Stable Implementation Agreements for Open Systems Interconnection Protocols: Part 9 - FTAM Phase 2

Output from the December 1993 Open Systems Environment Implementors' Workshop (OIW)

SIG Chair: **Joe Mohen, Proginet**
SIG Editor: **Larry Friedman, Digital Equipment Corporation**

Foreword

This part of the Stable Implementation Agreements was prepared by the File Transfer, Access and Management Special Interest Group (FTAM SIG) of the Open Systems Environment Implementors' Workshop (OIW). See Part 1 - Workshop Policies and Procedures of the "Draft Working Implementation Agreements Document" for the charter.

Text in this part has been approved by the Plenary of the above-mentioned Workshop. This part replaces the previously existing chapter on this subject. There is no significant technical change from this text as previously given.

Future changes and additions to this version of these Implementor Agreements will be published as change pages. Deleted and replaced text will be shown as struck. New and replacement text will be shown as shaded.

Table of Contents

List of Tables

List of Figures

Part 9 - File Transfer, Access and Management Phase 2

NOTE - In document type names, constraint set names, and abstract syntax definitions, the "NBS" designation will be preserved.

0 Introduction

This part defines Implementors' Agreements based on ISO File Transfer, Access and Management (FTAM), as defined in ISO 8571. This International Standard has four parts. Part 1 of the IS gives general concepts, part 2 defines the Virtual Filestore (VFS), part 3 defines the File Service, and part 4 defines the File Protocol.

FTAM, as described in the IS, is based on the following ISO documents: ACSE Service and Protocol (ISO 8649, ISO 8650), Presentation Service and Protocol (ISO 8822, ISO 8823), ASN.1 Abstract Syntax Notation and Basic Encoding Rules (ISO 8824, ISO 8825), and Session Service and Protocol (ISO 8326, ISO 8327). These services and protocols are defined architecturally in the OSI Reference Model (ISO 7498). These Agreements provide detailed guidance for the implementor, and eliminate ambiguities in interpretations.

The general agreements reached with respect to the ISO File Transfer, Access and Management Protocol (FTAM) are that the Phase 2 FTAM specification (this part) is based on the International Standard (IS).

1 Scope

These FTAM Phase 2 Agreements cover transfer of and access to files between the Filestores of two end systems, including the management of a Virtual Filestore. One end system acts in the Initiator role and initiates the file transfer/access, while the other end system acts in the Responder role and provides access to the file in the Virtual Filestore. This part describes Agreements for the actions and attributes of the Virtual Filestore, and the service provided by the file service provider to file service users, together with the necessary communications between the Initiator and Responder.

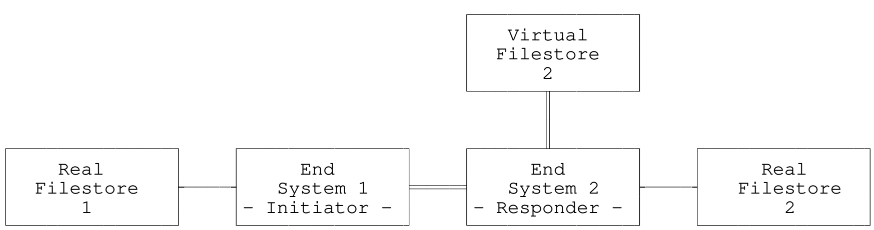

Figure 1 - Model of file transfer/access.

NOTE - Agreements apply on the double lines of figure 1. The mapping between the Virtual Filestore and the Real Filestore together with the local data management system is not part of these Agreements.

These Agreements define general Agreements in clauses 5 through 16, minimum functionality (Conformant Implementations) in clause 17, and functionality for several Implementation Profiles which are tailored to different classes of user requirements in clauses 18 and 19.

2 Normative References

ISO 8571-1: 1988(E), Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management Part 1: General Introduction

8571-2: 1988(E), Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management Part 2: Virtual Filestore Definition ISO

ISO 8571-3: 1988(E), Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management Part 3: The File Service Definition

ISO 8571-4: 1988(E), Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management Part 4: The File Protocol Specification

3 Status

Version 3 of the FTAM Phase 2 Implementation Agreements was completed December 15, 1989, and published in version 3 of the Stable Implementation Agreements (NIST Special Publication 500-177, December 1989). Some editorial clarifications and technical changes (due to international harmonization of Profiles) were added to Version 3 Agreements in the course of 1990. All these changes are now fully incorporated in this version 4 of the FTAM Phase 2 Stable Agreements.

No further enhancements will be made to this version 4 of FTAM Phase 2 (see the clause 4 ERRATA).

This version 4 of the FTAM Phase 2 Agreements fully replaces the version 3 as published in NIST SP 500-177. Therefore, the old version 3 of FTAM Phase 2 will no longer be maintained.

The following list summarizes the technical errata changes to FTAM Phase 2, Version 1 which were incorporated in Version 2. It also states the degree of possible interworking and backward compatibility to FTAM Phase 2, Version 1.

Technical Changes in FTAM Phase 2, Version 2 (Dec. '88)	**Backward Compatibility to FTAM Phase 2, Version 1 (Dec. '87)**
1. Check of already established presentation contexts for a document type not at CREATE time but at OPEN time	Interworking problems could occur when creating a document type which is not opened
2. Receivers shall not require sending of AETtitles	Interworking problems could occur if AETitles are not sent
3. Minimum requirement for FADU identities for document types which use the sequential flat constraint set	Interworking problems could occur if FADU identities beyond the minimum requirement are used

The following list summarizes the technical errata and alignment changes which were incorporated in FTAM Phase 2, Version 3. It also states the degree of possible interworking and backward compatibility to FTAM Phase 2, Version 2.

Technical Changes in FTAM Phase 2, Version 3 (Dec. ’89)	**Backward Compatibility to FTAM Phase 2, Version 2 (Dec. ’88)**
1. Unconstrained Service Class outside the scope of the Implementation Profiles	Full compatibility, since change only from "optional" "to "outside scope"
2. <contents-type-list> parameter: Both types of values optional for Initiators	Full compatibility, since this clarification is only for Initiators
3. Specification of supported values for <override> parameter in F-CREATE	Interworking problems may occur, if different values supported
4. Parameters <filesize>, <future filesize>, <fadu-number> may be encoded with up to 8 contents octets	Interworking problems may occur, since no minimum requirement was defined for Version 2.
5. For NBS-7 and NBS-8 in conjunction with T or TM Service Class, the FADU identities "current", "next", "previous" are not required	Full compatibility, since these identities were never useful in conjunction with T or TM Service Class
6. For NBS-8 files the minimum range for keys of string-type is (1-16) instead of (0-16)	Interworking problems may occur for key-length parameter 0
7. Restriction "NBS-9 files may not be Created or Deleted" removed from the document type definition. But both actions are outside the scope of the Agreements.	Full compatibility, since Creation, Deletion was never in the scope of the Agreements
8. Constraint set NBS-ordered-flat: The location after an Insert is "end"	Full compatibility, since this specification was already implicitly clear

4 Errata

NO. OF ERRATA	TYPE	REFERENCED DOCUMENT	CLAUSE	
CP 3/91-1	EDITORIAL	NIST-SP 500-183	All	Update to ISO style. General formatting and error corrections. Alignment with the wording of the ISP. Consistent naming conventions.
CP 9/91-1 CP 9/91-2 CP 9/91-3 CP 9/91-4 CP 9/91-5 CP 9/91-6	EDITORIAL	NIST-SP 500-183	Table 9 Table 10 Table 11 Clause 17 Annex A A.5.11.1 A.6.11.1 A.7.7 A.7.9.1 A.7.9.2	Datatype1 ASN.1 comment to use the object descriptor Include Corrigenda Editors note to point at ISP for registered objects. Change header to Structural Simplification. Added definitions for Datatype3

5 Assumptions

FTAM protocol machines must be able to parse and process at a minimum 7K octets of FTAM PCI, FTAM structuring (FTAM-FADU) and FTAM user data (including grouped FPDUs) as they would be encoded with the ASN.1 Basic Encoding Rules. It is recommended, however, that Presentation user data not be restricted in size.

In order to maximize interoperability, it is important that the implementations of FTAM service providers do not unnecessarily restrict the service user's ability to generate arbitrary file service requests. Otherwise, they may not be able to work with FTAM Responders whose operation is constrained by their mapping of the FTAM Virtual Filestore to their local filestore. For example, error procedures should only be invoked when an error actually occurs, not at the point of the specification of options which might result in an error.

Implementations must be able to parse all valid optional parameters if they are present in the PDU. Only those optional parameters specified as supported in these Agreements are required to be implemented. If these parameters are not present, a default value is assigned locally. A Responder should not refuse a request solely because a parameter that is optional in the FTAM standard, but is supported in these

Agreements, is not present.

Consideration of any standardized service interface is not covered by these Agreements.

These Agreements define no restrictions for the values used for the <communication quality of service> parameter in <F-INITIALIZE>.

FTAM is defined in phases. The Phase 1 FTAM implementation specification is based on the second ISO Draft Proposal, dated April 1985, and the ISO Draft Proposal 8824 and 8825.

The Phase 2 FTAM specification (this part) is based on the International Standard (IS). THERE IS NO BACKWARD COMPATIBILITY WITH FTAM PHASE 1. Backward compatibility is impossible, since Phase 1 uses Session services directly, while Phase 2 uses ACSE and Presentation services. Furthermore, there are differences in Filestore, PDU Abstract Syntax, FADU Abstract Syntax, and Transfer Syntax. There also are differences in the transparency mechanisms and service class negotiations.

The <implementation information> parameter of <F-INITIALIZE> FPDU as defined in ISO 8571-4, 20.3 is used to pass "user version" information with respect to different FTAM phases of the OSI Implementors' Agreements or with respect to FTAM profiles of other bodies (see clause 12). It is the goal of these Agreements to use the "user version" mechanism to provide at least one level of backward compatibility for all future FTAM Phases, facilitating backward compatibility for future FTAM products, assuming different new versions of the respective IS's also enable backward compatibility.

The FTAM specific ASE requirements for ACSE in the Upper Layers part of this document define a value (that carries no semantics) for the AETitle that can be used by FTAM ASEs for communication. Other values for the AETitle are outside the scope of these Agreements.

The association shall not be rejected/aborted if any of the following parameters either contains the defined value or is not sent:

Called - AETitle

Calling - AETitle

Responding - AETitle

The association may be rejected/aborted if any of these parameters contain other values.

Use of values outside the scope of these Agreements is discouraged until agreed upon semantics have been associated with AETitles.

Use of <shared ASE information> parameter and <charging> parameter is not defined within the scope of the Agreements.

Use of <application context name> parameter is not defined within the scope of these Agreements. This parameter does not prohibit the establishment of an FTAM association.

These Agreement use the term "supported" for a parameter to mean that the syntax and semantics of that parameter shall be implemented. However, it is not a requirement that the parameter be used in all instances of communication, unless stated otherwise.

Also these Agreements use the term "optionally supported" for a parameter to mean that it is left to the implementation whether the semantics of that parameter are implemented or not.

6 Presentation agreements

The following Abstract Syntaxes are recognized in these Agreements:

"FTAM FADU"

"FTAM PCI"

"FTAM unstructured text abstract syntax"

"FTAM unstructured binary abstract syntax"

"NBS abstract syntax AS1"

"NBS file directory entry abstract syntax"

The following Transfer Syntax is supported:

"Basic Encoding of a single ASN.1 type"

See also annex C.

7 Service class agreements

Implementation Agreements have been reached for the following service classes:

File Transfer

File Access

File Management

File Transfer and Management

Unconstrained

8 Functional unit agreements

Implementation Agreements have been reached for the following functional units:

Kernel

Read

Write

File Access

Limited File Management

Enhanced File Management

Grouping

Implementation of the Recovery, Restart Data Transfer, and FADU Locking functional units is not specified.

