime established.

The scope of the current processing mode attribute is the open regime.

13.6 Current calling application entity title

The current calling application entity title attribute is a scalar attribute. It indicates the application entity title of the application entity which established the FTAM regime.

The value of the attribute is an application entity title.

The scope of the current calling application entity title attribute is the FTAM regime.

13.7 Current responding application entity title

The current responding application entity title is a scalar attribute. It indicates the application entity title of the application entity which corresponds to the virtual filestore.

The value of the attribute is an application entity title.

The scope of the current responding application entity title attribute is the FTAM regime.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

13.8 Current account

The current account attribute is a scalar attribute. It identifies the accountable authority responsible for charges levied by the responder associated with the use of files during the FTAM regime.

The value of the attribute is of type GraphicString.

The scope of the current account attribute is the FTAM regime, however an alternative value may be used within the select regime to accumulate charges for actions on a specific file. In this case, the value is established when the file is selected and the previous value restored when the file is deselected.

NOTE - The attribute does not necessarily indicate responsibility for communication oriented charges levied by the file service provider.

13.9 Current concurrency control

The current concurrency control attribute indicates the restrictions on parallel access as requested by the initiator for each of the requested access actions defined in 13.2.

The current concurrency control attribute is a vector attribute; its elements correspond to the access actions listed in 13.2. The value of each element of the vector is one of the four values given in table 3.

Table 3 — Concurrency control options

	I may perform the action	Others may perform the action
Not Required	No	Yes
Shared	Yes	Yes
Exclusive	Yes	No
No Access	No	No

The scope of the current concurrency control attribute is either the select regime or the open regime depending on whether its value is established on the select action or on the open action.

13.10 Current locking style

The current locking style attribute is a scalar attribute. It indicates whether or not the FADU locking mechanisms have been enabled. The value is derived from the Enable FADU locking parameter when the file is opened.

The value of the attribute is a boolean; it is true if FADU locking is enabled.

The scope of the current locking style attribute is the open regime.

13.11 Current access passwords

The current access passwords attribute is a vector attribute. Each element gives a value associated with one of the eight elements of the current access request attribute. These values are set from the parameters of the select or create file service primitives.

The values become unset when the file selection regime is released. The values are verified against values in the access control attribute (see 12.16) to determine whether access is to be permitted.

The value of each element of the attribute is of type GraphicString or OCTET STRING.

The scope of the current access passwords attribute is the select regime.

NOTE - The attribute may be used to convey passwords, or capabilities, or elements of a public key encryption scheme.

13.12 Active legal qualification

The active legal qualification is a scalar attribute. Its meaning is not defined in ISO 8571, nor is its relationship to the file attribute Legal qualifications. The value is established when the file is opened, and is derived from the legal qualifications attribute.

The value of the attribute is of type GraphicString.

The scope of the active legal qualification attribute is the open regime.

14 Attribute groups

This part of ISO 8571 specifies a wide range of attributes which an implementation may support (see 9.4). For the purposes of specifying levels of support, these are placed into groups as specified below. For each group either

a) the entire group has no support; or

b) each attribute in the group is supported or partially supported, and when interrogated provides either a meaningful response or "no value available".

NOTE - In the FTAM protocol (ISO 8571-4) the attribute groups available are indicated at FTAM regime establishment.

14.1 Kernel group

The following attributes are always defined, and are available in any specification which references the virtual filestore. For these attributes, the value "no value available" cannot be provided:

a) file attributes
   1) filename
   2) permitted actions
   3) contents type

b) activity attributes
   1) active contents type
   2) current access request
   3) current initiator identity
   4) current location
   5) current processing mode
   6) current calling application entity title
   7) current responding application entity title

14.2 Storage group

The following attributes are only meaningful if for fully supported attributes the responder guarantees the storage of information, and allows one activity to reference information established by some previous activity. Any of these attributes may take the value "no value available". Specifying the storage group allows reference to the definitions of

a) file attributes
   1) storage account
   2) date and time of creation
   3) date and time of last modification
   4) date and time of last read access
   5) date and time of last attribute modification

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6) identity of creator
7) identity of last modifier
8) identity of last reader
9) identity of last attribute modifier
10) file availability
11) filesize
12) future filesize

b) activity attributes
1) current account
2) current concurrency control
3) current locking style

14.3 Security group

The following attributes are concerned with security and access control. They form the security group. Any of these attributes may take the value "No value available".

a) file attributes
1) access control
2) legal qualifications

b) activity attributes
1) current access passwords
2) active legal qualification

14.4 Private Group

The following attribute is outside the scope of OSI standardization. It is free standing, and may only be specified if the private use group is specified.

a) file attributes
1) private use

15 Minimum attribute ranges

This clause defines a restricted set of attribute ranges called the "virtual filestore minimum attribute ranges", which are the minimum required for interworking when an attribute is supported (see notes). They apply to the attributes defined in clauses 12 and 13 which have general data types such as integer or character string. Where an attribute's value is an enumeration or a vector of enumerations, any of the enumeration values shall be able to be accepted or generated, without giving an error indication, even where only partial support is provided.

For activity attributes which correspond directly to file attributes the same minimum range shall apply to the activity attribute as to the corresponding file attribute.

The character strings are restricted to the printing characters defined by the ASN.1 GraphicString, and shall not include the space character, unless explicitly stated. At least the G0 character set registered as character set register entry 2 shall be supported. It is not required to support any escape sequences.

All ranges are inclusive.

NOTES

1 The minimum attribute ranges enable standards which reference this part of ISO 8571 to specify a minimum set of values which any open implementations, either initiator or responder, are required to transmit or accept. They must also specify for which attributes "No value available" is the only value supported.

2 The virtual filestore minimum attribute ranges may be too restrictive for many specific uses of the file service. The definition of these ranges does not constrain particular implementations in any way from communicating or accepting arbitrary attribute values of the appropriate type outside the defined ranges.

3 The minimum ranges for activity attributes as specified in table 4 actually restrict the value ranges for parameters used in the corresponding service primitives (see ISO 8571-3).

Table 4 — Activity attributes

Attribute	Type	Minimum Attribute Range
Current access request	boolean vector	see note
Current initiator identity	GraphicString	1 to 8 characters
Current location	Node-Descriptor-Data-Element	see note
Current processing mode	boolean vector	see note
Current calling application entity title	application entity title	No minimum required
Current responding application entity title	application entity title	No minimum required
Current account	GraphicString	1 to 8 characters
Current concurrency control	vector of enumeration	see note
Current locking style	boolean	No minimum required
Current access passwords	GraphicString or OCTET STRING	0 to 8 characters or octets

NOTE - No minimum range can be specified for this activity attribute. The values which this attribute can take depends on the capabilities of the initiator and responder and may be further restricted by the constraint set in use, and/or the permitted actions file attribute, and/or the access control file attribute.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 5 — File attributes

Attribute	Type	Minimum Attribute Range
Filename	vector of GraphicStrings each element	single element 1 to 8 upper case letters or numbers starting with a letter
Permitted actions	boolean vector	any non-empty subset of the defined values
Contents type	abstract syntax, constraint set name pair or a document type name	no minimum required
Storage account	GraphicString	1 to 8 characters
Date and time of creation	date and time	as ISO 8601
Date and time of last modification	date and time	as ISO 8601
Date and time of last read access	date and time	as ISO 8601
Date and time of last attribute modification	date and time	as ISO 8601
Identity of creator	GraphicString	1 to 8 characters
Identity of last modifier	GraphicString	1 to 8 characters
Identity of last reader	GraphicString	1 to 8 characters
Identity of last attribute modifier	GraphicString	1 to 8 characters
File availability	enumeration	any non-empty subset of the defined values
Filesize	integer	no minimum required
Future filesize	integer	no minimum required
Access control action list	set of conditions boolean vector	single condition any non-empty subset of the defined values
concurrency access	vector of boolean vectors	any non-empty subset of the defined values
identity passwords location	GraphicString GraphicString or OCTET STRING application entity title	1 to 8 characters 0 to 8 characters or octets all application entity titles
Legal qualifications	GraphicString	0 to 80 GraphicString and/or space characters
Private use	any	not applicable

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex A
File access structure constraint sets

(This annex forms part of the standard.)

A.1 General considerations

A.1.1 Constraint Set function

Clause 7 defines a general hierarchical structure and clauses 8 and 11 define the actions applicable to it. This general 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. Constraint sets are referenced in protocol exchanges by names which are values of ASN.1 type object identifier.

The set of actions that are ultimately allowable in the file 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, and the functional units selected for the current FTAM regime.

Additional constraints on the file contents, beyond those specified in the constraint set, can also be applied through specification in the document type (for example, the document type may further restrict available data types).

A.1.2 Relation to bulk data transfer

Several FADUs may be inserted, replaced or extended as a result of a single bulk data transfer by repeatedly applying the actions described (see clause 8 and annex D).

A.1.3 Constraint Set content

The definition of a constraint set consists of:

a) a statement of the intended field of application;

b) a constraint set descriptor for reference in documentation;

c) a constraint set identifier for reference in the FTAM protocol;

d) constraints on the node names;

e) the set of valid file access actions;

f) any qualified actions when writing to the file (see clause 8);

g) the set of file access contexts in which the file can be read;

h) the state of the file on creation;

i) FADU located immediately after the file open action;

j) the definition of "beginning of file";

k) the definition of "end of file";

l) the way the whole file is read;

m) the way the whole file is written;

n) any general constraints on how the structure can be modified;

o) any special semantics of specific file access actions, over and above that given in clause 11;

p) FADU identity forms allowable for each of the file access actions.

A.1.4 Notation

In the tables which follow, the following abbreviations indicate the valid scope of FADU Identity when used with particular filestore actions:

a) **valid**: the action may be applied to any FADU;

b) **leaf**: the action is valid if it yields an FADU which is a leaf;

c) **whole**: the action yields the whole file (that is, starting at the root node).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.2 Constraint Set Definitions

A.2.1 Unstructured constraint set

A.2.1.1 Field of Application

The unstructured constraint set applies to files which may be transferred or accessed as a whole, or extended, but for which access to a part is not available.

A.2.1.2 Basic constraints

The basic constraints in the unstructured constraint set are given in table 6.

A.2.1.3 Structural constraints

All actions in the unstructured constraint set result in a structure with a root node which has a data unit (although this data unit may be empty).

A.2.1.4 Action constraints

Erase: the erase action yields root node with an empty data unit.

A.2.1.5 Identity constraints

In the unstructured constraint set, the FADU Identity is always "first". This form of FADU Identity is used with all the permitted file actions.

Table 6 — Basic constraints in the unstructured constraint set

Constraint set descriptor	Unstructured
Constraint set identifier	{iso standard 8571 constraint-set (4) unstructured (1)}
Node names	None
File access actions	Read, Replace, Extend, Erase
Qualified actions	None
Available access contexts	UA
Creation state	Root node with an empty data unit.
Location after open	First
Beginning of file	Not applicable
End of file	Not applicable
Read whole file	Read in access context UA with FADU Identity of "first".
Write whole file	Transfer a single Data Unit (without a node descriptor) with FADU Identity of "first" and file access action of replace.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.2.2 Sequential flat constraint set

A.2.2.1 Field of application

The sequential flat constraint set applies to files which are structured into a sequence of individual FADUs and to which access may be made on an individual basis by position in the sequence.

