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

Output from the June 1991 NIST Workshop for Implementors of OSI

SIG Chair: **Darryl Roberts, UNISYS**
SIG Editor: **Larry Friedman, DEC**

## Foreword

This part of the Stable Implementation Agreements was prepared by the File Transfer, Access and Management Special Interest Group (FTAM SIG) of the National Institute of Standards and Technology (NIST) Workshop for Implementors of Open Systems Interconnection (OSI). See Procedures Manual for Workshop 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. Deleted and replaced text will be strikeout. New and replacement text will be shown as shaded.

# Table of Contents

# Part 9 - ISO File Transfer, Access and Management Phase 2

**NOTE -** See Stable Document for text on this subject

## **Annex A** (normative)

## FTAM Document Types

(See Stable Document.)

# Annex B (normative)

## Constraint Sets

(See Stable Document.)

# Annex C (normative)

## Abstract Syntaxes

(See Stable Document.)

# 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 character string correspond 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.

NIST/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 NIST/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 NIST/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, relaxation to the minimally conformant value is not possible and interworking might not be possible.

### 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 NIST/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 NIST/OIW FTAM agreements contain no such prior arrangement (see NIST/OIW Part 9 clause 10.1.2).

It is strongly suggested that files being sent to a remote FTAM represent the local new line function as a <CR><LF> pair and files received from a remote FTAM 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 delimitation 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 VisibleString or GraphicString with string-significance of fixed or variable.

| String-1 | String-2 | | String-3 | String-4 | String-5 |
|---|---|---|---|---|---|
| Line-1 <CR><LF> | Line-2 <CR><LF> | Line-3 <CR><LF> | Line-4 <CR><LF> | | Line-5 <CR><LF> |

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 |

Note that the minimum requirement of 134 for maximum-string-length (see Maximum-String-length above) does not limit the length of a line when string-significance is not-significant. When string-significance is fixed or variable, a line of characters correspondes to a character string.

## D.5 Mapping Ftam-1 files to Real Files

For a not-significant file, the lack of an equivalence between a line of characters and a character string can cause implementation problems.

## D.6 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. Breaking the string adds a new line function which was not present in the original file.

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> . If the UNIX system then sends the file back to the MS-DOS system the original single <LF> characters will be sent as <CR><LF> pairs.

## D.7 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.