9 File attribute agreements

Implementation of the Kernel Group of file attributes is defined. If the optional Storage Group and Security Group are implemented, aspects of their implementation are defined. Implementation of the Private Group is not specified.

Responses to an attribute value request shall always include one of the following (as specified in ISO 8571-2, clause 4):

An actual file attribute value.

A value indicating that no value is available, optionally with a diagnostic.

No value and an error code, optionally with a diagnostic indicating that the attribute is not supported.

9.1 Mandatory group

Only the Kernel Group of attributes is required. A value for <filename>, <permitted actions>, and <contents type> will always be available.

A minimum range is required for <filename> values as specified in ISO 8571-2. No maximum length or format restrictions apply. A system that does not support <filename> values with a sequence of more than one Graphic String or extended <filename> characteristics may reject a request involving such a <filename>. All systems must be able to interpret a <filename> value with a sequence of one Graphic String.

A Responder shall not require an Initiator to use multiple component Graphic String filenames. Requests using a single component <filename> value with a sequence of one Graphic String shall be responded to using a single component <filename> value. Responses to requests involving <filename> values having two or more Graphic Strings are not defined here but may be interpreted via bilateral or other external agreements. Use of <filename> values with a sequence of more than one Graphic String is discouraged.

Apart from the minimum conformance requirements specified in ISO 8571-2, file names have to be specified in the naming convention of the responding FTAM implementation. It is a local implementation matter of the FTAM Responder, whether or not an additional name mapping onto the real FIlestore's file name convention is supported.

In order to enable interworking with all FTAM Responders' virtual Filestores, it is recommended that FTAM Initiators impose no restrictions on the attribute range supported for file names beyond those specified in ISO 8571-2.

For the purpose of interworking according to these Agreements the <contents type> attribute is limited to the <document type name> format. The <constraint set name, abstract syntax name> form is outside the scope of these Agreements. It should always be parsed correctly when received, but may result in an error.

9.2 Optional groups

If the optional Security Group of file attributes is implemented, an actual value must be available for the <access control> attribute.

The <access control> attribute is a SET OF <access control element>. The minimum requirement in these Agreements is the support of one <access control element>, according to the base standard. The terms <concurrency access>, <identity>, and <passwords> are each optionally supported. Details of their use shall be specified in the PICS. Use of the <location> term is not specified in these Agreements.

Implementation of the Private Group is not specified.

10 Document type agreements

These document types are defined:

FTAM-1 "ISO FTAM unstructured text"

FTAM-2 "ISO FTAM sequential text"

FTAM-3 "ISO FTAM unstructured binary"

NBS-6 "NBS-6 FTAM sequential file"

NBS-7 "NBS-7 FTAM random access file"

NBS-8 "NBS-8 FTAM indexed file"

NBS-9 "NBS-9 FTAM file directory file"

Detailed document type definitions are given in annex A and in ISO 8571-2, annex B.

NOTE - Document types NBS-1 to NBS-5 are not defined in these Agreements. The numbering starts with NBS-6 because of the original DIS version of these Agreements.

An implementation claiming conformance to these Agreements which also supports any or all of the document types FTAM-1, FTAM-2, and FTAM-3 as defined in ISO 8571-2, annex B, must minimally support the combinations of parameter values as specified in table 1

Table 1 - Parameters for FTAM-1, -2, -3

	Universal Class Number	**Maximum String Length**[6,7]	**String Significance**
FTAM-1	General String [1](27) IA5String [2](22)	134 or less	‘not-significant’[8]
FTAM-2	Graphic String [3,4](25)	134 or less[5]	‘not-significant’[8]
FTAM-3	<not applicable>	512 or less	‘not-significant’[4]

NOTES

1 The minimum level of support for General String is the ISO 646, IRV G0 character set and the 8859-1 G0 and G1 character sets, and ISO 646, IRV C0 set.

2 The support for IA5 String is the ISO 646, IRV G0 character set and the ISO 646, IRV C0 set.

3 The minimum level of support for Graphic String is the ISO 646, IRV G0 character set and the 8859-1 G0 and G1 sets.

4 This is the default when the parameter is not specified.

5 The implementation need not support Data Units whose total character count exceeds 134.

6 As per table 3.

7 The length refers to the number of characters from the applicable character set. It does not include any escape sequences or overhead from the encoding.

8 If escape sequences are used in the encoding of the data and string boundaries are not maintained when the file is stored, it may be necessary for the escape sequences to occur at different locations when the file is re-sent.

For document types which use the sequential flat constraint set, conformant implementations must minimally support FADU identities as follows:

for Transfer service class: "begin," "end" for Transfer and Management service class: "begin," "end"

for Access service class: "begin," "end," "first," "next"

For the document types NBS-7 and NBS-8 used in conjunction with the Transfer service class or the Transfer and Management service class, the support of the FADU identities of ‘current’, ‘next’ and ‘previous’ and for NBS-8 the support of the FADU identity of ‘end’ are outside the scope of these Agreements.

For the document types NBS-6, NBS-7 and NBS-8 parameters are used for which the Agreements apply as specified in table 2.

Table 2 - Parameters for NBS-6, NBS-7, NBS-8

Parameter	Prim Type	String-length	Length-1	Length-2
int	INTEGER	Number of octets required to represent, in 2’s complement format, the largest integer to be passed		
bit	BIT STRING	Number of bits in string (non-varying)		
ia5	IA5STRING	Max number of characters in string		
graphic	Graphic String	Max number of characters in string		
general	General String	Max number of characters in string		
octet	OCTET STRING	Max number of octets in string		
private- class-number	Floating Point Number		The minimum number of bits required to be maintained in the mantissa for relative precision	Number of bits required to represent the largest unbiased integer exponent in 2’s complement
univer-time	UTCTime	<not applicable>		
gen-time	Generalized Time	<not applicable>		
boolean	BOOLEAN	<not applicable>		
null	NULL	<not applicable>		

NOTE - The string length parameter specifies the actual number of from the referenced character set. It does not include any escape sequences or overhead from the encoding.

The primitive data types and minimal size ranges that an implementation must accept for storage are given in table 3.

Table 3 - FTAM primitive data types

Primitive Data Type	Minimum Range (Octets)
ASN.1 INTEGER	1 - 2
ASN.1 BIT STRING	0 - 1
ASN.1 IA5String	0 - 134
ASN.1 GeneralString	0 - 134
ASN.1 GraphicString	0 - 134
ASN.1 OCTET STRING	0 - 512
ASN.1 BOOLEAN	
ASN.1 NULL	
ASN.1 GeneralizedTime	
ASN.1 UniversalTime	
NBS-AS1 FloatingPointNumber	mantissa 1-23 bits exponent 0-8 bits

NOTES

1 The primitive data types and their maximum ranges for a specific file as described by the parameters above are maintained in the <contents type> file attribute. The <contents type> file attribute value is established at the file's creation and cannot be changed via FTAM for the life of the file. This implies that the data element types and ranges and data unit formats are fixed for all accessors of that file as long as the file exists.

2 The syntax for floating point numbers is part of the definition of NBS abstract syntax AS1 in annex C.3. It is derived from existing standards IEC 559 and IEEE 754.

10.1 Character set

Implementation of a character set in FTAM is understood as:

a transfer syntax is defined for the character set

document types are defined using the character set in their abstract syntactic definition

documents of those types are stored in the Virtual File Store as defined in the character set specification. They are written into the VFS and read from the VFS as defined by the abstract syntax and the transfer syntax for the document type. It is not in the scope of FTAM Agreements to specify the local representation of those documents in the Real Filestore, nor to specify rendition of graphic characters or control characters on character imaging devices. These renditions are possible agreements between applications using FTAM for their communication.

The character sets ISO 646, IRV and ISO 8859-1 shall always be implemented.

10.1.1 ISO 646 Character set

The International Reference Version (IRV) of ISO 646 is available for use when there is no requirement to use a national or an application-oriented version. In information interchange, the IRV is assumed unless a particular agreement exists between sender and receiver of the data. The graphic characters allocated to the IRV are as specified in table 4.

Table 4 - IRV graphic character allocations

Graphic	Name	Coded Representation
#	Number sign	2/3
o	Currency sign	2/4
@	Commercial at	4/0
[	Left square bracket	5/11
\	Reverse solidus	5/12
]	Right square bracket	5/13
^	Circumflex accent	5/14
’	Grave accent	6/0
{	Left curly bracket	7/11
|	Vertical line	7/12
}	Right curly bracket	7/13
~	Tilde, overline	7/14

It should be noted that no substitution is allowed when using the IRV and that the facility of combined vertical and horizontal movements of the active position does not apply to any format effectors.

It is permitted to use composite graphic characters and there is no limit to their number. Because of this freedom, their processing and imaging may cause difficulties at the receiving end. Therefore agreement between sender and receiver of the data is recommended if composite characters are used.

> **NOTE -** Attention is drawn to the fact that different national character sets exist.

(See ISO 646 or CCITT Recommendation T.50 for more information)

10.1.2 Format effectors

When a single format effector for vertical (or horizontal) movement is optionally permitted to effect a combined vertical and horizontal movement, implementations shall not use the single format effector for effecting the combined vertical and horizontal movement. It is recommended that OSI Implementations use <CR><LF> pairs as line terminators. Failure to follow this recommendation may result in an implementation which does not interoperate with other implementations conforming to these agreements.

NOTES

1 For further information see ISO 646:1983, clauses 4.1.22 and 6.4, ISO 6429:1988, clause E.1.2 and ISO 4873:1986, clause A.3.2.

2 The Agreements require only support of CO control characters of ISO 646, containing among others the format effectors <CR> and <LF>.

10.1.3 8859-1 Character set

The Latin Alphabet No.1 (ISO 8859-1) is used to specify the printable characters of G0 and G1. C0 control characters and their associated rules are taken from the ISO 646 definition.

10.2 Document type Negotiation rules

10.2.1 Connection establishment

In connection establishment the <contents type list> parameter is used only to establish presentation contexts. Both the <document type name> form and the <abstract syntax name> form are supported for Responders; they are optionally supported for Initiators.

10.2.2 File creation

An <F-CREATE request> FPDU must contain a <document type name> value in its <initial attributes> parameter.

If the specified document type requires parameterization, then these parameters must be supplied, otherwise the <F-CREATE request> may be rejected.

NOTES

1 It is understood that <permitted actions> sub-field of <initial attributes> parameter will always be used at <F-CREATE request>. The value may be changed by the Responder.

2 If the <document type name> used requires DU syntax parameters and one of the parameters specifies "FloatingPointNumber" as a primitive data type, the request may be rejected, in case the optional type "FloatingPointNumber" is not supported by the Responder.

10.2.3 File opening

The <document type name> form (with appropriate parameters as specified in 8871-2, clause 12.3) shall always be used when proposing a <contents type>; as an alternative the 'ContentsTypeUnknown' value may be used in the <F-OPEN request>. An <F-OPEN response> shall use the <document type name> option (with appropriate parameters) in the <contents type> field.

This allows the receiving entity to use the <document type name> attributed to the file instead of receiving a <constraint set name> and <abstract syntax name> pair, which does not reflect the file information contained in the FTAM and NBS document types.

This document type name is either a value from the set of base document type names as negotiated upon connection establishment or a document type name, for which an appropriate presentation context was established.

NOTES

1 An <F-OPEN response> without a <document type name> (but carrying the <constraint set name> and <abstract syntax name> form) may cause the Initiator to issue an <F-CLOSE request>.

2 If the <document type name> used requires DU syntax parameters and one of the parameters specifies 'FloatingPointNumber' as a primitive data type, the request may be rejected, in case the optional type 'FloatingPointNumber' is not supported by the Responder.