A.2.2.2 Basic Constraints

The basic constraints in the sequential flat constraint set are given in table 7.

A.2.2.3 Structural constraints

The root node shall not have an associated data unit; all children of the root node shall be leaf nodes and shall have an associated data unit; all arcs from the root node shall be of length one.

A.2.2.4 Action constraints

Insert: the insert action is allowed only at the end of file, with FADU Identity of "end"; the new node is inserted following all existing nodes in the file. The location following insert is "end".

Erase: the erase action is only allowed at the root node to empty the file, with FADU Identity of "begin". The result is an solitary root node without an associated data unit.

A.2.2.5 Identity constraints

The FADU Identity associated with the file action shall be one of the identities begin, end, first, last, current, next, previous or a traversal number greater than or equal to one. The actions with which these identities can be used are given in table 8.

Table 7 — Basic constraints in the sequential flat constraint set

Constraint set descriptor	Sequential flat
Constraint set identifier	{iso standard 8571 constraint-set (4) sequential-flat (2)}
Node names	None
File access actions	Locate, Read, Insert, Erase
Qualified actions	None
Available access contexts	FA, UA
Creation state	Root node without an associated data unit.
Location after open	Root node
Beginning of file	Root node
End of file	No node selected; previous gives last node in traversal sequence, current and next give an error.
Read whole file	Read in access context FA or UA with FADU Identity of "begin".
Write whole file	Transfer the series of leaf FADUs which would be generated by reading the whole file in access context FA; perform the transfer with an FADU Identity of "end" and a file access action of insert.

Table 8 — Identity constraints in the sequential flat constraint set

Action	Begin	End	First	Last	Current	Next	Previous	Traversal
Locate	valid	valid	valid	valid	valid	valid	valid	valid
Read	whole		leaf	leaf	leaf	leaf	leaf	leaf
Insert		leaf						
Erase	whole							

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.2.3 Ordered flat constraint set

A.2.3.1 Field of application

The ordered flat constraint set applies to files which are structured into a sequence of individual FADUs each having names and to which access may be made on a FADU basis by these names. The names are not necessarily unique.

A.2.3.2 Basic constraints

The basic constraints in the ordered flat constraint set are given in table 9.

NOTE - In the write whole file entry, merge combines the file transferred with an existing file, sorting the leaf nodes into a single sequence, while replace discards the existing file data and establishes new file contents.

A.2.3.3 Structural constraints

The root node shall not have an associated data unit and shall not have a node name; all children of the root node shall be leaf nodes and shall have both an associated data unit and a node name; all arcs from the root node shall be of length one.

A.2.3.4 Action constraints

Insert: an insert action with an FADU Identity of begin inserts a new leaf node in the traversal sequence according to the position of the name from the node descriptor (which must be transferred) in the defined ordering of names. If FADUs already exist with the same name value, the new FADU is inserted immediately following them. The location following Insert is begin.

Replace whole file: the FADU Identity is begin and a complete ordered flat FADU replaces the root node.

Extend or Replace leaf node: the semantics depends on the form of the data transferred.

a) If a single Data Unit is transferred without a node descriptor, then the Data Unit in the leaf node addressed by the FADU Identity is changed.

b) If a complete leaf FADU is transferred and the FADU Identity is a sequence of names with a single member, current, previous or next, then the action is performed only if the transferred node name in the node descriptor equals the existing node name.

c) If a complete leaf FADU is transferred and the FADU Identity is begin, then the node named in the transferred node descriptor is changed.

Erase: the erase action with an FADU Identity of begin yields an empty file; the erase action with an FADU Identity of current, next, previous or a sequence of node names with a single member deletes the leaf node thus identified.

NOTE - If multiple nodes exist with the same node name, either the first of these nodes in the traversal order or the current node will be replaced, extented or erased depending on the FADU Identity used.

A.2.3.5 Identity constraints

The FADU Identity associated with the file action shall be one of the identities begin, end, first, last, current, next, previous, or a sequence of node names with a single member, or a traversal number greater than or equal to one. The actions with which these identities can be used are given in table 10.

NOTES

1 If, following a Locate, Read, Insert, Replace or Extend action, or preceding an Erase action, one or more FADUs exist with the same name as the current FADU, a warning diagnostic is

Table 9 — Basic constraints in the ordered flat constraint set

Constraint set descriptor	Ordered flat
Constraint set identifier	{iso standard 8571 constraint-set (4) ordered-flat (3)}
Node names	All names shall be of the same type; the type of the names and an ordering of the names shall be defined when reference is made to the constraint set.
File access actions	Locate, Read, Insert, Replace, Extend, Erase
Qualified actions	None
Available access contexts	HA, FA, UA
Creation state	Root node without an associated data unit.
Location after open	Root node
Beginning of file	Root node
End of file	No node selected; previous gives last node in traversal sequence, current and next give an error.
Read whole file	Read in access context FA with FADU Identity of begin
Write whole file (1): merge	Transfer the series of leaf FADUs which would be generated by reading the whole file in access context FA; perform the transfer with a FADU Identity of begin and a file access action of insert.
Write whole file (2): replace	Transfer the FADU represented by the series of data elements which would be generated by reading the file in access context HA; perform the transfer with a FADU Identity of begin and a file access action of replace.

Table 10 — Identity constraints in the ordered flat constraint set

Action	Begin	End	First	Last	Current	Next	Previous	NodeSeq	Traversal
Locate	valid	valid	valid	valid	valid	valid	valid	valid	valid
Read	whole		leaf	leaf	leaf	leaf	leaf	leaf	leaf
Insert	leaf								
Replace	whole				leaf	leaf	leaf	leaf	
Extend					leaf	leaf	leaf	leaf	
Erase	whole				leaf	leaf	leaf	leaf	

NodeSeq = A sequence of node names with a single member.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

generated. Distinct diagnostics indicate whether the FADU currently located is the last in the traversal sequence with that name or not.

2 The FADU Identities next and previous are provided primarily to access FADUs from sets which share the same name.

3 Usage of traversal numbers is discouraged in this constraint set. The order of FADUs is determined by their node names, and thus the traversal number of a particular node will, in general, change when the file is modified.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.2.4 Ordered flat constraint set with unique names

A.2.4.1 Field of application

The ordered flat constraint set with unique names applies to files which are structured into a sequence of individual FADUs each having unique names and to which access may be made on an individual basis by these names.

A.2.4.2 Basic Constraints

The basic constraints in the ordered flat constraint set with unique names are given in table 11.

NOTE - In the write whole file entry, merge combines the file transferred with an existing file, sorting the leaf nodes into a single sequence, while replace discards the existing file data and establishes new file contents.

A.2.4.3 Structural constraints

The root node shall not have an associated data unit and shall not have a node name; all children of the root node shall be leaf nodes and shall have both an associated data unit and a node name; all arcs from the root node shall be of length one.

A.2.4.4 Action constraints

Insert: an insert action with an FADU Identity of begin inserts a new leaf node in the traversal sequence according to the position of the name from the node descriptor (which must be transferred) in the defined ordering of names. If a FADU already exists with the same name value, the insert action fails. The location following insert is begin.

Replace whole file: the FADU Identity is begin and a complete ordered flat FADU with unique names replaces the root node.

Extend or Replace leaf node: the semantics depends on the form of the data transferred.

a) If a single Data Unit is transferred without a node descriptor, then the Data Unit in the leaf node addressed by the FADU Identity is changed.

b) If a complete leaf FADU is transferred and the FADU Identity is a sequence of node names with a single member or current, then the action is performed only if the transferred node name equals the existing node name.

c) If a complete leaf FADU is transferred and the FADU Identity is begin, then the node named in the transferred node descriptor is changed.

Erase: the erase action with an FADU Identity of begin yields an empty file; the erase action with an FADU Identity of a sequence of node names with a single member or current deletes the leaf node thus identified.

A.2.4.5 Identity constraints

The FADU Identity associated with the file action shall be one of the identities begin, end, current, next, previous, or a sequence of node names with a single member, or a traversal number greater than or equal to one. The actions with which these identities can be used are given in table 12.

NOTE - Usage of traversal numbers is discouraged in this constraint set. The order of FADUs is determined by their node names, and thus the traversal number of a particular node will, in general, change when the file is modified.

Table 11 — Basic constraints in the ordered flat constraint set with unique names

Constraint set descriptor	Ordered flat with unique names
Constraint set identifier	{iso standard 8571 constraint-set (4) ordered-flat-unique-names (4)}
Node names	All names shall be of the same type; the type of the names and an ordering of the names shall be defined when reference is made to the constraint set.
File access actions	Locate, Read, Insert, Replace, Extend, Erase
Qualified actions	None
Available access contexts	HA, FA, UA
Creation state	Root node without an associated data unit.
Location after open	Root node
Beginning of file	Root node
End of file	No node selected; previous gives last node in traversal sequence, current and next give an error.
Read whole file	Read in access context FA with FADU Identity of begin
Write whole file (1): merge	Transfer the series of leaf FADUs which would be generated by reading the whole file in access context FA; perform the transfer with a FADU Identity of begin and a file access action of insert.
Write whole file (2): replace	Transfer the FADU represented by the series of data elements which would be generated by reading the file in access context HA; perform the transfer with a FADU Identity of begin and a file access action of replace.

Table 12 — Identity constraints in the ordered flat constraint set with unique names

Action	Begin	End	Current	Next	Previous	NodeSeq	Traversal
Locate	valid	valid	valid	valid	valid	valid	valid
Read	whole		leaf	leaf	leaf	leaf	leaf
Insert	leaf						
Replace	whole		leaf			leaf	
Extend			leaf			leaf	
Erase	whole		leaf			leaf	

NodeSeq = A sequence of node names with a single member.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.2.5 Ordered hierarchical constraint set

A.2.5.1 Field of application

The ordered hierarchical constraint set applies to files which are hierarchically structured, and in which the children of each node are ordered in the traversal sequence by the position of their names in a defined sequence.

A.2.5.2 Basic constraints

The basic constraints in the ordered hierarchical constraint set are given in table 13.

A.2.5.3 Structural constraints

There are no structural constraints; the full generality of the hierarchical file model is available.

A.2.5.4 Action constraints

Insert: a new node is inserted amongst the children of the located node, being positioned so that the names of all the children of that node are in order. The name of the inserted node must be distinct from that of any node already present as a child of the located node.

Other actions: other actions are exactly as specified in clause 11.

A.2.5.5 Identity constraints

The FADU Identity associated with the file action shall be one of the identities begin, end, current, next, previous, a node name, or a sequence of node names. The actions with which these identities can be used are given in table 12.

NOTES

1 An empty sequence of node names is used to address the root node when performing actions on the whole file.

2 It is expected that next and previous will generally be used in access context FS, or for accessing a series of leaf nodes of a single parent.

