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)

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

Printed in Switzerland

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).

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.

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.

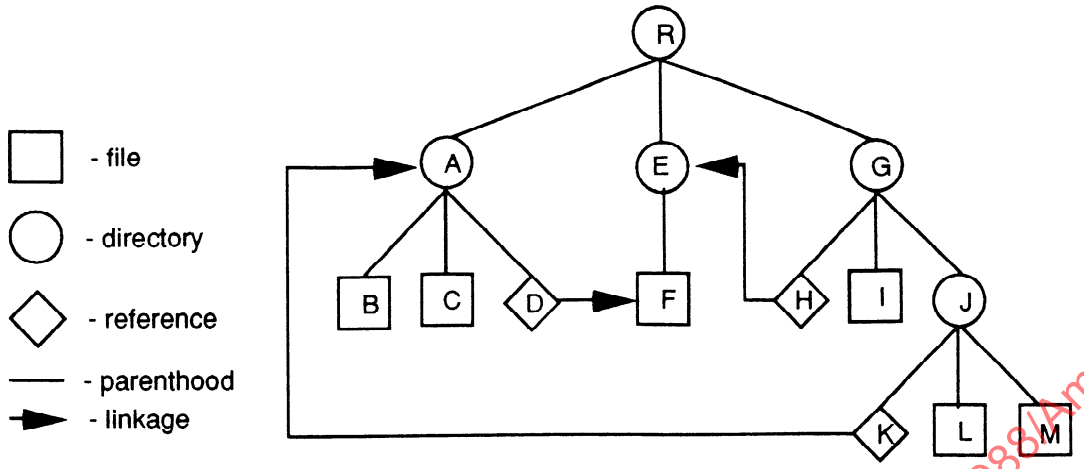

Figure 1a – An example tree structure of a VFS

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.

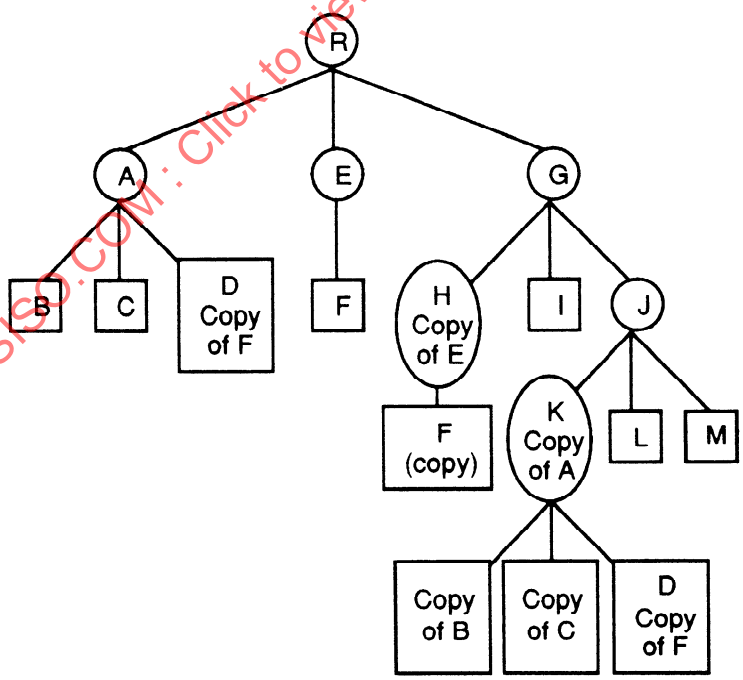

Figure 1b – An example of the apparent structure of a VFS

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.

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.

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.

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

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.

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

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.

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.

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

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.

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.

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

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

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.

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

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.