10.3 Relationship between DUs, DEs and document types

"Abstract Syntax" is used to refer to the syntactic information which is architecturally passed between the Application and Presentation Layers. The Abstract Syntax defines Data Element (DE) types which are not necessarily ASN.1 primitive types. A Data Element (DE) is the smallest piece of data whose identity is necessarily preserved by the Presentation Service. Data types may be made up of other data types. Data Elements are not defined in terms of other Data Elements.

A Data Unit (DU) is a sequence of one or more Data Elements. Architecturally, entire, single DEs are passed into and out of the application process. In a real implementation, DUs may be passed.

To maintain DU boundaries during transfer, file structuring information must be passed (ISO8571-FADU definition in ISO 8571-2, clause 7.5). A Data Element is referred to as a File-Contents- Data-Element in the ISO8571-FADU definition.

Document types refer to aspects of local processing and storage. They describe:

structural relationship between DUs,

structure of DUs, called DU syntax, and

DE types found in the file

Because document types pertain to local processing and storage, the DU syntax makes assertions about the syntax and the size of DUs (records) in storage. Parameters on the document types provide this information about the syntax and size of the DUs.

11 F-CANCEL action

When an F-CANCEL is sent or received, the following occurs:

no more data is sent,

checkpoint numbers are removed, and

state of the file is implementation dependent.

NOTE - When mapping F-CANCEL on P-RESYNCHRONIZE (abandon) it is required that P-SYNC-MINOR be used after F-READ/ F-WRITE (see ISO 8571-4 clauses 13, 14).

12 Implementation information agreements

The <implementation information> parameter of <F-INITIALIZE> FPDU is not required by these Agreements.

It may be used to pass user version information as a series of values, separated by ";"

The following will indicate conformance to the Phase 2 Agreements: NBS-Phase2.

NOTE - The list of possible values may be enlarged for future FTAM phases or FTAM profiles of other bodies.

This parameter is for information only; it is not used for negotiation.

The establishment of an FTAM regime should not be rejected only because of an unknown <implementation information> value.

13 Diagnostic agreements

The <diagnostic> parameter is supported; a value in the <response> PDU is needed when the <action result> or <state result> is not zero. (The nature of these agreements is to provide <diagnostic> information when any result parameter is not "success.")

General catch-all diagnostic action is discouraged.

The <further details> subfield is supported. It will be encoded as GraphicString, but is restricted to ISO 646 (IRV, graphic characters) and ISO 8859-1 only.

Use of F-P-ABORT for other than protocol errors and catastrophic situations is discouraged.

When returning an error status in a file management related diagnostic (i.e., <F-READ-ATTRIBUTE response> or <F-CHANGE-ATTRIBUTE response>), identify the erroneous attribute by using the first two characters of <further details> to hold a 2-digit number (encoded as IA5String) from the <F-READ-ATTRIBUTE request> attributes abstract syntax definition (ISO 8571-4, clause 20.3).

00	Filename
01	Permitted Actions
02	Contents Type
03	Storage Account
04	Date and Time of Creation
05	Date and Time of Last Modification
06	Date and Time of Last Read Access
07	Date and Time of Last Attribute Modification
08	Identity of Creator
09	Identity of Last Modifier
10	Identity of Last Reader
11	Identity of Last Attribute Modifier
12	File Availability
13	File Size
14	Future Filesize
15	Access Control
16	Legal Qualifications
17	Private Use

The set of file management diagnostics, found in ISO 8571-3 annex A, must be supported.

In the case where a specific parameter can in no way be accommodated then the request fails and a <diagnostic> indicating one such parameter should be returned by the responder. In the case where a negotiable parameter cannot be accommodated with exactly the value requested but is negotiated to a different value (as defined in the standard) then the request formally succeeds but informative <diagnostics> indicating those parameters negotiated should be returned.

In order to provide for robust applications using FTAM, well defined and precise diagnostics are required to be returned by responding implementations whenever an action cannot be carried out precisely as requested with respect to non-negotiable parameters. All such applicable diagnostics will be returned in those cases. An action is carried out precisely as requested with respect to a parameter when the value of that parameter on the <request> FPDU is equal to the value in effect during or subsequent to the action, depending on whether the action is regime control.

Diagnostics exist to signal "parameter not supported" and Responder implementations shall issue all appropriate diagnostics. The <further details> subfield of the <diagnostic> parameter shall specify the parameter which is not implemented.

14 Concurrency

The <concurrency control> used by default on actions requested by an <F-SELECT indication> or <F-CREATE indication> service are:

"shared" for read and read attribute

"exclusive" for all other actions

The default for actions not requested is specified as "not required" as per ISO 8571-3.

> **NOTE -** A local implementation may choose to be more restrictive in order to assure file consistency for concurrent accessors.

FADU locking is not required.

15 Requested access

The <requested access> parameter on <F-SELECT> or <F-CREATE> is used to specify the actions which the Initiator may perform during the file selection. The value of the <requested access> parameter is compared by the Responder to the <access control> and <permitted actions> file attributes and concurrency controls (including those requested by the Initiator) currently in place on the file. If the value of the <requested access> parameter is not consistent with either <access control>, <permitted actions>, or concurrency controls in place, then the <F-SELECT> or <F-CREATE> must be rejected.

<requested access> is consistent with <access control> if, for each action requested, that action either requires no password, or the required password has been specified on the <F-SELECT request> or <F-CREATE request>.

<requested access> is consistent with <permitted actions> if, for each action requested, that action is allowed by the <permitted actions> file attribute.

<requested access> is consistent with <concurrency control> requested on the <F-SELECT> or <F-CREATE> if, for each action requested, that action has not been specified as "not required" or "no access" in the <concurrency control> parameter.

<requested access> is consistent with concurrency controls in place on the file if for each action requested no other accessor of the file has set the concurrency control for that action to either "exclusive" or "no access."

16 Security

16.1 Initiator identity and filestore password

The <initiator identity> and <filestore password> parameters for an implementation acting as an Initiator are supported. These parameters are optional for an implementation acting as a Responder.

The syntax of <initiator identity> and <filestore password> is system-dependent. <initiator identity> and <filestore password> will represent account information on the local system, which may be different from the <account> parameter.

16.2 Access passwords

The <access passwords> and <create password> parameters for an implementation acting as an Initiator are supported if the Security Group of attributes is supported. These parameters for an implementation acting as a Responder are optionally supported if the Security Group is supported.

16.3 Implementation responsibilities

It is the responsibility of each local system to provide security for its own real filestore. Encryption of passwords will not be done by FTAM.

A user of the file service must be known by the Responder. "Known" is defined by the local Filestore, and is dependent on the level of security provided by the local Filestore.

17 Requirement for conformant implementations

This clause gives the criteria to be satisfied by every implementation of FTAM that conforms to these Agreements.

Conformance to these Agreements is stated in terms of the different roles occupied by FTAM implementations. The interoperability of certain configurations of these roles motivates this approach. Interoperable configurations of these roles are given in 17.1.

The only function provided by every conformant implementation is the transfer of unstructured binary files in their entirety. It must be recognized that such simple transfer, while commonly understood and generally important, will not support all applications of FTAM. Clause 18 defines Implementation Profiles of FTAM services and protocol that can provide other specific functions. Those other functions exploit the access and management capabilities of FTAM. The unconstrained service class (with appropriately chosen functional units) can be used to provide the functions of any of the Implementation Profiles. Users of FTAM must consider carefully what functions they require. They must examine all the Implementation Profiles and select according to their needs.

Implementations conforming to these Agreements require adherence to the General Agreements in clauses 5 through 16 of these Agreements.

Implementations conforming to these agreements shall implement the defect report solutions contained in:

ISO 8571-1/Cor.1:1990
ISO 8571-2/Cor.1:1990
ISO 8571-3/Cor.1:1990
ISO 8571-4/Cor.1:1990

ISO 8571-3/Cor.2:1990
ISO 8571-4/Cor.2:1990

> **Editor's Note -** The corrigenda ISO 8571-3/Cor.2 and ISO 8571-4/Cor.2 is to be published. Until it is available, the solutions can be found in the documents ISO/IEC JTC1/SC21 N 5234 and ISO/IEC JTC1/SC21 N 5235.

17.1 Interoperable configurations

Any implementation conforming to this specification must be able to act in at least one of the following role combinations:

1. initiator and receiver,
2. initiator and sender,
3. responder and sender,
4. responder and receiver.

Minimal implementations of combination 1 will interoperate with minimal implementations of combination 3. Minimal implementations of combination 2 will interoperate with minimal implementations of combination 4.

Any implementations of roles 1 and 3 will be able to interoperate at the intersection of their capabilities (which will be at least the minimal capabilities described in 17.3 to 17.8). Any implementations of roles 2 and 4 will be able to interoperate at the intersection of their capabilities (which will be at least the minimal capabilities described in 17.3 to 17.8).

These role combinations and this interoperability are shown in table 5 below.

Table 5 - Interoperable configurations.

		Initiator		**Responder**	
		sender	receiver	sender	receiver
Initiator	sender				x
	receiver			x	
Responder	sender		x		
	receiver	x			

17.2 Relationship to ISO 8571--The FTAM Standard

Any implementation in conformance to ISO 8571 (as defined in ISO 8571-4, clause 22 (Conformance)), in addition to the implementation of the minimal protocols and roles enumerated in subclauses 17.3 to 17.8, is considered to be in conformance with these Agreements. Any implementation violating any of the conformance statements in ISO 8571-4 is considered to be in violation of these Agreements.

17.3 Requirements for document type support

The document type FTAM-3 shall be supported for purposes of transfer and storage. The details regarding support for FTAM-3 in the FTAM dialogue are given in clause 10.

Support of document types other than FTAM-3 is not required for conformant implementations. Support for document types described in these Agreements also entails support for:

the semantics given in their description and further qualified in clause 10

the preferred transfer syntax "Basic Encoding of a single ASN.1 type"

17.4 Initiators

Every implementation of an FTAM Initiator shall support:

the kernel protocol and its mandatory parameters with minimum ranges [Minimum required ranges are specified in 17.8.],

the grouping protocol and the <threshold> parameter with a value of at least 2 for use in the file transfer class,

at least one of the read or write protocols [Specific conformance for reading and writing is defined in 17.6 and 17.7.],

and support the applicable procedures defined in ISO 8571-4 clauses 8.1 (FTAM regime establishment), 8.2 (FTAM regime termination), 8.3 (File selection), 8.4 (File deselection), 8.9 (File open), 8.10 (File close), 8.11 (Begin group), 8.12 (End group), and 10 (File general actions). To support the above protocols and procedures the implementation shall always support the kernel functional unit and additionally shall be able to:

request the grouping and at least one of the read or write functional units,

request the file transfer class with the <service class> parameter,

request the document type FTAM-3 using the <document type name> form of the <contents type> parameter,

request the <FTAM quality of service> parameter with value 0 and accept in all cases the returned value 0, and

request a <communication quality of service> consistent with the transport definition in these agreements.

as part of the Filestore initialization procedures in ISO 8571-4 clause 8.1, FTAM regime establishment.

Initiators must be able to operate under all circumstances if the above minimum values are successfully negotiated and returned on an <F-INITIALIZE response> PDU. Initiators must be able to operate with any downward negotiation of requested parameter values as described in the standard.

Should the supporting services break down, such that FTAM communication is impossible, the FTAM protocol machine shall notify the user with an <F-P-ABORT indication> and <diagnostic> value with identifier 1011, as well as any known <further details>.

NOTE - Interworking may not be possible between Initiators not supporting attributes of the Storage Group and Security Group, and Responders requiring these attributes to be used.