Table 13 — Basic constraints in the ordered hierarchical constraint set

Constraint set descriptor	Ordered hierarchical
Constraint set identifier	{iso standard 8571 constraint-set (4) ordered-hierarchical (5)}
Node names	All names shall be of the same type; the name of each FADU shall be unique with respect to the names of its sisters; the type of the names and an ordering of the names shall be defined when reference is made to the constraint set.
File access actions	Locate, Read, Insert, Replace, Extend, Erase
Qualified actions	None
Available access contexts	HA, HN, FA, FL, FS, UA, US
Creation state	Root node without an associated data unit.
Location after open	Beginning of file
Beginning of file	No node selected; next gives the first node in the traversal sequence (the root node); current and previous give an error.
End of file	No node selected; previous gives the last node in traversal sequence; current and next give an error.
Read whole file	Read in access context HA with FADU Identity consisting of an empty sequence of node names.
Write whole file	Transfer the FADU as the sequence of data-elements which would be generated by reading the file in access context HA; perform the transfer with a FADU Identity consisting of an empty sequence of node names and a file access action of replace.

Table 14 — Identity constraints in the ordered hierarchical constraint set

Action	Begin	End	Current	Next	Previous	NodeName	NodeSeq
Locate	valid	valid	valid	valid	valid	valid	valid
Read			valid	valid	valid	valid	valid(note1)
Insert			valid			valid	valid
Replace			valid			valid	valid(note1)
Extend			valid			valid	valid
Erase			valid			valid	valid(note1)

NodeSeq = A sequence of node names.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.2.6 General Hierarchical constraint set

A.2.6.1 Field of application

The general hierarchical constraint set applies to files which are hierarchically structured, with the full generality of the generic structure.

A.2.6.2 Basic constraints

The basic constraints in the general hierarchical constraint set are given in table 15.

A.2.6.3 Structural constraints

There are no structural constraints; the full generality of the hierarchical file model is available.

A.2.6.4 Action semantics

Insert: there are two forms of the insert action.

Normal action - Insert as sister: the created FADU is placed in the preorder traversal sequence after the current location so that the arc linking it to its parent node is of the length specified in the node descriptor. The inserted FADU and the current location share the same parent node.

Insert as sister cannot be performed if the current location is the root of the file.

The current location after the action is the root-node of the newly inserted FADU.

Qualified action - Insert as child: on beginning a series of qualified actions, a copy of the current location is recorded, and a new location established which is in the traversal sequence between the previous location and the next node in the traversal sequence (that is, before the first child of the previously located node, if any).

Qualified actions are then performed with the same semantics as the normal insert action insert as sister, but using this temporary location.

At the end of a series of qualified actions, the current location is restored from the copy of the location taken at the beginning of the series of qualified actions.

Other actions: other actions are exactly as specified in clause 11.

NOTE - If multiple nodes exist with the same node name, either the first of these nodes in the traversal order or the current node will be replaced, extented or erased, depending on the FADU Identity used.

A.2.6.5 Identity constraints

The FADU Identity associated with the file action shall be one of the identities begin, end, current, next, previous, a node name, or a sequence of node names. The actions with which these identities can be used are given in table 16.

NOTES

1 An empty sequence of node names is used to address the root node when performing actions on the whole file.

2 If, following a Locate, Read, Insert, Replace or Extend action, or preceding an Erase action, one or more FADUs exist with the same name and parent as the current FADU, a warning diagnostic is generated. Distinct diagnostics indicate whether the FADU currently located is the last in the traversal sequence with that name or not.

3 The FADU Identities next and previous are provided primarily to access FADUs from sets which share the same name.

4 It is expected that next and previous will generally be used in access context FS, or for accessing a series of leaf nodes of a single parent.

Table 15 — Basic constraints in the general hierarchical constraint set

Constraint set descriptor	General hierarchical
Constraint set identifier	{iso standard 8571 constraint-set (4) general-hierarchical (6)}
Node name	The types of the names shall be defined when reference is made to the constraint set; all names at a single level of the hierarchy shall be of the same type.
File access actions	Locate, Read, Insert, Replace, Extend, Erase
Qualified actions	**Insert**: normal action "insert as sister" qualified action "insert as child"
Available access contexts	HA. HN, FA, FL, FS, UA, US
Creation state	Root node without an associated data unit.
Location after open	Beginning of file
Beginning of file	No node selected; next gives the first node in the traversal sequence (the root node); current and previous give an error.
End of file	No node selected; previous gives the last node in traversal sequence; current and next give an error.
Read whole file	Read in access context HA with FADU Identity consisting of an empty sequence of node names.
Write whole file	Transfer the FADU as the sequence of data-elements which would be generated by reading the file in access context HA; perform the transfer with a FADU Identity consisting of an empty sequence of node names and a file access action of replace.

Table 16 — Identity constraints in the general hierarchical constraint set

Action	Begin	End	Current	Next	Previous	NodeName	NodeSeq
Locate	valid	valid	valid	valid	valid	valid	valid
Read			valid	valid	valid	valid	valid(note1)
Insert			valid	valid	valid	valid	valid
Replace			valid	valid	valid	valid	valid(note1)
Extend			valid	valid	valid	valid	valid
Erase			valid	valid	valid	valid	valid(note1)

NodeSeq = A sequence of node names.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.2.7 General hierarchical constraint set with unique names

A.2.7.1 Field of application

The general hierarchical constraint set with unique names applies to files which are hierarchically structured, with the full generality of the generic structure, except that the node names of the children of any given parent are unique.

A.2.7.2 Basic constraints

The basic constraints in the general hierarchical constraint set with unique names are given in table 17.

A.2.7.3 Structural constraints

There are no structural constraints; the full generality of the hierarchical file model is available.

A.2.7.4 Action constraints

Insert: there are two forms of the insert action. The name of the inserted node must be distinct from that of any node already present as a sister of the inserted node.

Normal action - Insert as sister: the created FADU is placed in the preorder traversal sequence after the current location so that the arc linking it to its parent node is of the length specified in the node descriptor. The inserted FADU and the current location share the same parent node.

Insert as sister cannot be performed if the current location is the root of the file.

The current location after the action is the root-node of the newly inserted FADU.

Qualified action - Insert as child: on beginning a series of qualified actions, a copy of the current location is recorded, and a new location established which is in the traversal sequence between the previous location and the next node in the traversal sequence (that is, before the first child of the previously located node, if any).

Qualified actions are then performed with the same semantics as the normal insert action insert as sister, but using this temporary location.

At the end of a series of qualified actions, the current location is restored from the copy of the location taken at the beginning of the series of qualified actions.

Other actions: other actions are exactly as specified in clause 11.

A.2.7.5 Identity constraints

The FADU Identity associated with the file action shall be one of the identities begin, end, current, next, previous, a node name, or a sequence of node names. The actions with which these identities can be used are given in table 18.

NOTES

1 An empty sequence of node names is used to address the root node when performing actions on the whole file.

2 It is expected that next and previous will generally be used in access context FS, or for accessing a series of leaf nodes of a single parent.

Table 17 — Basic constraints in the general hierarchical constraint set with unique names

Constraint set descriptor	General hierarchical with unique names
Constraint set identifier	{iso standard 8571 constraint-set (4) general-hierarchical-unique-names (7)}
Node name	The types of the names shall be defined when reference is made to the constraint set; all names at a single level of the hierarchy shall be of the same type. The names of the children of any particular parent shall be unique.
File access actions	Locate, Read, Insert, Replace, Extend, Erase
Qualified actions	**Insert**: normal action "insert as sister" qualified action "insert as child"
Available access contexts	HA, HN, FA, FL, FS, UA, US
Creation state	Root node without an associated data unit.
Location after open	Beginning of file
Beginning of file	No node selected; next gives the first node in the traversal sequence (the root node); current and previous give an error.
End of file	No node selected; previous gives the last node in traversal sequence; current and next give an error.
Read whole file	Read in access context HA with FADU Identity consisting of an empty sequence of node names.
Write whole file	Transfer the FADU as the sequence of data-elements which would be generated by reading the file in access context HA; perform the transfer with a FADU Identity consisting of an empty sequence of node names and a file access action of replace.

Table 18 — Identity constraints in the general hierarchical constraint set with unique names

Action	Begin	End	Current	Next	Previous	NodeName	NodeSeq
Locate	valid	valid	valid	valid	valid	valid	valid
Read			valid	valid	valid	valid	valid(note1)
Insert			valid			valid	valid
Replace			valid			valid	valid(note1)
Extend			valid			valid	valid
Erase			valid			valid	valid(note1)

NodeSeq = A sequence of node names.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex B
Document types

(This annex forms part of the standard.)

This annex contains a number of common document type definitions, to allow the implementations of FTAM to interwork.

NOTE - These definitions are included in this part of ISO 8571 pending the establishment of an ISO document type Registration Authority. Once this authority has been established, it is expected that these definitions will be included in a document type register, and that future editions of this part of ISO 8571 will omit this annex. The use of document types is not required for the use of FTAM; the contents of a file may be specified in terms of an abstract syntax and a constraint set to be applied to the file model.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.1 Unstructured text file document types

1 Entry number: FTAM-1

2 Information objects

Table 19 — Information objects in the unstructured text document type

document type name	{ iso standard 8571 document-type (5) unstructured-text (1) } "ISO FTAM unstructured text"
abstract syntax names a) name of asname1	{iso standard 8571 abstract-syntax (2) unstructured-text (3)} "FTAM unstructured text abstract syntax"
transfer syntax names	{joint-iso-ccitt asn1 (1) basic-encoding (1)} "Basic encoding of a single ASN.1 type"
Parameter syntax PARAMETERS ::= SEQUENCE {	universal-class-number [0] IMPLICIT INTEGER OPTIONAL, maximum-string-length [1] IMPLICIT INTEGER OPTIONAL, string-significance [2] IMPLICIT INTEGER {variable(0), fixed(1), not-significant(2)} OPTIONAL }
file model	{iso standard 8571 file-model (3) hierarchical (1)} "FTAM hierarchical file model"
constraint set	{iso standard 8571 constraint-set (4) unstructured (1)} "FTAM unstructured constraint set"
File contents Datatype1 ::= CHOICE {	PrintableString -- Universal class 19 --, TeletexString -- Universal class 20 --, VideotexString -- Universal class 21 --, IA5String -- Universal class 22 --, GraphicString -- Universal class 25 --, VisibleString -- Universal class 26 --, GeneralString -- Universal class 27 --}

3 Scope and field of application

These document types define the contents of a file for storage, for transfer and access by FTAM, and for transfer by JTM.

4 References

ISO 8571, *Information processing systems - Open Systems Interconnection - File Transfer, Access and Management.*

ISO 8832, *Information processing systems - Open Systems Interconnection - Specification of the protocol for Basic Class Job Transfer and Manipulation.*

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

5 Definitions

This definition makes use of the terms data element, data unit and file access data unit as defined in ISO 8571-1.

5.1 character string: an ordered series of zero, one or more characters from some specified character repertoire.

5.2 graphics character: a character from any character repertoire registered for use as a G0, G1, G2, or G3 set in the ISO International Register of Character Sets, or the character SPACE.

5.3 format effector: a control function which controls the layout and positioning of information on a printing or display device. Common format effector characters are BS, CR, FF, HT, LF and VT.

6 Abbreviations

FTAM File Transfer, Access, and Management
JTM Job Transfer and Manipulation

7 Document semantics

The document consists of one file access data unit, which consists only of zero, one or more character strings. The order of these elements is significant. The semantics of the character strings is not specified by this document type.