17.5 Responders

Every implementation of an FTAM Responder shall support:

the kernel protocol and its mandatory parameters with minimum ranges [Minimum required ranges are specified in 17.8.],

the grouping protocol and the <threshold> parameter with a value of at least 2 for use in the file transfer class,

at least one of the read or write protocols [Specific conformance for reading and writing is defined in 17.6 and 17.7],

and support the applicable procedures, defined in ISO 8571-4 clauses 9.1 (FTAM regime establishment), 9.2 (FTAM regime termination), 9.3 (File selection), 9.4 (File deselection), 9.9 (File open), 9.10 (File close), 9.11 (Begin group), 9.12 (End group), and 10 (File general actions). To support the above protocols and procedures the implementation shall always support the kernel functional unit and additionally shall be able to:

accept requests for the grouping and at least one of the read or write functional units,

accept requests for the file transfer class with the <service class> parameter,

accept the document type FTAM-3 using the <document type name> form of the <contents type> parameter,

accept requests for an <FTAM quality of service> parameter with any value but may respond with the value 0, and

accept requests for a <communication quality of service> consistent with the transport definition in these agreements

as part of the filestore initialization procedures in ISO 8571-4 clause 9.1, FTAM regime establishment.

Responders must be able to operate under all circumstances if the above minimum values are requested on an <F-INITIALIZE request> PDU. Responders must not negotiate upward in the sense described in the standard.

Responders must complete each action requested and supported in a manner consistent with its description in ISO 8571-2 clauses 10 (Actions on complete files) and 11 (Actions for file access), and must interpret each supported attribute in a manner consistent with its definition in ISO 8571-2 clause 12 (File attributes).

Under circumstances where actions cannot be carried out either as requested or consistently with ISO 8571-2 clause 10 (Actions on complete files) and 12 (Actions for file access), the Responder must return at least one diagnostic indicating:

if the failure was due to either a protocol or Filestore failure, and then:

1) precisely which action failed,

2) at least one of the parameters that could not be accommodated with the diagnostic type indicating at least the degree of failure, as given by the action and state result parameter, or

that the failure was due to unforeseen system shutdown.

Should the supporting services break down, such that FTAM communication is impossible, the FTAM protocol machine shall notify the user with an <F-P-ABORT indication> and <diagnostic> with identifier 1011, as well as inform the user of any known <further details>.

17.6 Senders

Every implementation of an FTAM sender shall support the read functional unit as Responder or the write functional unit as Initiator, and support the applicable procedures defined in ISO 8571-4 clauses 11 (State of the bulk data transfer activity), 12 (Bulk data transfer protocol data units), 15 (Bulk data transfer sending entity actions), 17.1 (Discarding), and 17.2 (Cancel).

To support those procedures the implementation shall be able to send files of the document type FTAM-3 and shall be able to send them as user data in PPDUs in blocks of up to 7168 octets.

17.6.1 Initiator Senders

Every implementation of an FTAM sender which is also an FTAM Initiator shall support:

the write functional unit and protocol, and

for the document type FTAM-3 the following bulk data transfer specification parameters:

FADU operation replace

FADU identity first

and support the applicable procedures, defined in ISO 8571-4 clause 13 (Bulk data transfer initiating entity actions).

17.6.2 Responder Senders

Every implementation of an FTAM sender which is also an FTAM Responder shall support:

the read functional unit and protocol, and

for the document type FTAM-3 the following bulk data transfer specification parameters:

1) FADU identity first

2) Access context UA

and support the applicable procedures, defined in ISO 8571-4 clause 14 (Bulk data transfer responding entity actions).

17.7 Receivers

Every implementation of an FTAM receiver shall support the read functional unit as Initiator or the write functional unit as Responder, and support the applicable procedures, defined in ISO 8571-4 clauses 11 (State of the bulk data transfer activity), 12 (Bulk data transfer protocol data units), 16 (Bulk data transfer receiving entity actions), 17.1 (Discarding), and 17.2 (Cancel).

To support those procedures the implementation shall be able to receive files of the document type FTAM-3 and shall be able to receive them as user data in PPDUs in blocks of at least 7168 octets.

17.7.1 Initiator Receivers

Every implementation of an FTAM receiver which is also an FTAM Initiator shall support:

the read functional unit and protocol, and

for the document type FTAM-3 the following bulk data transfer specification parameters:

1) FADU identity first

2) Access context UA

and support the applicable procedures, defined in ISO 8571-4 clause 13 (Bulk data transfer initiating entity actions).

17.7.2 Responder Receivers

Every implementation of an FTAM receiver which is also an FTAM Responder shall support:

the write functional unit and protocol, and

for the document type FTAM-3 the following bulk data transfer specification parameters:

1) FADU operation replace

2) FADU identity first

and support the applicable procedures, defined in ISO 8571-4 clause 14 (Bulk data transfer responding entity actions).

17.8 Minimum ranges

Any implementation of any conformant FTAM configuration shall be able to receive and meaningfully process all mandatory parameters for all functional units supported as well as the <diagnostic> parameter within at least the minimum ranges of values given in table 6. A conforming implementation may support a wider range of values for any parameter.

Table 6 - Required minimal parameter support

Parameter		Minimum Range
general	diagnostic	Values as specified in ISO 8571-3 annex A (Diagnostic parameter values) tables 44, 45 and 46 which correspond directly to mandatory parameters.
	action result	All values
	state result	All values
F-INITIALIZE	functional units[1]	'read' (for initiator/receivers and responder/senders) and 'grouping' or 'write' (for initiator/senders and responder/receivers) and 'grouping'
	presentation context management[2]	'False'
	all others	As specified in 17.4 and 17.5 above
F-SELECT	attributes	Only <filename> is used with a minimum supportable length of 8 characters. Any other attribute supported for other services must have minimum supported lengths as in ISO 8571-2 clause 15 (Minimum attribute ranges), table 2.
	requested access	'read' for initiator/receivers 'read' for responder/senders 'replace' for initiator/senders 'replace' for responder/receivers
F-CREATE	override	For Responders, the values 'create-failure', 'select-old-file' and 'delete-and-create-with-new-attributes' are supported. The value 'delete-and-create-with-old-attributes' is optionally supported. All values are optional for Initiators.

Table 6 - Required minimal parameter support. (concluded)

Parameter		Minimum Range
F-OPEN	processing mode	‘read’ for initiator/receivers ‘read’ for responder/senders ‘replace’ for initiator/senders ‘replace’ for responder/receivers
	contents type	‘FTAM-3’
F-READ	FADU identity	‘first’
	access context	‘UA’
F-WRITE	FADU operation	‘read’ for initiator/receivers ‘read’ for responder/senders ‘replace’ for initiator/senders ‘replace’ for responder/receivers
	FADU identity	‘first’
F-BEGIN-GROUP	threshold[3]	For file transfer (a minimal required function) 2

For any other supported parameters, minimum ranges are taken from the minimum ranges for the attribute corresponding to each as in ISO 8571-2 table 4.

NOTES

1 The parameters, functional units, and presentation context management are not ordered, so "minimum value" cannot be formally defined. The above values are those required for conformance to these Agreements but no value conformant to ISO 8571 for use in other applications is regarded to be in violation of these Agreements.

2 Other functional units (and service classes) for defined implementations may also be valid provided that they are implemented in accordance with these Agreements, specifically subclause 17.8.

3 Every implementation must support the <threshold> value 2 to provide the basic required function of file transfer; any other value in other applications is acceptable.

17.9 Range of values for INTEGER type parameters

The general range for parameters of type INTEGER to the FTAM PCI is as specified in the UNIVERSAL ASN.1 ENCODING RULES clause of the Upper Layers part.

The parameters

FTAM Attributes

filesize

future-filesize

FADU-Identity

fadu-number

may be encoded so that the length of its contents octets is no more than eight octets.

In the case of receiving more than 4 contents octets, the receiver may reject the corresponding FTAM PDU.

NOTE - To guarantee interworking, encoding should be restricted to the range -2^{31} to $2^{31}-1$.

17.10 Use of lower layer services

Support for the Presentation Context Management functional unit is not required.

Implementations will support the Session, Presentation, and ACSE requirements as stated in part 5 of this document.

NOTE - Implementation of the Session Resychronize and the Minor Synchronize functional units is highly recommended, since the F-CANCEL service may be less effective when mapped to S-DATA.

18 Implementation Profiles

This clause defines Implementation Profiles for the specific functions of:

File Transfer

File Access

File Management

Those definitions are expressed in terms of:

Document Types

Attributes

Service Classes (both service elements and their parameters).

This by no means defines all possible Implementation Profiles. The following Implementation Profiles are defined:

T1 : Simple File Transfer

T2 : Positional File Transfer

T3 : Full File Transfer

A1 : Simple File Access

A2 : Full File Access

M1 : Management.

Implementation Agreements have been reached for the following service classes.

File Transfer

File Access

File Management

Unconstrained

File Transfer and Management

NOTE - Any given implementation may support more than one service class.

Support of an Implementation Profile requires adherence to:

corresponding definition in 8571-3 clause 8 and any related procedures in 8571-4 clause 8-17,

requirements given in clauses 5-18 of these Agreements, and

requirements for parameter and attribute support as defined in 17.8.

18.1 General requirements for the defined Implementation Profiles

Implementations will be able to act either as Initiator or Responder or both.

Implementations must support diagnostics as described in clause 13 of these Agreements.

Implementations that support the file access service class will support access to sequential files. Support of sequential files entails hierarchy of depth and arc length equal to 1. Other hierarchy depth and arc lengths are not precluded by these agreements.

18.2 (deleted)

18.3 Document type requirements for the defined Implementation Profiles

Implementations conformant to Implementation Profiles defined in table 7 will support the following document types with the caveats and procedures given.

FTAM-1

FTAM-2

FTAM-3

NBS-6

NBS-7

NBS-8

NBS-9

NOTES

1 Support of this document type entails the naming of FADUs by their position in preorder traversal.

2 Caveat: Other methods of naming FADUs depend on the system, application, and specific file, and as such are not described here.

3 Those document types are defined in annex A and clause 10 of these Agreements, and in ISO 8571-2.

Support for any document type requires the ability to transfer and store the abstract syntax given in its definition. These Agreements do not specify techniques or formats for storage.

NOTE - Specific abstract syntaxes for the parameterized document types NBS-6,7,8 are not specified in these Agreements.

Any document type supported must be identifiable by its document type name as given in ISO 8571-2 and in annex A of these Agreements and, where defined, the parameterization scheme given in clause 10 of these Agreements.

For conformance to NBS-9 a Responder is only required to return the <filename> attribute, subject to local security and access control. All other requested attributes need not be returned.

Systems supporting the NBS-9 document type shall make available an NBS-9 document called "DIRLIS." This document can be used to obtain a listing of files and their associated attributes from a remote Filestore.

Creation and deletion of NBS-9 files are outside the scope of these Agreements.

File security issues related to NBS-9 are subject to the security agreements outlined in clause 16.

18.4 Parameters for the defined Implementation Profiles

Implementations will support the <contents type list> parameter on the <F-INITIALIZE> service element. The initiating service must supply a value for this parameter.

Implementations will support the <diagnostic> parameter as stated in clause 13 of these Agreements.

The <initiator identity> parameter is supported. Use must be consistent with clause 16 of these Agreements.

Implementations are not precluded from using other parameters for security and/or accounting. Responders must state the syntax and the semantics applying to <account> and <charging> parameters. The Responder's minimum implementation is to accept but ignore the <account>.

18.5 Parameter ranges for the defined Implementation Profiles

Parameter ranges for Implementations Profiles are as stated for primitive data types in clause 10 of these Agreements.

18.6 File attribute support for Implementations

Implementations of the Implementation Profiles will support file attributes or attribute groups in the following ways:

a) mandatory - This feature is mandatory in the ISO 8571-2 standard and shall therefore be implemented by all implementations claiming conformance to these Agreements;

b) supported - This feature shall be implemented by all implementations claiming conformance to these Agreements (for attributes, this implies that at least the minimum range of attribute values, as defined in ISO 8571-2 clause 15, shall be supported). Conformant implementations shall also be able to interwork with other implementations that do not support this feature by negotiating out the corresponding features;

c) optionally supported - Implementations claiming conformance to these Agreements may or may not implement this feature (for attributes, this implies that at least either the minimum range of attribute values, as defined in ISO 8571-2 clause 15, shall be supported or that the "no value available" result shall be supplied). If an attribute group with a support level of "optionally supported" is chosen to be supported, then all the attributes of this group that are classified as "mandatory" or "supported" shall be supported;

d) not supported - This feature is outside the scope of these Agreements.