The document structure takes the form allowed by the FTAM hierarchical file model as constrained by the unstructured constraint set (see table 19). These definitions appear in ISO 8571-2.

Each character string consists of characters from the character set defined by the ASN.1 (ISO 8824) character set type whose universal class number is given by the "universal-class-number" parameter. If the "universal-class-number" is not present, the default is GraphicString (25); that is, the character strings may contain characters from any of the character sets (plus SPACE) registered (in the International Register of Coded Character Sets to be used with Escape Sequences) for use as G0, G1, G2, or G3 sets.

There are no size or length limitations imposed by this definition, except those specified here. Each character string is of a length determined by the number of characters given by the "maximum-string-length" parameter. If the "maximum-string-length" parameter is not present, the default is that the character strings are unbounded.

NOTE - The length restriction refers to the number of characters from the applicable character set, not to the number of octets in the encoding, nor to the line length in any rendition of the document, where these are different.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The exact significance of the character strings is determined by the "string-significance" parameter. If its value is "variable", or the parameter is not present, the length of the character strings is less than or equal to the length given. If the value is "fixed", the length of each character string is exactly equal to the length given. If the value is "not-significant", the boundaries of the character strings are not necessarily preserved when the file is stored and do not contribute to the document semantics.

If the document is interpreted on a character imaging device (outside the scope of ISO 8571), the interpretation depends on the character set in use:

a) if the character set contains format effectors, they shall be interpreted as defined in ISO 6429; end of string and end of the file access data unit are given no formatting significance;

b) if the character set does not contain format effectors, the end of each character string is interpreted as implying carriage return and line feed formatting actions in any rendition. The end of the file access data unit is given no formatting significance beyond that attached to the end of the last string in it.

8 Abstract syntactic structure

The abstract syntactic structure of the document is a series of character strings, each of the ASN.1 character string type indicated by the universal-class-number parameter.

9 Definition of transfer

9.1 Datatype definition

The file consists of zero or more values of Datatype1 defined in table 19. The choice in this datatype is determined by the universal-class-number parameter, or by its default value.

9.2 Presentation data values

The document is transferred as a series of presentation data values. Each presentation data value shall consist of one value of the ASN.1 data type "Datatype1", carrying one of the character strings from the document. Each character shall be transmitted using one of the character sets identified by the universal-class-number parameter.

All values are transmitted in the same (but any) presentation context established to support the abstract syntax name "asname1" declared in table 19.

NOTE - Specific carrier standards may impose additional constraints on the presentation context to be used, where the above permits a choice.

Boundaries between P-DATA primitives are chosen locally by the sending entity at the time of transmission, and carry no semantics of the document type. Receivers which support this document type shall accept a document with any of the permitted transfer options.

9.3 Sequence of presentation data values

The sequence of presentation data values is the same as the sequence of character strings within the Data Unit in the file.

10 Transfer syntax

An implementation supporting these document types shall support the transfer syntax generation rules named in table 19 for all presentation data values transferred.

Implementations may optionally support other transfer syntaxes.

11 ASE specific specifications

11.1 ISO 8571 (FTAM)

11.1.1 Simplification and relaxation

11.1.1.1 Character set relaxation

This operation loses explicit information in the document type identification.

A document of type FTAM-1 may be relaxed to a different document of type FTAM-1 with a different "universal-class-number" parameter value, or no parameter value, if the resultant document type permits all characters from the original document type. If this relaxation involves including a set of format effectors, and none were present before the simplification, the characters "carriage return" and "line-feed" shall be added to the end of each string.

NOTE - If the characters "carriage return" and "line feed" are not part of the set of format effectors the formatting action may be represented by "newline", or some other implementation specific choice if there is no representation of "newline" defined.

11.1.1.2 String length relaxation

This operation loses explicit information in the document type identification.

A document of type "FTAM-1" may be relaxed to another document of type "FTAM-1" with a larger "maximum-string-length" parameter, or no "maximum-string-length" parameter (unbounded strings).

11.1.2 The EXTEND operation

When the EXTEND operation is applied to the data unit of an "FTAM-1" document, the transferred data shall be of an "FTAM-1" document type with parameters equal to those of the original "FTAM-1" document type. The resulting document consists of the character strings of the original "FTAM-1" document followed by the character strings of the new "FTAM-1" document. The boundary between the original and new character strings is not visible in the new document.

11.1.3 The REPLACE operation

When the REPLACE operation is applied to the root FADU of an "FTAM-1" document, the transferred material shall be any "FTAM-1" document of a type with the same parameter values.

11.2 ISO 8832 (JTM)

The following clauses apply to any JTM implementation claiming support of FTAM-1.

11.2.1 Support of other document types

Document type "JTM-1" shall also be supported.

NOTE - These document types are semantically identical.

11.2.2 Length and parameters

The implementation shall be capable of handling any "FTAM-1" document type with all values of parameters, subject to a possible implementation limit based on the total number of characters from the specified character sets in the document.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

NOTE - See ISO 8832 for a specification of "support", particularly in relation to character sets.

The implementation limit shall permit the use of documents containing up to 64000 characters. The limitations shall be described in the protocol implementation conformance statement.

11.2.3 Concatenation

Concatenation of a document of type "FTAM-1" with itself shall be supported, and shall produce a document of the same type consisting of the combined sequence of character strings.

NOTE - The boundary of the original documents is only visible in the resulting document as an normal boundary between two character strings.

11.2.4 Refinement

11.2.4.1 Character set and length refinements

Any document requested by J-GIVE using an "FTAM-1" document type shall be obtained from any stored "FTAM-1" document type which is a relaxation of it under the rules of 11.1.1.1 or 11.1.1.2 or both, by processing the document. This processing may produce a document, or may produce a diagnostic. See ISO 8832 for details.

11.2.5 Relaxations

Any document requested by J-GIVE using an "FTAM-1" document type which is a relaxation of the document type under it which it was stored shall be supplied as the requested "FTAM-1" document type.

NOTE - The value in the J-GIVE response is never tightened.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.2 Sequential text file document type

1 Entry number: FTAM-2

2 Information objects

Table 20 — Information objects in the sequential text document type

document type name	{ iso standard 8571 document-type (5) sequential-text (2) } "ISO FTAM sequential text"
abstract syntax names a) name for asname1 b) name for asname2	 {iso standard 8571 abstract-syntax (2) unstructured-text (3)} "FTAM unstructured text abstract syntax" {iso standard 8571 abstract-syntax (2) ftam-fadu (2)} "FTAM FADU"
transfer syntax names	 {joint-iso-ccitt asn1 (1) basic-encoding (1)} "Basic encoding of a single ASN.1 type"
Parameter syntax PARAMETERS ::= SEQUENCE {	universal-class-number [0] IMPLICIT INTEGER OPTIONAL, maximum-string-length [1] IMPLICIT INTEGER OPTIONAL, string-significance [2] IMPLICIT INTEGER {variable(0), fixed(1), not-significant(2)} OPTIONAL }
file model	{iso standard 8571 file-model (3) hierarchical (1)} "FTAM hierarchical file model"
constraint set	{iso standard 8571 constraint-set (4) sequential-flat (2)} "FTAM sequential flat constraint set"
File contents Datatype1 ::= CHOICE {	PrintableString -- Universal class 19 --, TeletexString -- Universal class 20 --, VideotexString -- Universal class 21 --, IA5String -- Universal class 22 --, GraphicString -- Universal class 25 --, VisibleString -- Universal class 26 --, GeneralString -- Universal class 27 --}
Datatype2 ::= CHOICE {	Node-Descriptor-Data-Element, Enter-Subtree-Data-Element, Exit-Subtree-Data-Element}

3 Scope and field of application

These document types define the contents of a file for storage, for transfer and access by FTAM, and for transfer by JTM.

4 References

ISO 8571, *Information processing systems - Open Systems Interconnection - File Transfer, Access and Management.*

ISO 8832, *Information processing systems - Open Systems Interconnection - Specification of the protocol for Basic Class Job Transfer and Manipulation.*

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

5 Definitions

This definition makes use of the terms data element, data unit and file access data unit as defined in ISO 8571-1.

5.1 character string: an ordered series of zero, one or more characters from some specified character repertoire.

5.2 graphics character: a character from any character repertoire registered for use as a G0, G1, G2, or G3 set in the ISO International Register of Character Sets, or the character SPACE.

5.3 format effector: a control function which controls the layout and positioning of information on a printing or display device. Common format effector characters are BS, CR, FF, HT, LF and VT.

6 Abbreviations

FTAM	File Transfer, Access, and Management
JTM	Job Transfer and Manipulation

7 Document semantics

The document consists of zero, one or more file access data units, each of which consists of zero, one or more character strings. The order of each of these elements is significant. The semantics of the character strings is not specified by this document type.

The document structure takes any of the forms allowed by the FTAM hierarchical file model as constrained by the sequential flat constraint set (see table 20). These definitions appear in ISO 8571-2.

Each character string consists of characters from the character set defined by the ASN.1 (ISO 8824) character set type whose universal class number is given by the "universal-class-number" parameter. If the "universal-class-number" is not present, the default is GraphicString (25); that is, the character strings may contain characters from any of the character sets (plus space) registered (in

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

the International Register of Coded Character Sets to be used with Escape Sequences) for use as G0, G1, G2, or G3 sets.

There are no size or length limitations imposed by this definition, except those specified here. Each character string is of a length determined by the number of characters given by the "maximum-string-length" parameter. If the "maximum-string-length" parameter is not present, the default is that the character strings are unbounded.

NOTE - The length restriction refers to the number of characters from the applicable character set, not to the number of octets in the encoding, nor to the line length in any rendition of the document, where these are different.

The exact significance of the character strings is determined by the "string-significance" parameter. If its value is "variable", or the parameter is not present, the length of the character strings is less than or equal to the length given. If the value is "fixed", the length of each character string is exactly equal to the length given. If the value is "not-significant", the boundaries of the character strings are not necessarily preserved when the file is stored and do not contribute to the document semantics.

If the document is interpreted on a character imaging device (outside the scope of ISO 8571), the interpretation depends on the character set in use:

a) if the character set contains format effectors, they shall be interpreted as defined in ISO 6429; end of string and end of file access data unit are given no formatting significance, and do not contribute to the document semantics;

b) if the character set does not contain format effectors, the end of each character string is interpreted as implying carriage return and line feed formatting actions in any rendition. The end of file access data unit is given no formatting significance beyond that attached to the end of the last string in it.

8 Abstract syntactic structure

The abstract syntactic structure of the document is a hierarchically structured file as defined in the ASN.1 modules ISO8571-FADU and ISO8571-CONTENTS in ISO 8571, in which each of the file contents data elements has the abstract syntactic structure of a "ISO FTAM unstructured text" document defined in document type registration entry "FTAM-1" in ISO 8571.

9 Definition of transfer

9.1 Datatype definitions

The file consists of data values which are of either

a) Datatype1 defined in table 20, where the choice in the datatype is determined by the universal-class-number parameter, or by its default value; or

b) Datatype2 defined in table 20, the ASN.1 datatype declared as "Data-Element" in the ASN.1 module ISO8571-FADU.