Kernel Group	mandatory
Filename	mandatory
Permitted Actions	mandatory
Contents Type	mandatory
Storage Group	optionally supported
Storage Account	optionally supported
Date and Time of Creation	optionally supported
Date and Time of Last Modification	optionally supported
Date and Time of Last Read Access	optionally supported
Date and Time of Last Attribute Modification	optionally supported
Identity of Creator	optionally supported
Identity of Last Modifier	optionally supported
Identity of Last Reader	optionally supported
Identity of Last Attribute Modifier	optionally supported
File Availability	supported
Filesize	supported
Future Filesize	optionally supported
Security Group	optionally supported
Access Control	supported
Legal Qualifications	optionally supported
Private Group	not supported

Table 7 - Implementation profile support requirements

Functional Units	Service Classes (see note 8)				
	T	M	A	TM	Uncon-strained
Kernel	T1, T2, T3	M1	A1, A2	see note 4	see note 5
Read (see note 3)	T1, T2, T3		A1, A2		
Write (see note 3)	T1, T2, T3		A1, A2		
Limited File Management	see note 6	M1	see note 6		
Enhanced File Management		M1			
Grouping	T1, T2, T3	M1			
File Access			A1, A2		
Document Types					
FTAM-1	T1, T2, T3	[M1]	A1, A2		
FTAM-2	T2, T3	[M1]	A1, A2		
FTAM-3	T1, T2, T3	[M1]	A1, A2		
NBS-6	[T2], T3	[M1]	[A1], A2		
NBS-7	[T2], T3	[M1]	[A1], A2		
NBS-8	T3	[M1]	A2		
NBS-9	[T1], [T2], [T3]	[M1]			

NOTES
to 18.3 and table 7
1 The Management Implementation Profile is only to be implemented in conjunction with one of the Transfer or Access Profiles.

2 Profile T2 is subset of T3. A1 and T1 are subsets of A2 and T2, respectively.

3 Profiles T1, T2, and T3 require the support of read and/or write functional units.

4 Support of the <File Transfer and Management> service class is optional. The rules for including it in a request and for the response to it are as given in ISO 8571-3, clause 10.1. Any implementation including TM in the request must be prepared for the possibility that it might be removed from the response.

5 The support of the <Unconstrained> service class is outside the scope of these Implementation Profiles.

6 Limited File Management is not required for the T- and A- Implementation Profiles, but very often it will be

a user request to have limited file management functionality available together with file transfer and file access functions. So Limited File Management may be added as an option to the T- and A- Implementation Profiles.

7 [ ] in table 7 specifies that the document type is optional for the respective Implementation Profile. For M1 the support level depends on the T- or A- Implementation Profile, in conjunction with which M1 is implemented.

8 The Implementation Profiles specify functionality which includes the requirements for conformant implementations as specified in clause 17. This is a general basic requirement and is not also reflected in table 7.

19 PROVISION OF SPECIFIC FUNCTION

19.1 Implementation Profile T1: Simple File Transfer

Implementation Profile T1 provides the function of transferring entire files at the external file service level for files with an unstructured constraint set. This includes support of the document types:

FTAM-1 "ISO FTAM unstructured text"

FTAM-3 "ISO FTAM unstructured binary"

NBS-9 "NBS-9 file directory file" (optional)

This Implementation Profile supports file transfer and not file access, that is, the ability to

read a complete file, and/or

write (replace, extend) to a file.

19.2 Implementation Profile T2: Positional File Transfer

Implementation Profile T2 provides the function of transferring files at the external file service level for files with an unstructured or flat constraint set. This includes support of the document types:

FTAM-1 "ISO FTAM unstructured text"

FTAM-2 "ISO FTAM sequential text"

FTAM-3 "ISO FTAM unstructured binary"

NBS-6 "NBS-6 FTAM sequential file" (optional)

NBS-7 "NBS-7 FTAM random access file" (optional)

NBS-9 "NBS-9 file directory file" (optional)

This Implementation Profile supports file transfer and not file access, that is, the ability to

> read a complete file or a single FADU which is identified by position, and/or
>
> write (replace, extend, insert depending on constraint set and document type) to a file or an FADU.

This Implementation Profile is upwardly compatible to T1 for the transfer of unstructured files.

19.3 Implementation Profile T3: Full File Transfer

Implementation Profile T3 provides the function of transferring files at the external file service level for files with an unstructured, flat or general hierarchical constraint set. This includes support of the document types:

FTAM-1 "ISO FTAM unstructured text"

FTAM-2 "ISO FTAM sequential text"

FTAM-3 "ISO FTAM unstructured binary"

NBS-6 "NBS-6 FTAM sequential file"

NBS-7 "NBS-7 FTAM random access file"

NBS-8 "NBS-8 FTAM indexed file"

NBS-9 "NBS-9 file directory file" (optional)

This Implementation Profile supports file transfer and not file access, that is, the ability to

> read a complete file or a single FADU which is identified by key or by position, and/or
>
> write (replace, extend, insert depending on constraint set and document type) to a file or an FADU.

This Implementation Profile is upwardly compatible to T1 for the transfer of unstructured files.

19.4 Implementation Profile A1: Simple File Access

Implementation Profile A1 provides the function of transfer of and access to files with unstructured or flat constraint sets at the external file service level. This includes support of the document types:

FTAM-1 "ISO FTAM unstructured text"

FTAM-2 "ISO FTAM sequential text"

FTAM-3 "ISO FTAM unstructured binary"

NBS-6 "NBS-6 FTAM sequential file" (optional)

NBS-7 "NBS-7 FTAM random access file" (optional)

This Implementation Profile supports file transfer and file access, that is the ability to

read a complete file or FADUs which are identified by position,

write (replace, extend, insert depending on constraint set and document type) to a file or an FADU,

locate and erase within files.

19.5 Implementation Profile A2: Full File Access

Implementation Profile A2 provides the function of transfer of and access to files with unstructured or flat constraint sets at the external file service level. This includes support of the document types:

FTAM-1 "ISO FTAM unstructured text"

FTAM-2 "ISO FTAM sequential text"

FTAM-3 "ISO FTAM unstructured binary"

NBS-6 "NBS-6 FTAM sequential file"

NBS-7 "NBS-7 FTAM random access file"

NBS-8 "NBS-8 FTAM indexed file"

This Implementation Profile supports file transfer and file access, that is, the ability to

read from a complete file, or from a series of FADUs which are identified by key or by position,

write (replace, extend, insert depending on constraint set and document type) to a file or an FADU,

locate and erase within files.

19.6 Implementation Profile M1: Management

Implementation Profile M1 provides the function for an Initiator to manage the files within the Virtual Filestore, to which access is provided by the Responder. Management includes the services of:

creating a file

deleting a file

reading attributes of a file

changing attributes of a file.

20 Harmonization

The Implementation Profiles for File Transfer, File Access and Management correspond to the Profiles of SPAG (Standards Promotion and Application Group) in Europe, so that interworking will be possible. Those Profiles are described in the "Guide to the Use of Standards" (GUS); they are the basis for the Functional Standards as defined by CEN/CENELEC (Comite Europeenne de Normalization).

Table 8 - Implementation profiles (OIW) and profiles (SPAG/CEN-CLC)

Implementation Profile	SPAG / CEN-CENELEC
T1	A/111
T2	A/112
T3	A/113
A1	A/122
A2	A/123
M1	A/13

Annex A (normative)

FTAM Document Types

Editor's Note - The objects defined in A.5 through A.8, B.2, C.3 and C.4 will be removed from this document after ISO/IEC ISP 10607-2 and ISO/IEC ISP 10607-2/Amd.1 are published. During the period between publishing the ISP and removal of the definitions from this document, the definitions in the ISP will take precedence over this document. When the object definitions are removed, clauses A.1 through A.4, B.1, C.1 and C.2 will be changed to point to the ISP.

The objects defined in A.5 through A.8, B.2, C.3 and C.4 will be removed from this document after ISO/IEC ISP 10607-2 and ISO/IEC ISP 10607-2/Amd.1 are published. During the period betweem publishing the ISP and removal of the definitions from this document, the definitions in the ISP will take preceedence over this document. When the object definitions are removed, clauses A.1 through A.4, B.1, C.1 and C.2 will be changed to point to the ISP.

A.1 NBS-6 Sequential file document type

This object with Object Identifier

{iso identified-organization icd(9999) organization-code(1) document-type(5) sequential(6)}

was withdrawn on March 16, 1990. It was replaced with the object NBS-6 Sequential file document type with the Object Identifier

{iso identified-organization oiw(14) ftamsig(5) document-type(5) sequential(6)}

defined in part 9, A.5.

A.2 NBS-7 Random access file

This object with Object Identifier

{iso identified-organization icd(9999) organization-code(1) document-type(5) random-file(7)}

was withdrawn on March 16, 1990. It was replaced with the object NBS-7 Random access file document type with the Object Identifier

{iso identified-organization oiw(14) ftamsig(5) document-type(5) random-access(7)}

defined in part 9, A.6.

A.3 NBS-8 Indexed sequential file

This object with Object Identifier

{iso identified-organization icd(9999) organization-code(1) document-type(5) indexed-file(8)}

was withdrawn on March 16, 1990. It was replaced with the object NBS-8 Indexed sequential file document type with the Object Identifier

{iso identified-organization oiw(14) ftamsig(5) document-type(5) indexed-file(8)}

defined in part 9, A.7.

A.4 NBS-9 File directory file

This object with Object Identifier

{iso identified-organization icd(9999) organization-code(1) document-type(5) file directory(9)}

was withdrawn on March 16, 1990. It was replaced with the object NBS-9 File directory file document type with the Object Identifier

{iso identified-organization oiw(14) ftamsig(5) document-type(5) file-directory(9)}

defined in part 9, A.8.

A.5 NBS-6 Sequential file document type

A.5.1 Entry Number: NBS-6

A.5.2 Information objects

Table 9 - Information objects in NBS-6

<table>
<tr><td>document type name</td><td>{iso identified-organization oiw(14) ftamsig(5) document-type(5) sequential(6)}
"NBS-6 FTAM sequential file"</td></tr>
<tr><td>abstract syntax names:
a) name for asname1

b) name for asname2</td><td>{iso identified-organization oiw(14) ftamsig(5) abstract-syntax(2) nbs-as1(1)}
"NBS abstract syntax AS1"

{iso standard 8571 abstract-syntax(2) ftam-fadu(2)}
"FTAM FADU"</td></tr>
<tr><td>transfer syntax names:</td><td>{joint-iso-ccitt asn1(1) basic-encoding(1)}
"Basic Encoding of a single ASN.1 type"</td></tr>
<tr><td colspan="2">parameter syntax:
<pre>
PARAMETERS ::= SEQUENCE OF CHOICE {Parameter0, Parameter1, Parameter2}
Parameter0 ::= [0] INTEGER {univer-time (23),
                    gen-time (24),
                    boolean (1),
                    null (5) }
Parameter1 ::= [1] SEQUENCE {
        universal-class-number-1 INTEGER { int   (2),
                                          bit   (3),
                                          ia5   (22),
                                          graphic (25),
                                          general (27),
                                          octet   (4)},
        string-length INTEGER }
Parameter2 ::= [2] SEQUENCE {
          private-class-number INTEGER {float (0)},
          length-1   INTEGER,
          length-2   INTEGER }
</pre></td></tr>
<tr><td>file model</td><td>{iso standard 8571 file-model(3) hierarchical(1)}
"FTAM hierarchical file model"</td></tr>
<tr><td>constraint set</td><td>{iso standard 8571 constraint-set(4) sequential-flat(2)}
"FTAM sequential flat constraint set"</td></tr>
<tr><td colspan="2">file contents:
<pre>
        Datatype1 ::= PrimType -- NBS-AS1

        Datatype2 ::= Node-Descriptor-Data-Element
</pre></td></tr>
</table>

A.5.3 Scope and field of application

The document type defines the contents of a file for storage, for transfer and access by FTAM.

> **NOTE -** Storage refers to apparent storage within the Virtual Filestore.

A.5.4 References

ISO 8571, Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management

A.5.5 Definitions

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

A.5.6 Abbreviations

FTAM File Transfer, Access and Management

A.5.7 Document semantics

The document consists of zero, one or more file access data units. Each FADU contains precisely zero or one data unit which consists of zero, one or more data elements. The order of each of these elements is significant.

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 9) These definitions appear in ISO 8571-2. As additional constraints FADU identity will be limited to "begin," "end," "first" and "next".

For a specific file the number of data elements in a data unit is given by the parameters. Each data element is a data type from the set of primitive data types defined in the annex C.3 of this document. Each data unit contains the same data element types in the same order as all other data units. These types are determined by the parameters 0 through 2.

For datatype 1, the string length field of Parameter1 specifies the length of the value in octets for the INTEGER, BIT STRING and OCTET STRING types. For character-type data elements, the string-length indicates the actual number of characters from the specified character set, not including any escape sequences or overhead from the character encoding.

For floating point numbers, finite form, length-1 and length-2 specify the length in bits of mantissa and exponent, respectively. The length-1 and length-2 values are irrelevant for the other choices of floating point numbers.

A.5.8 Abstract syntactic structure

The abstract syntactic structure of the document is a hierarchically structured file as defined in the ASN.1 module ISO8571-FADU in ISO 8571, in which each of the file access data units has the abstract syntactic structure of NBS-AS1 as defined by the parameters.

A.5.9 Definition of transfer

A.5.9.1 Datatype definitions

The file consists of data values which are of either

a) Datatype1 defined in table 9, where the PrimType in the datatype is given by the NBS-AS1 definition; or

b) Datatype2 defined in table 9, the ASN.1 datatype declared as "Data-Element" in the ASN.1 module ISO8571-FADU.

A.5.9.2 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 data elements from the document. All values are transmitted in the same (but any) presentation context established to support the abstract syntax name "asname1" or

b) a value of "Datatype2." All values are transmitted in the same (but any) presentation context established to support the abstract syntax name "asname2."

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 presentation data values in the same presentation context, and 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 (e.g., document type parameters and transfer syntaxes).

A.5.9.3 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 data elements 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.

A.5.10 Transfer syntax

An implementation supporting this document type shall support the transfer syntax generation rules named in table 9 for all presentation data values transferred. Implementation may optionally support other named transfer syntaxes.

A.5.11 ASE specific specifications for FTAM

A.5.11.1 Structural Simplification

This structural simplification loses information.

The document type NBS-6 may be simplified to the document type FTAM-3 (allowed only when reading the file). The octet representation of the transferred data is unpredictable. It will usually correspond to the data values as stored in the local Real Filestore of the Responder.

A.5.11.2 Access context selection

A document of type NBS-6 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.

A.5.11.3 The INSERT operation

When the <INSERT> operation is applied at the end of file, the transferred material shall be the series of FADUs which would be generated by reading any NBS-6 document with the same parameter values in access context FA.

A.6 NBS-7 Random access file

A.6.1 Entry number: NBS-7

A.6.2 Information objects

Table 10 - Information objects in NBS-7

document type name	{iso identified-organization oiw(14) ftamsig(5) document-type(5) random-file(7)} "NBS-7 FTAM random access file"
abstract syntax names: a) name for asname1 b) name for asname2	 {iso identified-organization oiw(14) ftamsig(5) abstract-syntax(2) nbs-as1(1)} "NBS abstract syntax AS1" {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 OF CHOICE {Parameter0, Parameter1, Parameter2}
Parameter0 ::= [0] INTEGER {univer-time (23),
gen-time (24),
boolean (1),
null (5) }
Parameter1 ::= [1] SEQUENCE {
universal-class-number-1 INTEGER { int (2),
bit (3),
ia5 (22),
graphic (25),
general (27),
octet (4)},
string-length INTEGER }
Parameter2 ::= [2] SEQUENCE {
private-class-number INTEGER {float (0)},
length-1 INTEGER,
length-2 INTEGER }	

file model	{iso standard 8571 file-model(3) hierarchical(1)} "FTAM hierarchical file model"
constraint set	{iso identified-organization oiw(14) ftamsig(5) constraint-set(4) nbs ordered-flat(1)} "NBS ordered flat constraint set"
file contents: Datatype1 ::= PrimType -- NBS-AS1 Datatype2 ::= CHOICE { Node-Descriptor-Data-Element, Enter-Subtree-Data-Element } Exit-Subtree-Data-Element }	

A.6.3 Scope and field of application

The document type defines the contents of a file for storage, for transfer and access by FTAM.

> **NOTE -** Storage refers to apparent storage within the Virtual Filestore.

A.6.4 References

ISO 8571, Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management

A.6.5 Definitions

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

A.6.6 Abbreviations

FTAM File Transfer, Access and Management

A.6.7 Document semantics

The document consists of zero, one or more file access data units. Each FADU contains precisely zero or one data unit which consists of zero, one or more data elements. The order of each of these elements is significant.

The document structure takes any of the forms allowed by the FTAM hierarchical file model as constrained by the NBS-ordered-flat constraint set (see table 10). These definitions appear in annex B.2 of this document.

For a specific file the number of data elements in a data unit is given by the parameters. Each data element is a data type from the set of primitive data types defined in the annex C.3 of this document. Each data unit contains the same data element types in the same order as all other data units. These types are determined by the parameters 0 through 2.

For datatype 1, the string length field of Parameter1 specifies the length of the value in octets for the INTEGER, BIT STRING and OCTET STRING types. For character-type data elements, the string-length indicates the actual number of characters from the specified character set, not including any escape sequences or overhead from the character encoding.

For floating point numbers, finite form, length-1 and length-2 specify the length in bits of mantissa and exponent, respectively. The length-1 and length-2 values are irrelevant for the other choices of floating point numbers.

A.6.8 Abstract syntactic structure

The abstract syntactic structure of the document is a hierarchically structured file as defined in the ASN.1 module ISO8571-FADU in ISO 8571, in which each of the file access data units has the abstract syntactic structure of NBS-AS1 as defined by the parameters.

A.6.9 Definition of transfer

A.6.9.1 Datatype definitions

The file consists of data values which are of either:

a) Datatype1 defined in table 10, where the PrimType in the datatype is given by the NBS-AS1 definition;

b) Datatype2 defined in table 10, the ASN.1 datatype declared as "Data-Element" in the ASN.1 module ISO8571-FADU.

A.6.9.2 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 data elements from the document. All values are transmitted in the same (but any) presentation context established to support the abstract syntax name "asname1";

b) a value of "Datatype2." All values are transmitted in the same (but any) presentation context established to support the abstract syntax name "asname2."

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 presentation data values in the same presentation context, and 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 (e.g., document type parameters and transfer syntaxes).

A.6.9.3 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 data elements 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.

A.6.10 Transfer syntax

An implementation supporting this document type shall support the transfer syntax generation rules named in table 10 for all presentation data values transferred. Implementation may optionally support other named transfer syntaxes.

A.6.11 ASE specific specifications for FTAM

A.6.11.1 Structural simplification

This structural simplification loses information.

The document type NBS-7 may be accessed as a document type FTAM-3 (allowed only when reading the file) by specifying document type FTAM-3 in the <contents type> parameter in <F-OPEN request>, and limiting access context to UA on F-READ.

The octet representation of the transferred data is unpredictable. It will usually correspond to the data values as stored in the local Real Filestore of the Responder.

A document of type NBS-7 can be accessed as a document of type NBS-6 (allowed only when reading the file) by specifying document type NBS-6 with appropriate data type parameters in the <contents type> parameter on the <F-OPEN request>.

A.6.11.2 Access context selection

A document of type NBS-7 may be accessed in any one of the access contexts defined in the NBS ordered 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.

A.6.11.3 The INSERT operation

When the <INSERT> operation is applied at the end of file, the transferred material shall be the series of FADUs which would be generated by reading any NBS-7 document with the same parameter values in access context FA.

A.7 NBS-8 Indexed sequential file

A.7.1 Entry Number: NBS-8

A.7.2 Information Objects

Table 11 - Information objects in NBS-8

<table>
<tr><td>document type name</td><td>{iso identified-organization oiw(14) ftamsig(5) document-type(5) indexed-file(8)}
"NBS-8 FTAM indexed file"</td></tr>
<tr><td>abstract syntax names:
a) name for asname1

b) name for asname2</td><td>{iso identified-organization oiw(14) ftamsig(5) abstract syntax(2) nbs-as1(1)}
"NBS abstract syntax AS1"

{iso standard 8571 abstract-syntax(2) ftam- fadu(2)}
"FTAM FADU"</td></tr>
<tr><td>transfer syntax names:</td><td>{joint-iso-ccitt asn1(1) basic-encoding(1)}
"Basic Encoding of a single ASN.1 type"</td></tr>
<tr><td colspan="2">parameter syntax:
PARAMETERS ::= SEQUENCE {DataTypes, KeyType, KeyPosition}

DataTypes ::= SEQUENCE OF CHOICE {Parameter0, Parameter1, Parameter2}

KeyType ::= CHOICE {Parameter0, Parameter1, Parameter2}
-- Parameter0, Parameter1, Parameter2, as defined for the
-- document types NBS-6, NBS-7

KeyPosition::= INTEGER</td></tr>
<tr><td>file model</td><td>{iso standard 8571 file-model(3) hierarchical(1)}
"FTAM hierarchical file model"</td></tr>
<tr><td>constraint set</td><td>{iso standard 8571 constraint-set(4) ordered-flat(3) }
"FTAM ordered flat constraint set"</td></tr>
<tr><td colspan="2">file contents:
Datatype1 ::= PrimType -- NBS-AS1

Datatype2 ::= CHOICE { Node-Descriptor-Data-Element,
Enter-Subtree-Data-Element,
Exit-Subtree-Data-Element }

Datatype3 ::= Primtype -- as defined by the NBS abstract syntax AS1</td></tr>
</table>

A.7.3 Scope and field of application

The document type defines the contents of a file for storage, for transfer and access using FTAM.

> **NOTE -** Storage refers to apparent storage within the Virtual Filestore.

A.7.4 References

ISO 8571, Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management

A.7.5 Definitions

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

A.7.6 Abbreviations

FTAM File Transfer, Access and Management

A.7.7 Document semantics

The document consists of zero, one or more file access data units. Each FADU contains precisely one data unit which consists of zero, one or more data elements. The order of each of these elements is significant.

The document structure takes any of the forms allowed by the FTAM hierarchical file model as constrained by the FTAM ordered flat constraint set (see table 11). These definitions appear in ISO 8571-2.

The following additional requirements are specified for the use of the ordered flat constraint set:

> The FADU identities "first," "last," and "node number" are not required for conformant implementations
>
> The identities "next" and "previous" are allowed for all FADUs

Each data element is a data type from the set of primitive data types defined in annex C.3 of this document. Each data unit contains the same data element types in the same order as all other data units. These types and their respective maximum lengths are defined by the <DataTypes> parameter.

For Datatype1 and Datatype3, the string length field of Parameter1 specifies the length of the value in octets for the INTEGER, BIT STRING and OCTET STRING types. For character-type data elements, the string-length indicates the actual number of characters from the specified character set, not including any escape sequences or overhead from the character encoding.