9.2 Presentation data values

The document is transferred as a series of presentation data values, each of which is either

a) one value of the ASN.1 datatype "Datatype1", carrying one of the character strings from the document. Each character shall be transmitted using one of the character sets identified by the universal-class-number parameter. All values are transmitted in the same (but any) presentation context established to support the abstract syntax name "asname1" declared in table 20.

b) one value of the ASN.1 datatype "Datatype2". All values are transmitted in the same (but any) presentation context established to support the abstract syntax name "asname2" declared in table 20.

NOTES

1 Specific carrier standards may impose additional constraints on the presentation context to be used, where the above permits a choice.

2 Any document type defined in this entry either makes no use of Datatype2, or starts with a Datatype2 transmission.

Boundaries between P-DATA primitives are chosen locally by the sending entity at the time of transmission, and carry no semantics of the document type. Receivers which support this document type shall accept a document with any of the permitted transfer options.

9.3 Sequence of presentation data values

The sequence presentation data values of type (a) and the sequence of presentation data values of types (a) and (b) is the same as the sequence of character strings within a Data Unit, and Data Units in the hierarchical structure, when flattened according to the definition of the hierarchical file model in ISO 8571-2.

10 Transfer syntax

An implementation supporting these document types shall support the transfer syntax generation rules named in table 20 for all presentation data values transferred.

Implementations may optionally support other transfer syntaxes.

11 ASE specific specifications

11.1 ISO 8571 (FTAM)

11.1.1 Simplification and relaxation

11.1.1.1 Structural simplification

This simplification loses information.

The document type FTAM-2 may be simplified to the document type FTAM-1. The resultant document contains the same sequence of data values as would result from accessing the structured text file in access context UA. That is, only the presentation data values in the abstract syntax "asname1" are present.

NOTE - The boundary between file access data units remains a boundary between strings, but any special significance given to it is lost.

11.1.1.2 Character set relaxation

This operation loses explicit information in the document type identification.

A document of type FTAM-2 may be relaxed to a different document of type FTAM-2 with a different "universal-class-number" parameter value, or no parameter value, if the resultant document type permits all characters from the original document type. If this relaxation involves including format effectors and none were present before the simplification, the characters "carriage return" and "line-feed" shall be added to the end of each string.

NOTE - If the characters "carriage return" and "line feed" are not part of the set of format effectors, the formatting action may be

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

represented by "newline", or some other implementation specific choice if there is no representation of "newline" defined.

11.1.1.3 String length relaxation

This operation loses explicit information in the document type identification.

A document of type "FTAM-2" may be relaxed to another document of type "FTAM-2" with a larger "maximum-string-length" parameter, or no "maximum-string-length" parameter (unbounded strings).

11.1.2 Access context selection

A document of type "FTAM-2" may be accessed in any one of the access contexts defined in the sequential flat constraint set. The presentation data units transferred in each case are those derived from the structuring elements defined for that access context in ISO 8571-2.

In access contexts FA and HA the resultant document conforms to type "FTAM-2". In access context UA the resultant document type conforms to "FTAM-1".

11.1.3 The INSERT operation

When the INSERT operation is applied at end of file, the transferred material shall be the series of FADUs which would be generated by reading any "FTAM-2" document type with the same parameter values in access context FA.

11.2 ISO 8832 (JTM)

The following clauses apply to any JTM implementation claiming support of FTAM-2.

11.2.1 Support of other document types

Document types "FTAM-1" and "JTM-1" shall also be supported.

NOTE - These document types are semantically identical.

11.2.2 Length and parameters

The implementation shall be capable of handling any "FTAM-2" document type with all values of parameters, subject to a possible implementation limit based on the total number of characters from the specified character sets in the document.

NOTE - See ISO 8832 for a specification of "support", particularly in relation to character sets.

The implementation limit shall permit the use of documents containing up to 64000 characters. The limitations shall be described in the protocol implementation conformance statement.

11.2.3 Concatenation

Concatenation of a document of type "FTAM-2" with itself shall be supported, and shall produce a document of the same type consisting of the combined sequence of data units.

NOTE - The boundary of the original documents is only visible in the resulting document as an normal boundary between two data units.

11.2.4 Refinement

11.2.4.1 Character set and length refinements

Any document requested by J-GIVE using an "FTAM-2" document type shall be obtained from any stored "FTAM-2" document type which is a relaxation of it under the rules of 11.1.1.2 or 11.1.1.3 or both, by processing the document. This processing may produce a document, or may produce a diagnostic. See ISO 8832 for details.

11.2.4.2 Structural refinement

Any document requested by J-GIVE using an "FTAM-2" document type shall be obtained from any stored "FTAM-1" document type by treating the entire "FTAM-1" document as a single data unit (the only data unit) in the "FTAM-2" document.

11.2.5 Structural simplification

Any document requested by J-GIVE using an "FTAM-1" document type shall be obtained from any "FTAM-2" document type by applying the structural simplification specified in 11.1.1.1.

11.2.6 Relaxations

Any document requested by J-GIVE using an "FTAM-2" document type which is a relaxation of the document type under it which it was stored shall be supplied as the requested "FTAM-2" document type.

NOTE - The value in the J-GIVE response is never tightened.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.3 Unstructured binary file document type

1 Entry number: FTAM-3

2 Information objects

Table 21 — Information objects in the unstructured binary document type

document type name	{ iso standard 8571 document-type (5) unstructured-binary (3) } "ISO FTAM unstructured binary"
abstract syntax names a) name for asname1	{iso standard 8571 abstract-syntax (2) unstructured-binary (4)} "FTAM unstructured binary abstract syntax"
transfer syntax names	{joint-iso-ccitt asn1 (1) basic-encoding (1)} "Basic encoding of a single ASN.1 type"
Parameter syntax PARAMETERS ::= SEQUENCE {	maximum-string-length [1] IMPLICIT INTEGER OPTIONAL, string-significance [2] IMPLICIT INTEGER {variable(0), fixed(1), not-significant(2)} OPTIONAL }
file model	{iso standard 8571 file-model (3) hierarchical (1)} "FTAM hierarchical file model"
constraint set	{iso standard 8571 constraint-set (4) unstructured (1)} "FTAM unstructured constraint set"
File contents Datatype1 ::=	OCTET STRING

3 Scope and field of application

These document types define the contents of a file for storage, for transfer and access by FTAM, and for transfer by JTM.

4 References

ISO 8571, *Information processing systems - Open Systems Interconnection - File Transfer, Access and Management.*

ISO 8832, *Information processing systems - Open Systems Interconnection - Specification of the protocol for Basic Class Job Transfer and Manipulation.*

5 Definitions

This definition makes use of the terms data element, data unit and file access data unit as defined in ISO 8571-1.

5.1 binary string: an ordered series of zero, one or more octets.

6 Abbreviations

FTAM	File Transfer, Access, and Management
JTM	Job Transfer and Manipulation

7 Document semantics

The document consists of one file access data unit, which consists only of zero, one or more binary strings. The order of these elements is significant. The semantics of the binary strings is not specified by this document type.

The document structure takes the form allowed by the FTAM hierarchical file model as constrained by the unstructured constraint set (see table 21). These definitions appear in ISO 8571-2.

Each binary string consists of octets of any value from 0 to 255. The meaning attached to these values is not constrained by this document type.

There are no size or length limitations imposed by this definition, except those specified here. Each binary string is of a length determined by the number of octets given by the "maximum-string-length" parameter. If the "maximum-string-length" parameter is not present, the binary strings are unbounded.

The exact significance of the binary strings is determined by the "string-significance" parameter. If its value is "variable" the length of the binary strings is less than or equal to the length given. If the value is "fixed", the length of each binary string is exactly equal to the length given. If the value is "not-significant", or the parameter is not present, the boundaries of the binary strings are not necessarily preserved when the file is stored and do not contribute to the document semantics.

8 Abstract syntactic structure

The abstract syntactic structure of the document is a series of ASN.1 datatypes of type OCTET STRING.

9 Definition of transfer

9.1 Datatype definition

The file consists of zero or more values of Datatype1 defined in table 21.

9.2 Presentation data values

The document is transferred as a series of presentation data values. Each presentation data value shall consist of one value of the ASN.1 data type "Datatype1", carrying one of the binary strings from the document.

All values are transmitted in the same (but any) presentation context established to support the abstract syntax name "asname1" declared in table 21.

NOTE - Specific carrier standards may impose additional constraints on the presentation context to be used, where the above permits a choice.

Boundaries between P-DATA primitives are chosen locally by the sending entity at the time of transmission, and carry

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

no semantics of the document type. Receivers which support this document type shall accept a document with any of the permitted transfer options.

9.3 Sequence of presentation data values

The sequence of presentation data values is the same as the sequence of binary strings within the Data Unit in the file.

10 Transfer syntax

An implementation supporting these document types shall support the transfer syntax generation rules named in table 21 for all presentation data values transferred.

Implementations may optionally support other transfer syntaxes.

11 ASE specific specifications

11.1 ISO 8571 (FTAM)

11.1.1 Simplification and relaxation

11.1.1.1 String length relaxation

This operation loses explicit information in the document type identification.

A document of type "FTAM-3" may be relaxed to another document of type "FTAM-3" with a larger "maximum-string-length" parameter, or no "maximum-string-length" parameter (unbounded strings).

11.1.2 The EXTEND operation

When the EXTEND operation is applied to the data unit of an "FTAM-3" document, the transferred data shall be of an "FTAM-3" document type with parameters equal to those of the original "FTAM-3" document type. The resulting document consists of the binary strings of the original "FTAM-3" document followed by the binary strings of the new "FTAM-3" document. The boundary of the original and new binary strings is not visible in the new document.

11.1.3 The REPLACE operation

When the REPLACE operation is applied to the root FADU of an "FTAM-3" document, the transferred material shall be any "FTAM-3" document of a type with the same parameter values.

11.2 ISO 8832 (JTM)

The following clauses apply to any JTM implementation claiming support of FTAM-3.

11.2.1 Length and parameters

The implementation shall be capable of handling any "FTAM-3" document type with all values of parameters, subject to a possible implementation limit based on the total number of octets.

NOTE - See ISO 8832 for a specification of "support".

The implementation limit shall permit the use of documents containing up to 64000 octets. The limitations shall be described in the protocol implementation conformance statement.

11.2.2 Concatenation

Concatenation of a document of type "FTAM-3" with itself shall be supported, and shall produce a document of the same type consisting of the combined sequence of binary strings.

NOTE - The boundary of the original documents is only visible in the resulting document as an normal boundary between two binary strings.

11.2.3 Refinement

11.2.3.1 Length refinements

Any document requested by J-GIVE using an "FTAM-3" document type shall be obtained from any stored "FTAM-3" document type which is a relaxation of it under the rules of 11.1.1.1, by processing the document. This processing may produce a document, or may produce a diagnostic. See ISO 8832 for details.

11.2.4 Relaxations

Any document requested by J-GIVE using an "FTAM-3" document type which is a relaxation of the document type under it which it was stored shall be supplied as the requested "FTAM-3" document type.