For floating point numbers, finite form, length-1 and length-2 specify the length in bits of mantissa and exponent, respectively. The length-1 and length-2 values are irrelevant for the other choices of floating point numbers.

Each data unit in the file has a key associated with it, which is the user-coded form of Node-Name. The key of each data unit is of the same data type as the key of all other data units in the file and is a single data element from the set of primitive data types defined in annex C.3.

The type and length of the key are defined by the <KeyType> parameter.

The primitive data types and minimum size ranges of each unit which an implementation must accept as a key value are given in the following table 12.

Table 12 - Datatypes for keys

Key Type	Minimum Range (octets)	Order
ASN.1 INTEGER	(1-2)	increasing numeric value
ASN.1 IA5String	(1-16)	lexical order
ASN.1 GraphicString	(1-16)	lexical order
ASN.1 GeneralString	(1-16)	lexical order
ASN.1 OCTET STRING	(1-16)	increasing value
ASN.1 GeneralizedTime		increasing time value
ASN.1 UniversalTime		increasing time value
NBS-AS1 FloatingPointNumber		increasing numeric value

The position of the key in the data unit is specified by the < KeyPosition> parameter.

KeyPosition = 0 implies the key is not part of the data

position > 0 specifies the actual data element in the data unit.

A.7.8 Abstract syntactic structure

The abstract syntactic structure of the document is a hierarchically structured file as defined in the ASN.1 module ISO8571-FADU in ISO 8571, in which each of the file access data units has the abstract syntactic structure of NBS-AS1 as defined by the parameters.

A.7.9 Definition of transfer

A.7.9.1 Datatype definitions

The file consists of data values which are of

a) Datatype1 defined in table 11, where the PrimType in the datatype is given by the NBS-AS1 definition; or

b) Datatype2 defined in table 11, the ASN.1 datatype declared as "Data-Element" in the ASN.1 module ISO8571-FADU; or

c) Datatype3 defined in table 11, which specifies the user-coded form of the Node-Name in the FTAM FADU abstract syntax, where user-coded is defined as EXTERNAL.

A.7.9.2 Presentation data values

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

a) one value of the ASN.1 datatype "Datatype1," carrying one of the data elements from the document. All values are transmitted in the same (but any) presentation context established to support the abstract syntax name "asname1" or

b) a value of "Datatype2." All values are transmitted in the same (but any) presentation context established to support the abstract syntax name "asname2"; or

c) a value of "Datatype3" carrying a key. All values are transmitted in the same (but any) presentation context established to support the abstract syntax name "asname1."

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 presentation data values in the same presentation context, and 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 (e.g., document type parameters and transfer syntaxes).

A.7.9.3 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 data elements 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.

A.7.10 Transfer syntax

An implementation supporting this document type shall support the transfer syntax generation rules named in table 11 for all presentation data values transferred. Implementation may optionally support other named transfer syntaxes.

A.7.11 ASE specific specifications for FTAM

A.7.11.1 Structural simplification

This simplification loses information.

The document type NBS-8 may be accessed as a document type FTAM-3 (allowed only when reading the file) by specifying document type FTAM-3 in the <contents type> parameter in <F-OPEN request>, and limiting access context to UA on F-READ.

The octet representation of the transferred data is unpredictable. It will usually correspond to the data values as stored in the local Real Filestore of the Responder.

A document of type NBS-8 can be accessed as a document of type NBS-6 (allowed only when reading the file) by specifying document type NBS-6 with appropriate data type parameters in the <contents type> parameter on the <F-OPEN request>. The traversal order of the FADUs must be maintained.

> **NOTE -** The traversal order is as reading the file as NBS-8 in key order.

A.7.11.2 Access context selection

A document of type NBS-8 may be accessed in any one of the access contexts defined in the FTAM ordered 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.

A.7.11.3 The INSERT operation

When the <INSERT> operation is applied the transferred material shall be the series of FADUs which would be generated by reading any NBS-8 document with the same parameter values in access context FA.

The insertion of a new FADU after an already existing FADU will be indicated via a diagnostic on TRANSFER-END.

A.7.11.4 The EXTEND operation

This operation is excluded for use with this document type.

A.8 NBS-9 File directory file

A.8.1 Entry Number: NBS-9

A.8.2 Information objects

Table 13 - Information objects in NBS-9

document type name	{iso identified-organization oiw(14) ftamsig(5) document-type(5) file-directory(9)} "NBS-9 FTAM file directory file"
abstract syntax names:	{iso identified-organization oiw(14) ftamsig(5) abstract-syntax(2) nbs-as2(2)} "NBS file directory entry abstract syntax"
transfer syntax names:	
parameter syntax	

```
PARAMETERS ::=  [0] IMPLICIT BIT STRING {

        -- Kernel group
            read-filename (0),
            read-permitted-actions (1),
            read-contents-type (2),

        -- Storage group
            read-storage-account (3),
            read-date-and-time-of-creation (4),
            read-date-and-time-of-last-modification (5),
            read-date-and-time-of-last-read-access (6),
            read-date-and-time-of-last-attribute-modification(7),
            read-identity-of-creator (8),
            read-identity-of-last-modifier (9),
            read-identity-of-last-reader (10),
            read-identity-of-last-attribute-modifier (11),
            read-file-availability (12),
            read-filesize (13),
            read-future-filesize (14),

        -- Security group
            read-access-control (15),
            read-legal-qualifications (16),

        -- Private group
            read-private-use (17) }
```

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 ::= FileDirectoryEntry
                --As defined by NBS-AS2 in annex C,
                --C.4  of this document
```

A.8.3 Scope and field of application

This document defines the contents of a file for transfer (not for storage) using FTAM.

A.8.4 References

ISO 8571, Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management.

A.8.5 Definitions

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

A.8.6 Abbreviations

FTAM File Transfer, Access and Management.

A.8.7 Document Semantics

The document consists of one file access data unit, which consists only of zero, one or more data elements of type <FileDirectoryEntry> (defined in NBS-AS2).

The document structure takes any of the forms allowed by the FTAM hierarchical file model as constrained by the unstructured constraint set. These definitions appear in ISO 8571-1.

The parameter of the document type is used on <F-OPEN request> to specify the desired attributes of each of the files on the Filestore, when reading the document.

A.8.8 Abstract syntactic structure

The abstract syntactic structure of the document is a series of file directory entries, each of which is defined by the <FileDirectoryEntry> definition in NBS-AS2.

Additional constraints are defined for this document type: File access actions are restricted to Read. File-directory files may not be Written or Modified (except as a side effect of actions performed on individual files contained within a file directory).

A.8.9 Definition of transfer

A.8.9.1 Datatype definition

The file consists of zero or more values of Datatype1 defined in table 13.

A.8.9.2 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 file directory entries 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 13.

A.8.9.3 Sequence of presentation data values

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

A.8.10 Transfer syntax

An implementation supporting this document type shall support the transfer syntax generation rules named in table 13 for all presentation data values transferred. Implementations shall optionally support other named transfer syntaxes.

A.8.11 ASE specific specifications for FTAM

Relaxation is allowed to any bitstring combination of the document type parameter.

Annex B (normative)

Constraint Sets

B.1 NBS ordered flat constraint set

This object with Object Identifier

{iso identified-organization icd(9999) organization-code(1) constraint-set(4) nbs-ordered-flat(1)}

was withdrawn on March 16, 1990. It was replaced with the object NBS ordered flat constraint set with the Object Identifier

{iso identified-organization oiw(14) ftamsig(5) constraint-set(4) nbs-ordered-flat(1)}

defined in part 9, B.2.

B.2 NBS ordered flat constraint set definition

B.2.1 Field of application

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

B.2.2 Basic constraints

Table 14 - Basic constraints for NBS ordered flat

Constraint set descriptor	"NBS ordered flat constraint set"
Constraint set identifier	{iso identified-organization oiw(14) ftamsig(5) constraint-set(4) nbs-ordered-flat(1)}
Node name	None
File access actions	Locate, Read, Insert, Erase, Replace
Qualified action	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 or UA with FADU identity of 'begin'.
Write whole file (append)	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'.
Write whole file (replace)	Transfer the series of leaf FADUs which would be generated by reading the whole file in access-context HA; perform the transfer with FADU identity 'begin' and file action of 'replace'.

B.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 may have an associated data unit; all arcs from the root node shall be of length one.

B.2.4 Action constraints

Insert: The <Insert> action is allowed only at the end of file.If the FADU identity is "end" the new node is inserted following all existing nodes in the file. If the FADU identity is "node number," the number must be at least one greater than the node number of the last existing node. Any nodes between the last existing node and the new node are empty, i.e., nodes without data. If the FADU identity is a "node number" not greater than that of the last existing node, an error will occur. 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 a solitary root node without an associated data unit.

NOTE - It is the intention when using this constraint set to allow for emptying an FADU, i.e., leaving an FADU with a DU of data length 0 (or without a DU); afterwards data may be reinserted into this hole. In order to empty an FADU, the <Replace> operation may be used with new data of length zero (or with an FADU whose <data exists> bit is set to "false" and no DU). Refilling the hole is accomplished by a <Replace> operation with the new DU (or with the new FADU, whose <data exists> bit is set to "true" and the new DU).

B.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 "node number" greater than or equal to one. The actions with which these identities can be used are given in the following table.

Table 15 - Identity constraints in NBS ordered flat

Action	Begin	End	First	Last	Current	Next	Previous	Node No.
Locate	valid	valid	valid	valid	valid	valid	valid	valid
Read	whole		leaf	leaf	leaf	leaf	leaf	leaf
Insert		leaf						leaf
Erase	whole							
Replace	whole		leaf	leaf	leaf	leaf	leaf	leaf

Annex C (normative)

Abstract Syntaxes

C.1 Abstract Syntax NBS-AS1

This object with Object Identifier

{iso identified-organization icd(9999) organization-code(1) abstract-syntax(2) nbs-as1(1)}

was withdrawn on March 16, 1990. It was replaced with the object Abstract syntax NBS-AS1 with the Object Identifier

{iso identified-organization oiw(14) ftamsig(5) abstract-syntax(2) nbs-as1(1)}

defined in part 9, C.3.

C.2 Abstract Syntax NBS-AS2

This object with Object Identifier

{iso identified-organization icd(9999) organization-code(1) abstract-syntax(2) nbs-as2(2)}

was withdrawn on March 16, 1990. It was replaced with the object Abstract syntax NBS-AS2 with the Object Identifier

{iso identified-organization oiw(14) ftamsig(5) abstract-syntax(2) nbs-as2(2)}

defined in part 9, C.4.

C.3 Abstract Syntax NBS-AS1 definition

Abstract syntax name: {iso identified-organization oiw(14) ftamsig(5) abstract-syntax(2) nbs-as1(1)}

"NBS abstract syntax AS1"

This is an abstract syntax for the set of presentation data values, each of which is a value of the ASN.1 type NBS-AS1.PrimType

```
NBS-AS1 DEFINITIONS ::=
BEGIN
PrimType ::= CHOICE
                        {       INTEGER,
                                BIT STRING,
                                BOOLEAN,
                                IA5String,
                                GraphicString,
                                GeneralString,
                                OCTET STRING,
                                UTCTime,
                                GeneralizedTime,
                                NULL,
                                FloatingPointNumber }

                                --      The support for IA5String is the ISO 646, IRV G0
                                --       character set and the ISO 646, IRV C0 set
                                --      The minimum level of support for GraphicString is
                                --      the ISO 646, IRV G0 character set and the 8859-1
                                --      G0 and G1 sets.
                                --      The minimum level of support for GeneralString is
                                --      the ISO 646, IRV G0 character set and the 8859-1
                                --      G0 and G1 character sets, and ISO 646, IRV C0 set.