NOTE - The value in the J-GIVE response is never tightened.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.4 Sequential binary file document type

1 Entry number: FTAM-4

2 Information objects

Table 22 — Information objects in the sequential binary document type

document type	{ iso standard 8571 document-type (5) sequential-binary (4) } "ISO FTAM sequential binary"
abstract syntax names a) name for asname1 b) name for asname2	 {iso standard 8571 abstract-syntax (2) unstructured-binary (4)} "FTAM unstructured binary abstract syntax" {iso standard 8571 abstract-syntax (2) ftam-fadu (2)} "FTAM FADU"
transfer syntax names	 {joint-iso-ccitt asn1 (1) basic-encoding (1)} "Basic encoding of a single ASN.1 type"
Parameter syntax PARAMETERS ::= SEQUENCE {	maximum-string-length [1] IMPLICIT INTEGER OPTIONAL, string-significance [2] IMPLICIT INTEGER {variable(0), fixed(1), not-significant(2)} OPTIONAL }
file model	{iso standard 8571 file-model (3) hierarchical (1)} "FTAM hierarchical file model"
constraint set	{iso standard 8571 constraint-set (4) sequential-flat (2)} "FTAM sequential flat constraint set"
File contents Datatype1 ::=	 OCTET STRING
Datatype2 ::= CHOICE {	Node-Descriptor-Data-Element, Enter-Subtree-Data-Element, Exit-Subtree-Data-Element}

3 Scope and field of application

These document types define the contents of a file for storage, for transfer and access by FTAM, and for transfer by JTM.

4 References

ISO 8571 *Information processing systems - Open Systems Interconnection - File Transfer, Access and Management.*

ISO 8832 *Information processing systems - Open Systems Interconnection - Specification of the protocol for Basic Class Job Transfer and Manipulation.*

5 Definitions

This definition makes use of the terms data element, data unit and file access data unit as defined in ISO 8571-1.

5.1 binary string: an ordered series of zero, one or more binary octets.

6 Abbreviations

FTAM	File Transfer, Access, and Management
JTM	Job Transfer and Manipulation

7 Document semantics

The document consists of zero, one or more file access data units, each of which consists of zero, one or more binary strings. The order of each of these elements is significant. The semantics of the binary strings is not specified by this document type.

The document structure takes any of the forms allowed by the FTAM hierarchical file model as constrained by the sequential flat constraint set (see table 22). These definitions appear in ISO 8571-2.

Each binary string consists of octets of any value from 0 to 255. The meaning attached to these values is not constrained by this document type.

There are no size or length limitations imposed by this definition, except those specified here. Each binary string is of a length determined by the number of octets given by the "maximum-string-length" parameter. If the "maximum-string-length" parameter is not present, the default is that the binary strings are unbounded.

The exact significance of the binary strings is determined by the "string-significance" parameter. If its value is "variable", the length of the binary strings is less than or equal to the length given. If the value is "fixed", the length of each binary string is exactly equal to the length given. If the value is "not-significant", or the parameter is not present, the boundaries of the binary strings are not necessarily preserved when the file is stored and do not contribute to the document semantics.

8 Abstract syntactic structure

The abstract syntactic structure of the document is a hierarchically structured file as defined in the ASN.1 modules ISO8571-FADU and ISO8571-CONTENTS in ISO 8571, in which each of the file contents data elements has the abstract syntactic structure of a "ISO FTAM unstructured binary" document defined in document type registration entry "FTAM-3" in ISO 8571.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9 Definition of transfer

9.1 Datatype definition

The presentation data values used for transfer are either

a) Datatype1 defined in table 22; or

b) Datatype2 defined in table 22, the ASN.1 datatype declared as "Data-Element" in the ASN.1 module ISO8571-FADU.

9.2 Presentation data values

The document is transferred as a series of presentation data values, each of which is either

a) one value of the ASN.1 datatype "Datatype1", carrying one of the binary strings from the document. All values are transmitted in the same (but any) presentation context established to support the abstract syntax name "asname1" declared in table 22.

b) one value of the ASN.1 datatype "Datatype2", All values are transmitted in the same (but any) presentation context established to support the abstract syntax name "asname2" declared in table 22.

NOTES

1 Specific carrier standards may impose additional constraints on the presentation context to be used, where the above permits a choice.

2 Any document type defined in this entry either makes no use of Datatype2, or starts with a Datatype2 transmission.

Boundaries between P-DATA primitives are chosen locally by the sending entity at the time of transmission, and carry no semantics of the document type. Receivers which support this document type shall accept a document with any of the permitted transfer options.

9.3 Sequence of presentation data values

The sequence of presentation data values of type (a) and the sequence of presentation data values of types (a) and (b) is the same as the sequence of binary strings within a Data Unit, and Data Units in the hierarchical structure, when flattened according to the definition of the hierarchical file model in ISO 8571-2.

10 Transfer syntax

An implementation supporting these document types shall support the transfer syntax generation rules named in table 22 for all presentation data values transferred.

Implementations may optionally support other transfer syntaxes.

11 ASE specific specifications

11.1 ISO 8571 (FTAM)

11.1.1 Simplification and relaxation

11.1.1.1 Structural simplification

This simplification loses information.

The document type FTAM-4 may be simplified to the document type FTAM-3. The resultant document contains the same sequence of data values as would result from accessing the structured binary file in access context UA. That is, only the presentation data values in the abstract syntax "asname1" are present.

NOTE - The boundary between file access data units remains a boundary between strings, but any special significance given to it is lost.

11.1.1.2 String length relaxation

This operation loses explicit information in the document type identification.

A document of type "FTAM-4" may be relaxed to another document of type "FTAM-4" with a larger "maximum-string-length" parameter, or no "maximum-string-length" parameter (unbounded strings).

11.1.2 Access context selection

A document of type "FTAM-4" may be accessed in any one of the access contexts defined in the sequential flat constraint set. The presentation data units transferred in each case are those derived from the structuring elements defined for that access context in ISO 8571-2.

In access contexts FA and HA the resultant document conforms to type "FTAM-4". In access context UA the resultant document type conforms to "FTAM-3".

11.1.3 The INSERT operation

When the INSERT operation is applied at end of file, the transferred material shall be the series of FADUs which would be generated by reading any "FTAM-4" document type with the same parameter values in access context FA.

11.2 ISO 8832 (JTM)

The following clauses apply to any JTM implementation claiming support of FTAM-4.

11.2.1 Support of other document types

Document type "FTAM-3" shall also be supported.

11.2.2 Length and parameters

The implementation shall be capable of handling any "FTAM-4" document type with all values of parameters, subject to a possible implementation limit based on the total number of octets in the document.

NOTE - See ISO 8832 for a specification of "support".

The implementation limit shall permit the use of documents containing up to 64000 octets. The limitations shall be described in the protocol implementation conformance statement.

11.2.3 Concatenation

Concatenation of a document of type "FTAM-4" with itself shall be supported, and shall produce a document of the same type consisting of the combined sequence of data units.

NOTE - The boundary of the original documents is only visible in the resulting document as an normal boundary between two data units.

11.2.4 Refinement

11.2.4.1 Length refinements

Any document requested by J-GIVE using an "FTAM-4" document type shall be obtained from any stored "FTAM-4" document type which is a relaxation of it under the rules of 11.1.1.2, by processing the document. This processing may produce a document, or may produce a diagnostic. See ISO 8832 for details.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.2.4.2 Structural refinement

Any document requested by J-GIVE using an "FTAM-4" document type shall be obtained from any stored "FTAM-3" document type by treating the entire "FTAM-3" document as a single data unit (the only data unit) in the "FTAM-4" document.

11.2.5 Structural simplification

Any document requested by J-GIVE using an "FTAM-3" document type shall be obtained from any "FTAM-4" document type by applying the structural simplification specified in 11.1.1.1.

11.2.6 Relaxations

Any document requested by J-GIVE using an "FTAM-4" document type which is a relaxation of the document type under it which it was stored shall be supplied as the requested "FTAM-4" document type.

NOTE - The value in the J-GIVE response is never tightened.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.5 Simple Hierarchical file document type

1 Entry number: FTAM-5

2 Information objects

Table 23 — Information objects in the hierarchical document type

document type name	{ iso standard 8571 document-type (5) simple-hierarchy (5) } "ISO FTAM simple hierarchical file"
abstract syntax names	specify on reference to document type
transfer syntax names	specify on reference to document type
Parameter syntax	specify on reference to document type
file model	{iso standard 8571 file-model (3) hierarchical (1)} "FTAM hierarchical file model"
constraint set	specify on reference to document type
File contents	specify on reference to document type

3 Scope and field of application

This document type defines the structure of the contents of a simple hierarchical file with node identifiers which are strings, for storage and for transfer in FTAM. It is referenced by document type definitions which specify the abstract syntax of the data elements within the structure, in order to give a complete definition of the file.

NOTE - This is not a valid value for Contents Type, and may only be referenced indirectly.

4 References

ISO 8571, *Information processing systems - Open Systems Interconnection - File Transfer, Access and Management*

5 Definitions

This definition makes use of the terms data element, data unit and file access data unit as defined in ISO 8571-1.

6 Abbreviations

FTAM File Transfer, Access, and Management

7 Document semantics

This document consists of a hierarchical file in which the file access data units are named by unique string identifiers.

8 Abstract syntactic structure

The abstract syntactic structure of this document is a hierarchically structured file as defined in the ASN.1 module ISO8571-FADU in ISO 8571, in which each of the file contents data elements may have any abstract syntax, specified when the document type is referenced, and the data element Node-Name in ISO8571-FADU takes the option "ftam-coded".

9 Definition of transfer

The definition of the transfer procedures shall be specified in any document type which references FTAM-5.

10 Transfer syntax

The transfer syntax shall be specified in any document type which references FTAM-5.

11 ASE specific specifications

Any ASE specific specifications shall be specified in any document type which references FTAM-5.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Annex C
Reading of structured files

(This annex does not form part of the standard.)

C.1 Introduction

This annex contains examples of the content and sequence of items occurring in various file structures. The effect of reading such files in different access contexts is shown in a series of figures. For each example, the file structure is shown, together with the order of the structural items in the traversal sequence and the selection of items transferred in the various access contexts. In each example, the root node of the file is assumed to be identified on the read operation.

The definitions of the file model, the constraint sets applied to it and the access contexts in which a file can be read are given in clause 7. Terms defined in clause 7 are abbreviated in the descriptions below as follows

a) "node" stands for a node-descriptor-data-element,

b) "DU" stands for a "data unit", which is a sequence of file-contents-data-elements,

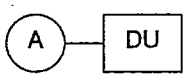
c) "enter subtree" stands for an enter-subtree-data-element,

d) "exit subtree" stands for an exit-subtree-data-element.

A dot in the entry indicates that the data element concerned is transmitted in the access context concerned.

C.2 Unstructured files

An unstructured file consists of a root node with a single associated data unit (see figure 5). This is a special case of the general hierarchical file model expressed by the unstructured constraint set.