FloatingPointNumber ::=         [PRIVATE 0]    CHOICE {
                                finite [0] IMPLICIT SEQUENCE
                        {       Sign,
                                mantissa BIT STRING,
                                --      first bit must be 1
                                        exponent INTEGER},
                                        infinity [1] IMPLICIT Sign,
                                        signalling-nan [2] IMPLICIT NaN,
                                        quiet-nan [3] IMPLICIT NaN,
                                        zero [4] IMPLICIT NULL }

Sign    ::= INTEGER { positive (0), negative (1) }
NaN     ::= INTEGER
END
```

For this abstract syntax the following transfer syntax can be used

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

"Basic Encoding of a single ASN.1 type"

NOTES

1 The mantissa is a number in the range (1/2 <mantissa<1).

2 The value is equal to mantissa * 2 ^{exponent}.

3 The first bit in the mantissa is most significant.

4 See IEEE 754 for definitions of terminology, such as NaN.

5 A minimum length range (in bits) is required for the components of <FloatingPointNumber>, as follows: mantissa 1-23 bits, and exponent 0-8 bits.

C.4 Abstract Syntax NBS-AS2 definition

Abstract syntax name: { iso identified-organization oiw(14) ftamsig(5) abstract-syntax(2) nbs-as2(2) }

"NBS file directory entry abstract syntax"

This is an abstract syntax for the set of presentation data values, each of which is a value of the ASN.1 Type NBS-AS2.FileDirectoryEntry.

```
NBS-AS2 DEFINITIONS ::=
BEGIN
FileDirectoryEntry ::=[PRIVATE 2] Read-Attributes
Read-Attributes ::=ISO8571-FTAM.Read-Attributes
END
```

For this abstract syntax the following transfer syntax will be used

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

"Basic Encoding of a single ASN.1 type"

C.5 Abstract Syntax "FTAM unstructured text abstract syntax"

This abstract syntax is defined as DataType1 (File Contents) in table 19 of ISO 8571-2, annex B.

C.6 Abstract Syntax "FTAM unstructured binary abstract syntax"

This abstract syntax is defined as DataType1 (File Contents) in table 21 of ISO 8571-2, annex B.

Annex D (informative)

FTAM-1 Document Type Tutorial

D.1 Introduction

This annex is informative. It does not specify any additional requirements.

The purpose of this tutorial is to describe methods to convey lines of text in a FTAM-1 document type.

ISO 8571-2 defines a number of document types for files. One of these document types is FTAM-1. ISO defines the FTAM-1 document type for usage with files that contain unstructured text. A file that has a document type of FTAM-1 consists of one FADU that consists of zero or more character strings. In order to reduce ambiguities it is useful to assume that one string corresponds to one Data Element.

FTAM-1 document type parameters are defined in ISO 8571-2 clause B.1. These parameters are used to define:

the allowed character sets that may be contained in the strings (universal-class-number);

the maximum allowed length of a string (maximum-string-length);

the significance of the boundaries of string (string-significance)

D.2 Document type Parameters

D.2.1 Universal-Class-Number

The universal-class-number parameter determines the character sets that are allowed to be used in a FTAM-1 file. The values of the universal-class-number parameter are ASN.1 types whose definition can be found in ISO 8824. For example, GraphicString, IA5String, and GeneralString are some ASN.1 universal types. The important thing for this discussion is that some string classes allow only graphic characters to be used while other string classes allow both graphic and control characters to be used. (Control characters include "format effector" characters such as carriage return <CR> and line feed <LF>).

D.2.2 Maximum-String-Length

The maximum-string-length parameter determines the maximum number of characters allowed in a string of the FTAM-1 file. It does not determine the maximum number of octets allowed in the string.

GeneralStrings illustrate how the number of octets in a string can differ from the number of characters in a string. GeneralStrings can contain escape sequences that are used for purposes such as invoking different character sets. An escape sequence is considered to be a bit string, not a character string.

Therefore, the combined length of any escape sequences contained in a GeneralString contributes to the number of octets in the GeneralString but does not contribute to the number of characters in the GeneralString.

The length value of the ASN.1 encoding of a character string always reflects the number of octets in the character string. This value will always be greater than or equal to the number of characters in the string. The ASN.1 string must be processed to determine the actual number of characters in the string.

OIW FTAM Phase 2 agreements state that a conformant FTAM implementation must support a maximum-string-length parameter of at least 134 for a FTAM-1 file (see part 9 clause 10). There is no minimum requirement for maximum-string-length in the FTAM phase 3 agreements. The minimum requirement implies that a minimally conformant OIW FTAM responding implementation will not accept a FTAM-1 file whose actual maximum-string-length parameter has a value greater than 134. The relaxation rules for FTAM-1 files allow a FTAM-1 file to be opened for read using a maximum-string-length parameter that is greater than or equal to the value of the maximum-string-length file attribute actually associated with the file, a smaller value is not allowed (see ISO 8571-2 B.1 clause 11.1.1.2). This implies that a minimally conformant OIW FTAM initiating implementation can not read a FTAM-1 file whose actual string length parameter has a value greater than 134.

To increase interoperability, a sending FTAM system should be able to divide a file with string-significance of not-significant into strings of no more than 134 characters. A receiving FTAM system should be able to use the strings to form the file which was sent. If a file has a maximum-string-length associated with it that is greater than 134 interworking will not be possible with a minimally conformant system.

D.2.3 String-Significance

The string-significance parameter determines the significance of the character strings (semantics of string boundaries). Fixed string-significance means that each string contains exactly the number of characters defined by the maximum-string-length parameter. Variable string-significance means that the length of each string is less than or equal to the maximum-string-length parameter. When string-significance is fixed, then maximum-string-length must be present. For string-significance of fixed or variable the boundaries of the character strings are preserved and contribute to the document's semantic. A value of not-significant means that the length of each string is less than or equal to the maximum-string-length parameter and that the boundaries of the character strings are not necessarily preserved when the file is stored and do not contribute to the document's semantics. In this case, string-significance may not be maintained, thus the sender entity explicitly declares that string boundaries have no meaning.

Note the OIW FTAM Phase 2 agreements require the support of only the not-significant value for string-significance. Fixed and variable string-significance are outside the scope of the Phase 2 agreements, but are required in the Phase 3 agreements.

It is in the area of not-significant strings where most interoperability problems have occurred.

> **NOTE -** the difference between variable significance and not-significant significance. If a file has a significance of fixed or variable, it is the responsibility of any storer of the file to "remember" where the boundaries of each character string are located within the file. The storer of a file with a significance of not-significant has no such responsibility. For example, when working with a not-significant file, the sending application may find that 512 byte chunks of data is convenient and useful. The 512 byte size may have no relation to the file layout, but

is easy to read from disk.

D.3 New Line Function

When a sequence of characters are being displayed on a character imaging device, e.g., printer or video display terminal the term "new line function" is used to mean the repositioning of the current character display position one row down and back to column one. A new line function may be implemented in a variety of ways. A UNIX system implements the new line function with a <LF> character (sometimes called <NL>). A MS-DOS system implements the new line function with a <CR><LF> character sequence. A typical word processor will implement a new line function as a "wrap around" function that depends upon a defined page width. A record oriented file system may interpret an end of record condition as implying a new line function.

ISO suggests (see ISO 646 clause 4.1.2.2) that a new line function be accomplished with a <CR><LF> combination. If there is a prior arrangement,e.g., a bilateral agreement, between a sender and a receiver, and only in this case, may a vertical format effector, i.e., a <LF> be used to accomplish a new line function. The OIW FTAM agreements contain no such prior arrangement (see OIW Part 9 clause 10.1.2).

It is strongly suggested that files being sent to a remote FTAM implementation represent the local new line function as a <CR><LF> pair and files received from a remote FTAM implementation have <CR><LF> pairs converted to the local new line function. See D.5 for the reasons for this suggestion.

It is important to realize that a new line function represents a display positioning function and it does not represent anything more that. A new line function is not intended to act as either a string terminator or a string separator.

D.4 Character Strings Versus Lines

A line of characters is generally considered to be a sequence of graphic characters followed by a new line function (or possibly by an end of line condition).

A character string is simply that, a string of characters from one or more character sets. Characters within a string come from allowed character sets. It is the "universal-class-number" parameter defined in ISO 8571-2 B.1 that determines which character sets may be used to compose a string. For example, a GraphicString consists of characters from any graphic character set but may not contain characters from a control character set (it can not contain format effectors); a GeneralString consists of characters from any graphic character set and characters from any control character set (it can contain format effectors).

Text files will be transferred using the Document type FTAM-1. The supported character sets and their recommended line delimiters are:

IA5String (line boundaries via format effectors, preferably <CR><LF>)

GeneralString (i.e. ISO 646 International Reference Version and ISO 8859-1. Line boundaries via format effectors. preferably <CR><LF>)

VisibleString (IA5 String without control characters, line boundaries via Data Element boundaries)

GraphicString (i.e. ISO 646 International Reference Version without control characters and ISO 8859-1, line boundaries via Data Element boundaries)

NOTE - A string is really a language (programming or otherwise) concept. File systems generally have no concept of a string, although a file system, especially a record oriented file system may have some concept of a line.

The standard gives no relation between character string and a line of characters. A character string may contain a portion of a line of characters or it may contain multiple lines of characters. A character string can contain zero, one, or many <CR><LF> pairs. For those character sets which include format effectors, a character string may or may not end with a <CR><LF> pair. In fact, an entire file of character strings may not contain a single <CR><LF> pair, even when those characters are allowed to be used in the character strings.

The following figure is an example of how lines of text could be conveyed using IA5String or GeneralString with string-significance of not-significant.

<table>
<tr><td colspan="2">String-1</td><td colspan="2">String-2</td><td>String-3</td><td>String-4</td><td>String-5</td></tr>
<tr><td>Line-1
<CR><LF></td><td colspan="2">Line-2
<CR><LF></td><td>Line-3
<CR><LF></td><td colspan="2">Line-4
<CR><LF></td><td>Line-5
<CR><LF></td></tr>
</table>

The following figure is an example of how lines of text could be conveyed using VisibleString or GraphicString with string-significance of fixed or variable.

String-1	String-2	String-3	String-4	String-5
Line-1	Line-2	Line-3	Line-4	Line-5

D.5 Mapping FTAM-1 Files to Real Files

The lack of equivalence between a line of characters and a character string can cause implementation problems. It is common for a record oriented file system to store a line of characters as a record. How does such a system decide how large a record to allocate for a line of characters? A line of characters may be contained in a part of one string, one or more strings, or it may actually consist of an entire file. How does such a system identify the end of a line (record)? It must scan the string for a <CR><LF> pair (or end of transmission) and probably remove the <CR><LF> before storing a record. What happens if the line is bigger than the size of the record allocated? The system would likely break the string and store it in the available record size. In this case, the FTAM-1 document type should not be used to transfer this file when the sender is not sensitive to the receiver's limitations.

Another problem can occur when a system whose new line function is implemented by a <CR><LF> pair sends a file to a system whose new line function is implemented by <LF>. For example, a MS-DOS system could send a file that contains <CR><LF> pairs and also contains single <LF> characters to a UNIX system. The UNIX system would likely translate both <CR><LF> and <LF> to UNIX new line functions, i.e. a <LF>. In this case, the FTAM-1 document type should not be used to transfer this type of file.

D.6 Printing or Displaying a File without Format Effectors

There is no relation between a character string and a line of characters (see ISO 8571-2 B.1 clause 7) except when character strings that come from character sets that do not contain format effector characters (for example, VisibleStrings and GraphicStrings) are transferred to a device such as a printer. In this case the end of a string implies the invocation of the device's new line function. This means that, in this case, a string is equivalent to a line.

The rendition of such a file made of character strings belonging to a set that does not contain format effector characters (for example, VisibleString, and GraphicString) to be transferred first to disk and then to a character imaging device might not be equivalent to the rendition of the same file transferred directly to a character imaging device.