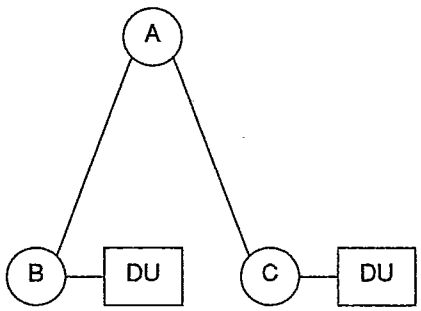
Traversal Sequence	Access Context
	UA
Node (A)	
Data Unit (A)	•

Figure 5 — Unstructured file transfer

C.3 Flat files

A flat file consists of a root node and a series of leaf nodes connected to the root node by arcs of length 1. Every leaf node has an associated data unit (see figure 6). There are three flat constraint sets which may be applied to the general hierarchical model: sequential flat, ordered flat, and ordered flat with unique names.

In access context FA, the root node descriptor "A" is not transferred because it does not have an associated data unit, and hence the data exists indicator is set to false.



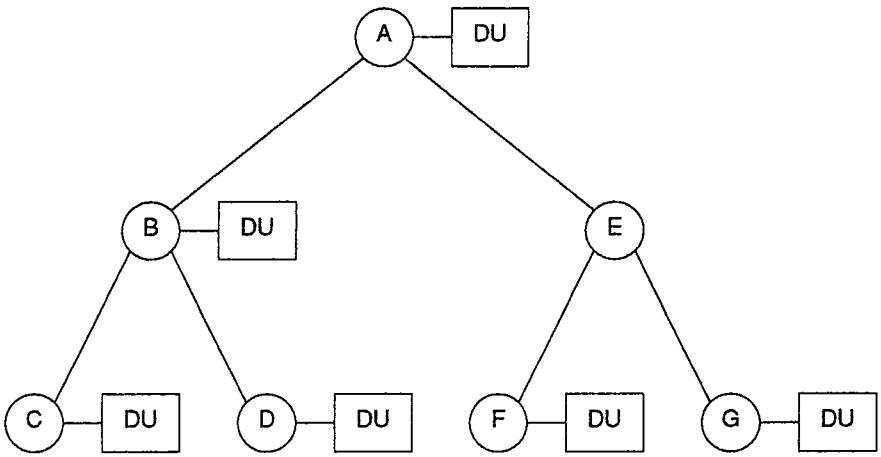
Traversal Sequence	Access Context		
	HA	FA	UA
Node (A)	•		
Enter Subtree	•		
Node (B)	•	•	
Data Unit (B)	•	•	•
Node (C)	•	•	
Data Unit (C)	•	•	•
Exit Subtree	•		

Figure 6 — Flat file transfer

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

C.4 Hierarchical files

There are three constraint sets specifying hierarchical files with different kinds of node identification: ordered hierarchical, general hierarchical, and general hierarchical with unique names. Structurally, they all yield general hierarchical files. An example hierarchical file is shown in figure 7.



Traversal Sequence	Access Context						
	HA	HN	FA	FL (note)	FS	UA	US
Node (A)	•	•	•		•		
DU (A)	•		•		•	•	•
Enter Subtree	•	•					
Node (B)	•	•	•				
DU (B)	•		•			•	
Enter Subtree	•	•					
Node (C)	•	•	•	•			
DU (C)	•		•	•		•	
Node (D)	•	•	•	•			
DU (D)	•		•	•		•	
Exit Subtree	•	•					
Node (E)	•	•					
Enter Subtree	•	•					
Node (F)	•	•	•	•			
DU (F)	•		•	•		•	
Node (G)	•	•	•	•			
DU (G)	•		•	•		•	
Exit Subtree	•	•					
Exit Subtree	•	•					

NOTE - In this example, access in context FL is at level 2.

Figure 7 — Hierarchical file transfer

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex D
Insertion in a structured file

(This annex does not form part of the standard.)

D.1 Basis of examples

This annex presents a series of examples of the modification of a structured file, illustrating the way normal and qualified actions are used to produce different effects on the structure. For simplicity, the actions are described as atomic, but in practice they will be performed item by item.

In the first five examples, the information to be transmitted is derived from the three node file shown in figure 8. The file to be modified is the slightly more complex file shown in figure 9.

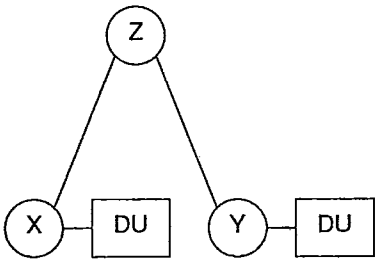


Figure 8 — Source of transferred data

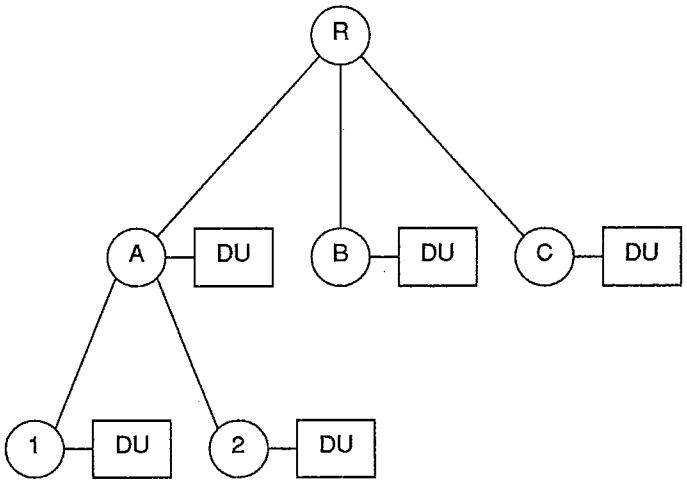


Figure 9 — Initial state of destination file

D.2 Adding sisters to A

The two leaf FADUs X and Y from the source file are to be added as sisters of A, B and C. Assume the FADU identity quoted for the transfer is "A". The data transferred is:

Node(X) DU(X) Node(Y) DU(Y)

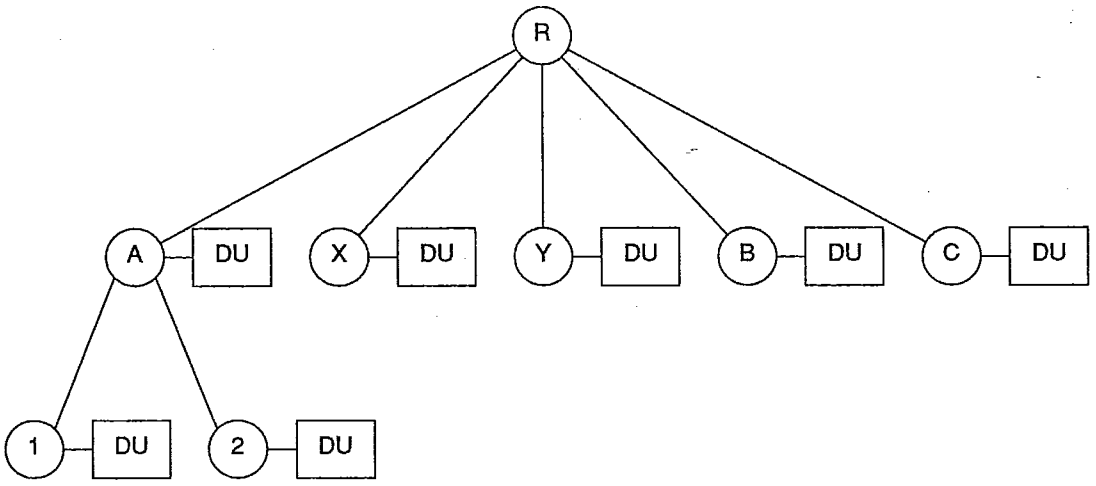


Figure 10 — Final state of insert as sisters to A

At the start of the transfer, the location is set to "A" as a result of the FADU identity quoted. The receiver parses the incoming data stream, recognizing and acting on file structural items as follows:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a) the FADU "X" (Node(X) + DU(X)) is inserted after the FADU "A" in the traversal sequence; "X" becomes the current location.

b) the FADU "Y" (Node(Y) + DU(Y)) is inserted after the FADU "X" in the traversal sequence; "Y" becomes the current location.

The resultant file is shown in figure 10.

D.3 Adding children of C (normal usage)

The two leaf FADUs X and Y from the source file are to be added as children of C. The FADU identity quoted for the transfer is "C". The data transferred is:

Enter Node(X) DU(X) Node(Y) DU(Y) Exit

At the start of the transfer, the location is set to "C" as a result of the FADU identity quoted. The receiver parses the incoming data stream, recognizing and acting on file structural items as follows:

a) a "children" item (Enter + Node(X) + DU(X) + Node(Y) + DU(Y) + Exit) is recognized; the location is saved and a temporary location created before the first child of "C" (although in fact there are no children of "C" at this stage);

b) FADU "X" is inserted after this location. FADU "X" becomes the current location;

c) FADU "Y" is inserted after this location.

d) the end of the sequence of qualified actions is recognized by the presence of the first exit subtree, and the location is restored to the saved value, "C".

The resultant structure is shown in figure 11.

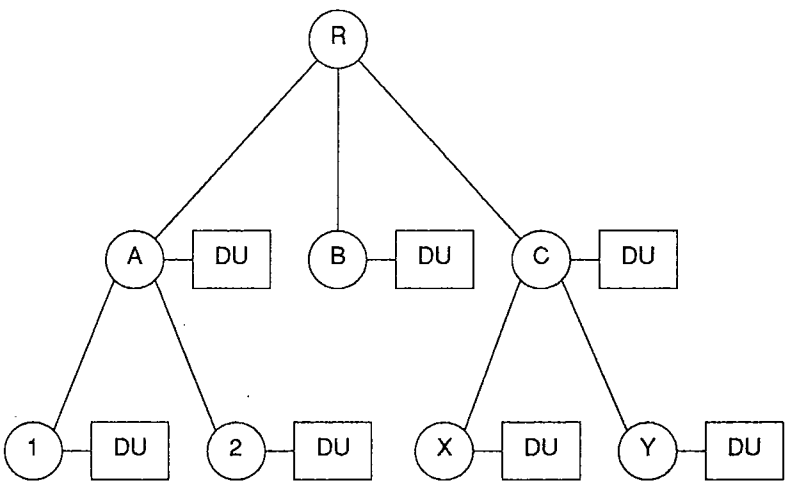


Figure 11 — Final state of insert as children of C (normal)

D.4 Adding children of C (variant)

This illustrates the effect of interrupting the qualified sequence while adding children; the order of insertion is reversed. The practical utility of doing this is not clear, but the example is included to illustrate the insertion procedure. The two leaf FADUs X and Y from the source file are to be added as children of C. The FADU identity quoted for the transfer is "C". The data transferred is:

Enter Node(X) DU(X) Exit Enter Node(Y) DU(Y) Exit

At the start of the transfer, the location is set to "C" as a result of the FADU identity quoted. The receiver parses the incoming data stream, recognizing and acting on file structural items as follows:

a) a "children" item (Enter + Node(X) + DU(X) + Exit) is recognized; the location is saved and a temporary location created before the first child of "C" (although in fact there are no children of "C" at this stage); FADU "X" is inserted after this location. Briefly, FADU "X" becomes the current location, but once the end of the sequence of qualified actions is recognized by the presence of the first exit subtree, the location is restored to the save value, "C".

b) a second "children" item (Enter + Node(Y) + DU(Y) + Exit) is recognized; the location is again saved and a temporary location created before the first child of "C" (this time before "X"); FADU "Y" is inserted after this location. Again the end of the sequence of qualified actions is recognized by the presence of the exit subtree, and the location is restored to the saved value, "C".

The resultant structure is shown in figure 12.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

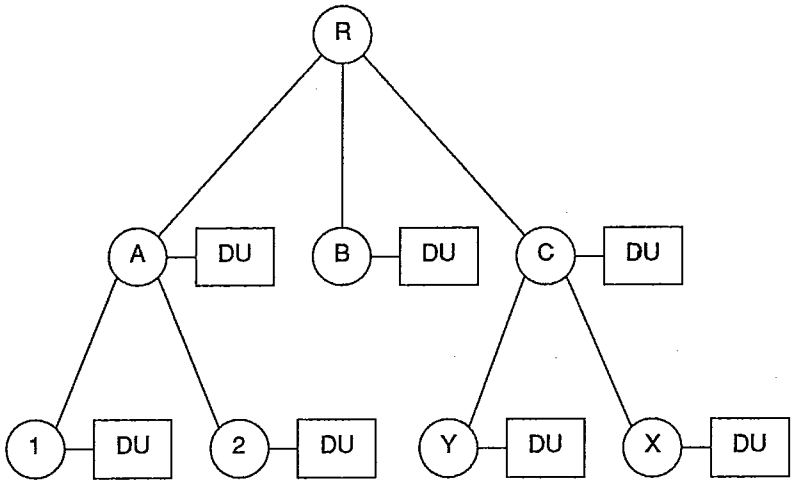


Figure 12 — Final state of insert as children of C (variant)

D.5 Insert subtree as sister

The whole source subtree is to be added as a sister of nodes A, B and C. Assume the FADU identity quoted for the transfer is "C". The data transferred is

Node(Z) Enter Node(X) DU(X) Node(Y) DU(Y) Exit

At the start of the transfer, the location is set to "C" as a result of the FADU identity quoted. The receiver parses the incoming data stream and recognizes the entire data stream as encoding a subtree rooted at Z. It therefore inserts the FADU represented by this subtree into the file as a sister to A, B and C, following C in the traversal sequence. "Z" becomes the current location.

The resultant file is shown in figure 13.

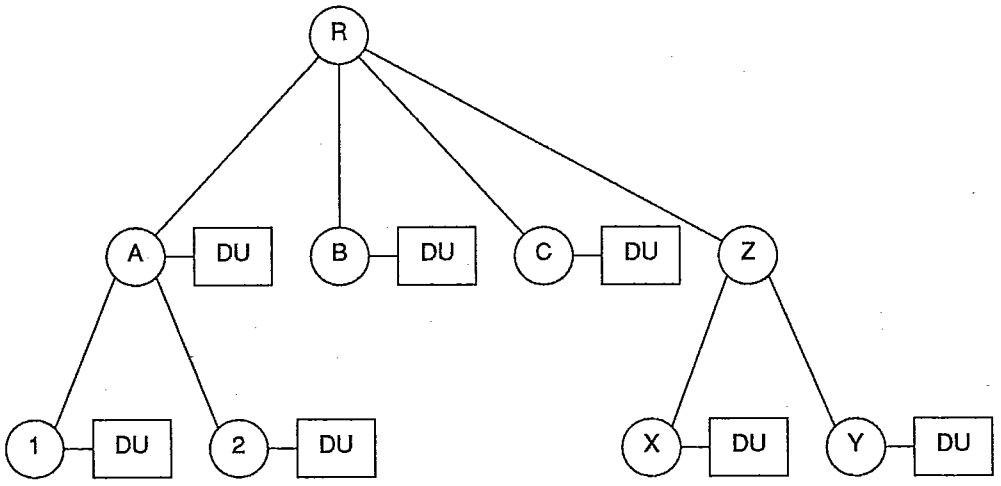


Figure 13 — Final state of insert subtree as sister

D.6 Insert subtree as child

The whole source subtree is to be added as a child of node C. The FADU identity quoted for the transfer is "C". The data transferred is

Enter Node(Z) Enter Node(X) DU(X) Node(Y) DU(Y) Exit Exit

At the start of the transfer, the location is set to "C" as a result of the FADU identity quoted. The receiver parses the incoming data stream and recognizes the entire data stream as encoding a "children"item containing only a subtree rooted at Z. The location is saved and a temporary location created before the first child of "C" (although in fact there are no children of "C" at this stage); the subtree is inserted after this location. Briefly, the root of the subtree "Z" becomes the current location, but once the end of the sequence of qualified actions is recognized by the presence of the first exit subtree, the location is restored to the saved value, "C".

The resultant state of the file is shown in figure 14.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

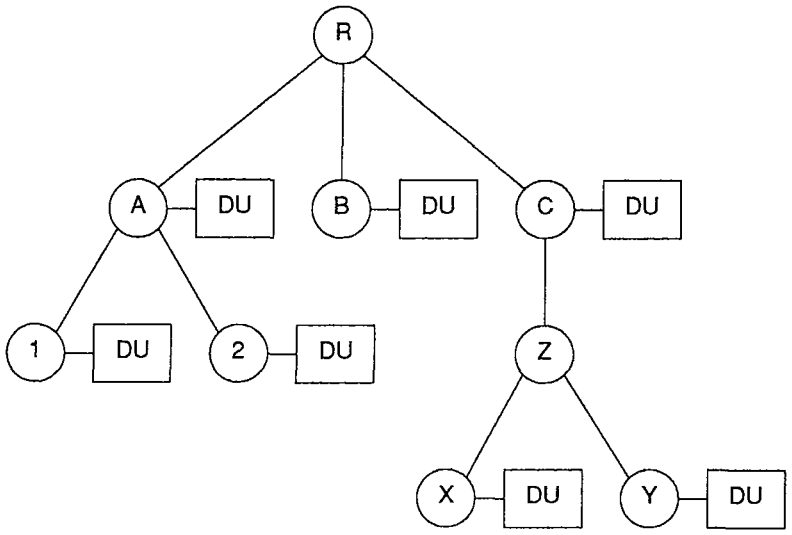


Figure 14 — Final state of insert subtree as child

D.7 Insert in an ordered flat file

The final example illustrates the merging of named leaf nodes into an ordered flat file. In this example, the destination file is assumed to have its leaf nodes identifiers ordered alphabetically. The initial state of the file is shown in figure 15.

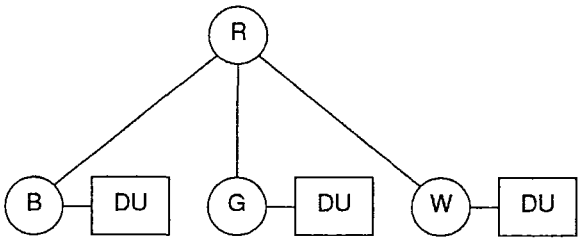


Figure 15 — Initial state of ordered flat file

Nodes with identifiers "A", "P" and "X" are to be merged into the file. The data transferred is

Node(X) DU(X) Node(A) DU(A) Node(P) DU(P)

At the start of the transfer, the location is set to the root node as a result of the FADU Identity quoted. The receiver parses the incoming data stream, recognizing and acting on file structural items as follows:

a) the FADU "X" (Node(X) +DU(X)) is inserted in the collating sequence after the existing FADU "W".

b) the FADU "A" (Node(A) +DU(A)) is inserted in the collating sequence before the existing FADU "B".

c) the FADU "P" (Node(P) +DU(P)) is inserted in the collating sequence after the existing FADU "G".

The resulting file is shown in figure 16.

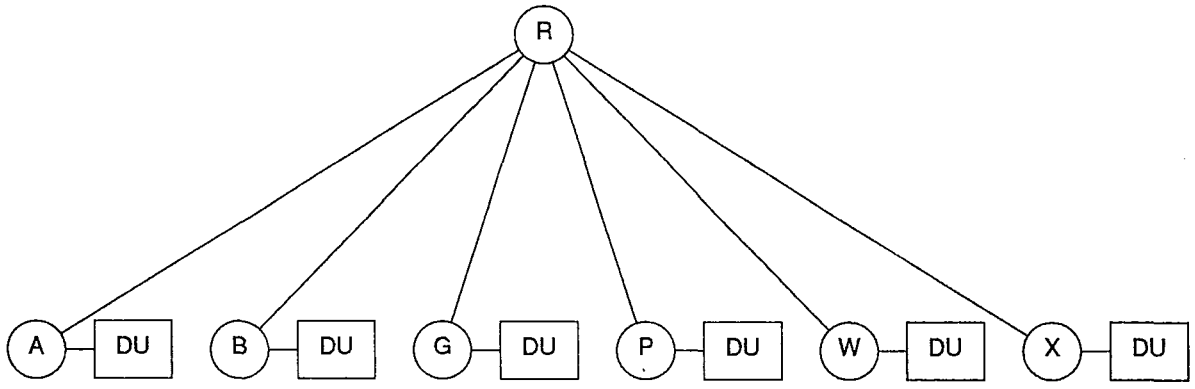


Figure 16 — Final state of merge of ordered flat files

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex E
ASN.1 cross reference

(This annex does not form part of the standard.)

E.1 Cross reference of types defined/referenced in module ISO8571-CONTENTS

```
AnyType . . . . . referenced from line: 5

File-Contents-Data-Element . . . . . defined in line: 5
                                     never referenced
```

E.2 Cross reference of types defined/referenced in module ISO8571-FADU

```
BooleanType . . . . . referenced from line: 27 ([2] IMPLICIT)

Children . . . . . defined in line: 12
                   referenced from line: 9 ([1] IMPLICIT)

ChoiceType . . . . . referenced from line: 31
                                     line: 52

DU . . . . . defined in line: 19
                   referenced from line: 7 ([0] IMPLICIT)

Enter-Subtree-Data-Element . . . . . defined in line: 44
                   referenced from line: 13
                                     line: 54

Exit-Subtree-Data-Element . . . . . defined in line: 46
                   referenced from line: 17
                                     line: 55

ExternalType . . . . . referenced from line: 40

FADU . . . . . defined in line: 50
                                     never referenced

GraphicStringType . . . . . referenced from line: 32 ([0] IMPLICIT)

IntegerType . . . . . referenced from line: 24 ([1] IMPLICIT)

Node-Descriptor-Data-Element . . . . . defined in line: 21
                   referenced from line: 6
                                     line: 53

Node-Name . . . . . defined in line: 31
                   referenced from line: 22

NullType . . . . . referenced from line: 44 ([APPLICATION 1] IMPLICIT)
                                     line: 46 ([APPLICATION 2] IMPLICIT)

SequenceOfType . . . . . referenced from line: 14
                                     line: 19

SequenceType . . . . . referenced from line: 5
                                     line: 12
                                     line: 21 ([APPLICATION 0] IMPLICIT)

Structuring-Data-Element . . . . . defined in line: 52
                                     never referenced

Subtree . . . . . defined in line: 5
                   referenced from line: 14
                                     line: 50
```

E.2.1 References to types defined in module ISO8571-CONTENTS

```
File-Contents-Data-Element . . . . referenced from line: 19
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ISO 8571-2 : 1988 (E)

UDC 681.3 : 621.39

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

Price based on 47 pages

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale