NISTIR 4435

FTAM INTEROPERABILITY TESTS

Carol A. Edgar
Coordinator, NIST

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Gaithersburg, MD 20899

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

NIST

NISTIR 4435

FTAM INTEROPERABILITY TESTS

Carol A. Edgar
Coordinator, NIST

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Gaithersburg, MD 20899

August 1990

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

FTAM Interoperability Tests

Table of Contents

FTAM Interoperability Tests

1. Introduction

This document contains the File Transfer, Access and Management (FTAM) interoperability test suite that was originally developed by the OSINET Technical Committee. OSINET is a regional network that was established to promote OSI through activities related to interoperation testing. This interoperability test suite has been coordinated internationally through OSINET's participation in OSIone, an association of regional OSI networks.

Special acknowledgement is given to John Dempsey, Unisys Corporation, who acted as the editor of the FTAM Interoperability Test Suite for OSINET.

2. Overview

The File Transfer, Access and Management (FTAM) standard enables users to transfer, access and manage large groups of information in a consistent manner within an open environment. This document defines FTAM interoperability tests designed to test the interoperability between FTAM implementations supplied by different vendors. Implementations are assumed to be based on the FTAM International Standard (IS 8571) [1-4].

Interoperability testing tests the ability to perform overall operations between two or more vendor specific implementations. Examples of FTAM operations are send a complete file to a remote system, delete a file from a remote system, and replace a specified data unit in a file on a remote system. The complexity of OSI protocols makes exhaustive testing impractical on both technical and economic grounds. Furthermore, there is no guarantee that a system which has passed a set of interoperability tests will interoperate with other systems or conform to any specification. Rather, passing the tests provides a level of confidence that the system will likely interoperate with other systems and should behave in a consistent manner in representative instances of communication.

The following sections explain the organization of the FTAM interoperability test suite. Developers of OSI profiles (e.g., NIST, TOP) may specify the set of tests that must be successfully completed to meet their requirements. (See Appendix A.) Test participants are free to select additional tests to run from the remaining optional tests. Additional tests may be added in the future to satisfy the requirements of national and international profile developers.

2.1 Test Cases

The tests are written from the point of view of the initiator. The FTAM roles that can be tested are: initiator-sender, initiator-receiver, responder-sender, and responder-receiver.

The FTAM test cases are organized by document types and management attributes. Each test case is defined by:

1. Name
2. Purpose
3. Procedure
4. Pass Condition

The naming convention used to uniquely identify a test *name* is as follows. The first component identifies the document type or management attribute for which the test case is written. Values for the first component are:

FTAM1	-	FTAM-1 Document Type
FTAM2	-	FTAM-2 Document Type
FTAM3	-	FTAM-3 Document Type
NBS6	-	NBS-6 Document Type
NBS7	-	NBS-7 Document Type
NBS8	-	NBS-8 Document Type
NBS9	-	NBS-9 Document Type
LFM	-	Limited File Management
EFM	-	Enhanced File Management

The second component identifies the action that the test case performs. This component is not used for naming management attribute test cases. Values for the possible actions are:

R	-	Read
W	-	Write
RR	-	Multiple Reads
WW	-	Multiple Writes
RW	-	Multiple Reads and Writes

A read action corresponds to the initiator-receiver/responder-sender role. A write action corresponds to the initiator-sender/responder-receiver role. And, read and write corresponds to the initiator-receiver/responder-sender and initiator-sender/responder-receiver role. The single characters 'R' and 'W' are used by the transfer and access service class tests. The "RR", "WW", and "RW" are used by the access service class tests.

The last component identifies the test case number. Test case numbers are used to distinguish between those test cases where the name of the first three components are the same.

Next, the test *purpose* briefly describes the capability being tested.

The test *procedure* defines the steps that the initiator needs to carry out to run the test. The procedure is written from the point of view of the initiator. For example, in the test named FTAM1-R-1, the procedure is:

> Read the complete file **R1R1** and store the received contents into file **A1R1**.

The actual steps described may not correspond directly to a particular FTAM implementation, and added operations may be needed in order to perform the test.

The *pass condition* provides the criteria for determining whether or not a test has successfully passed. Continuing our above example, the pass condition for the test case FTAM1-R-1 is:

> The initiator verifies that the contents of file **A1R1** are identical to the contents defined for file **R1R1**.

To verify that the contents of two files are the same (in this example, files A1R1 and R1R1), their virtual filestore representations must be compared. The contents of the virtual filestore representations of the two files should be identical. However, the contents of the real filestore's particular representations of these files may vary. For example, some systems will use escape sequences, different coded character sets, record delimiters, compaction flags, etc. in a different manner to store files on their system. If a tester compares the real filestores, the tester must take into account how their system represents the virtual files. To inspect a file's contents, the tester can:

1. Display or dump the contents of the files and manually compare the file contents.
2. Compare the two files using utilities (e.g., Unix *diff* command).
3. Understanding how the file is generated, write a program to analyze the received contents of the actual file to see if it contains what was expected. This allows testers not to have to store large files on their systems.
4. Any other method.

In some instances, the pass condition will specify that the responder must verify that a test case has successfully passed.

2.2 Test Files

The contents of the test files are defined in section 3. Each test file is identified by its filename, document type, and file contents. The minimum length of a filename supported by all FTAM systems is eight characters. As such, the names of the test files used by the test cases may seem somewhat cryptic. However, the name of a test file contains valuable information. The naming convention for a test filename follows.

The first character identifies the location of the test file. Values for the first character of a test file can be:

- R - Responder: file is located on the responder's virtual filestore.
- I - Initiator: file is located on the initiator's virtual filestore.
- A - Actual: file is located on the initiator's virtual filestore. This file contains the actual contents received from performing a read operation on a remote file.

The second component identifies the document type of the test file. For limited file management and enhanced file management tests, the second component identifies which management attribute is being tested. Values for the second component of a test file can be:

- 1 - FTAM-1
- 2 - FTAM-2
- 3 - FTAM-3
- 6 - NBS-6
- 7 - NBS-7
- 8 - NBS-8
- LFM - Limited File Management
- EFM - Enhanced File Management

The third component identifies the type of action for which the test file is being used. This component is not used for naming management attribute test files. Values for the third component of a test file can be:

- R - Read
- W - Write
- RR - Multiple Reads
- WW - Multiple Writes
- RW - Multiple Reads and Writes

Files identified by the single characters 'R' and 'W' are used by the transfer and access service class tests. Files identified by "RR", "WW", and "RW" are used by the access service class tests.

The last component of a test file identifies the test case number.

As an example, the test file R1R1 states that this file resides on the responder's virtual filestore (the first 'R'), is used to test an FTAM-1 document type (the first '1'), is a file that is read by the initiator (the second 'R'), and is the first test in the test group (the second '1'). Given the name of a test file, one can deduce the test case that uses the file. In our example, file R1R1 is used by the test named FTAM1-R-1. Notice that the last three non-hyphen characters of the test name match the last three characters of the test filename.

*****

NOTE:
ALL TEST FILES USED BY A TEST CASE MUST EXIST
PRIOR TO RUNNING THE TEST CASE, UNLESS OTHERWISE NOTED.

*****

2.3 Connections

The ability to establish a connection is basic to the operation of all of the FTAM tests, therefore a test to explicitly exercise connection establishment is not proposed. The FTAM functional units, service classes, attribute groups and contents types will be given as required to parameterize the channel for a particular test group. (See Appendix B.) If the required connection parameters for a test can not be provided, the test fails. There is no dependency on the connection state between tests.

A single connection can be used to run one or more test cases if the required connection parameters for each test are supported. Thus, there is no requirement that a new connection be made in order to run each test case.

Some implementations may not provide to an end user the service of connection establishment outside of the context of a file operation. Those implementations which do not perform a simple connection establishment must establish connections prior to any file operations, and therefore their ability to establish a connection will have been tested if they pass the required set of tests.

2.4 Test Suites

The organization of this document defines one set of FTAM interoperability tests that may be grouped differently to satisfy the testing requirements of various organizations.

A test suite specifies the service elements that define a complete group of functions to be tested. For each test suite, the required connection parameters and set of test cases that must be passed need to be clearly stated.

In defining the FTAM interoperability tests, the goals were to write a minimum number of test cases that provide the maximum test coverage in an implementation independent manner. A minimum number of files should be required to exist before testing begins. It is desirable to allow the test cases to run without operator intervention. This will require leaving behind files that are examined either mechanically or manually at the conclusion of testing. There should be no negative or unpredictable side effects produced by running a test suite one or more times on the same files in a directory. This may require having a set up procedure which is run before any testing begins to remove all unwanted files.

3. FTAM Interoperability Tests

3.1 FTAM-1 Test Cases

Transfer Service Class Tests

Name:	FTAM1-R-1
Purpose:	Read an FTAM-1 file containing IA5 character strings.
Procedure:	Read the complete file **R1R1** and store the received contents into file **A1R1**.
Pass Condition:	The initiator verifies that the contents of file **A1R1** are identical to the contents defined for file **R1R1**.

Name:	FTAM1-R-2
Purpose:	Read an FTAM-1 file that contains ISO 8859-1 character strings.
Procedure:	Read the complete file **R1R2** and store the received contents into file **A1R2**.
Pass Condition:	The initiator verifies that the contents of file **A1R2** are identical to the contents defined for file **R1R2**.

Name: FTAM1-R-3

Purpose: Read an FTAM-2 file as an FTAM-1 file. Tests for structural simplification, character relaxation, and string length relaxation of an FTAM-2 file.

Procedure: If the responder supports the FTAM-2 document type, read the complete file **R2R** and store the received contents into file **A1R3.**

Pass Condition: The initiator verifies that the contents of file **A1R3** contains the same sequence of data values as would result from accessing the structured text file in access context UA. The initiator verifies that the characters "carriage return" and "line feed" were added to the end of each string in the virtual file. The resultant virtual file is a single FADU whose contents contains:

10 'O' characters, CR, LF,
20 'S' characters, CR, LF,
30 'I' characters, CR, LF,
40 'F' characters, CR, LF,
50 'T' characters, CR, LF,
60 'A' characters, CR, LF,
70 'M' characters, CR, LF,
60 'T' characters, CR, LF,
50 'E' characters, CR, LF,
40 'S' characters, CR, LF,
30 'T' characters, CR, LF, and
20 'S' characters, CR, LF.

This test is only required if the responder supports the FTAM-2 document type.

Name: FTAM1-W-1

Purpose: Replace an FTAM-1 file containing IA5 character strings.

Procedure: Replace file **R1W1** with file **I1W1.**

Pass Condition: The responder verifies that the contents of file **R1W1** are identical to the contents defined for file **I1W1.**

Name: FTAM1-W-2

Purpose: Replace an FTAM-1 file containing ISO 8859-1 character strings.

Procedure: Replace file **R1W2** with file **I1W2.**

Pass Condition: The responder verifies that the contents of file **R1W2** are identical to the contents defined for file **I1W2.**

Name:	FTAM1-W-3
Purpose:	Extend an FTAM-1 file containing IA5 character strings.
Procedure:	Extend file **R1W3** with the contents of file **I1W3**.
Pass Condition:	The responder verifies that the contents of file **R1W3** are identical to the contents defined by concatenating file **I1W3** to the end of the original file **R1W3**.

Name:	FTAM1-W-4
Purpose:	Extend an FTAM-1 file containing ISO 8859-1 character strings.
Procedure:	Extend file **R1W4** with the contents of file **I1W4**.
Pass Condition:	The responder verifies that the contents of file **R1W4** are identical to the contents defined by concatenating file **I1W4** to the end of the original file **R1W4**.

Access Service Class Tests

Name:	FTAM1-WW-1
Purpose:	Erase an FTAM-1 file.
Procedure:	Erase file **R1WW1**.
Pass Condition:	The responder verifies that file **R1WW1** only contains a root node with an empty data unit.

3.2 FTAM-2 Test Cases

The access context used by the FTAM-2 test cases is Flat All Data Units Access Context (FA).

Transfer Service Class Tests

Name:	FTAM2-R-1
Purpose:	Read the "begin" FADU (the whole file) in an FTAM-2 file.
Procedure:	Read the "begin" FADU in file **R2R** and store the received contents into **A2R1**.
Pass Condition:	The initiator verifies that the contents of file **A2R1** are identical to the contents defined for file **R2R**.

Name:	FTAM2-W-1
Purpose:	Insert an FADU at the end of an FTAM-2 file.
Procedure:	Insert an FADU at the end of file **R2W1**. The data unit contents contains 80 'B' characters.
Pass Condition:	The responder verifies that 80 'B' characters were inserted at the end of file **R2W1**.

Access Service Class Tests

Name: FTAM2-RR-1

Purpose: Perform multiple reads of an FTAM-2 file.

Procedure: The following actions operate on file **R2R**. The received contents of these operations are stored into file **A2RR1**. The actions are:

1. Locate "next" FADU.
2. Read "next" FADU.
3. Read "next" FADU.
4. Locate "first" FADU.
5. Read "next" FADU.
6. Read "first" FADU.
7. Read "begin" FADU.

Pass Condition: The initiator verifies that file **A2RR1** contains:

20 'S' characters,
30 'I' characters,
20 'S' characters,
10 'O' characters,
10 'O' characters,
20 'S' characters,
30 'I' characters,
40 'F' characters,
50 'T' characters,
60 'A' characters,
70 'M' characters,
60 'T' characters,
50 'E' characters,
40 'S' characters,
30 'T' characters, and
20 'S' characters.

Name: FTAM2-WW-1
Purpose: Perform multiple writes to an FTAM-2 file.
Procedure: The following actions operate on file **R2WW1**. The actions are:

1. Insert an FADU that contains 60 'A' characters.
2. Insert an FADU that contains 80 'B' characters.
3. Insert an FADU that contains 40 'C' characters.

Pass Condition: The responder verifies that the contents of file **R2WW1** are identical to the contents of file **R2R** followed by 60 'A' characters, 80 'B' characters, and 40 'C' characters.

Name: FTAM2-WW-2
Purpose: Erase an FTAM-2 file.
Procedure: Erase "begin" FADU of file **R2WW2**.
Pass Condition: The responder verifies that file **R2WW2** only contains a root node with no data units.

Name: FTAM2-RW-1

Purpose: Perform multiple reads and writes to an FTAM-2 file.

Procedure: The following actions operate on file **R2RW1**. The received contents of these operations are stored into file **A2RW1**. The actions are:

1. Insert an FADU that contains 80 'U' characters.
2. Insert an FADU that contains 80 'C' characters.
3. Locate "begin" FADU.
4. Read "next" FADU.
5. Locate "next" FADU.
6. Read "next" FADU.
7. Locate "first" FADU.
8. Read "next" FADU.
9. Insert an FADU that contains 80 'L' characters.
10. Insert an FADU that contains 80 'A' characters.
11. Read "begin" FADU.

Pass Condition: The initiator verifies that the contents of file **A2RW1** contains:

10 'O' characters,
30 'I' characters,
20 'S' characters,
10 'O' characters,
20 'S' characters,
30 'I' characters,
40 'F' characters,
50 'T' characters,
60 'A' characters,
70 'M' characters,
60 'T' characters,
50 'E' characters,
40 'S' characters,
30 'T' characters,
20 'S' characters,
80 'U' characters,
80 'C' characters,
80 'L' characters, and
80 'A' characters.

3.3 FTAM-3 Test Cases

Transfer Service Class Tests

Name:	FTAM3-R-1
Purpose:	Read an FTAM-3 file.
Procedure:	Read file **R3R1** and store its contents into file **A3R1**.
Pass Condition:	The initiator verifies that the contents of file **A3R1** are identical to the contents defined for file **R3R1**.

Name:	FTAM3-R-2
Purpose:	Read a large FTAM-3 file.
Procedure:	Read file **R3R2** and store its contents into file **A3R2**.
Pass Condition:	The initiator verifies that the contents of file **A3R2** are identical to the contents defined for file **R3R2**.

Name:	FTAM3-R-3
Purpose:	Read a file whose filename is eight characters in length and contains numeric values.
Procedure:	Read file **R1234567** and store its contents into file **A1234567**.
Pass Condition:	The initiator verifies that the contents of file **A1234567** are identical to the contents defined for file **R3R1**.

Name:	FTAM3-R-4
Purpose:	Attempt to read a file that does not exist.
Procedure:	Read file **RXYZ** and store its contents into file **AXYZ**. The file **RXYZ** must not exist on the responder's filestore.
Pass Condition:	The initiator verifies that an error indicating that the file does not exist is returned.

Name:	FTAM3-R-5
Purpose:	Check error handling when an invalid filename format is used.
Procedure:	Attempt to read a file using an invalid filename format for the test partners file system. This may include case sensitivity on some systems. Details will need to be obtained from the test partner.
Pass Condition:	The initiator verifies that a suitable error message is returned. This can not be better defined as it will be system specific. The responder verifies that no adverse effects result from this request.

Name:	FTAM3-R-6
Purpose:	Check the close down and error handling of a lost connection.
Procedure:	Request a file transfer, and during this drop/crash the FTAM regime by some means (or get the test partner to drop the connection). The connection can be lost at any level.
Pass Condition:	The initiator and responder check that there is no adverse effect on their FTAM system. Recovery is not necessary, just graceful handling of the situation.

Name:	FTAM3-R-7
Purpose:	Permitted action prevents reading file.
Procedure:	Read file **R3R7** and store its contents into file **A3R7**.
Pass Condition:	The initiator verifies that an error indicating that the file does not allow its contents to be read is returned.

Name:	FTAM3-W-1
Purpose:	Replace an FTAM-3 file.
Procedure:	Replace file **R3W1** with file **I3W1**.
Pass Condition:	The responder verifies that the contents of file **R3W1** are identical to the contents defined for file **I3W1**.

Name:	FTAM3-W-2
Purpose:	Replace a small FTAM-3 file with a large FTAM-3 file.
Procedure:	Replace file **R3W2** with file **I3W2**.
Pass Conditions:	The responder verifies that the contents of file **R3W2** are identical to the contents defined for file **I3W2**.

Name:	FTAM3-W-3
Purpose:	Replace a large FTAM-3 file with a small FTAM-3 file.
Procedure:	Replace file **R3W3** with file **I3W3**.
Pass Condition:	The responder verifies that the contents of file **R3W3** are identical to the contents defined for file **I3W3**. The responder verifies the filesize attribute and/or verifies that no extra information exists.

Name:	FTAM3-W-4
Purpose:	Extend an FTAM-3 file.
Procedure:	Extend file **R3W4** with the contents of file **I3W4**.
Pass Condition:	The responder verifies that the contents of file **R3W4** are identical to the contents defined by concatenating file **I3W4** to the end of the original file **R3W4**.

Name:	FTAM3-W-5
Purpose:	Permitted actions allow extend, but not replace.
Procedure:	Replace file **R3W5** with the contents of file **I3W5**. Extend file **R3W5** with the contents of file **I3W5**.
Pass Condition:	The initiator verifies that an error indicating that the file does not allow its contents to be replaced is returned. The responder verifies that the contents of file **R3W5** are identical to the contents defined by concatenating file **I3W5** to the end of the original file **R3W5.**

Access Service Class Tests

Name:	FTAM3-WW-1
Purpose:	Erase an FTAM-3 file.
Procedure:	Erase file **R3WW1.**
Pass Condition:	The responder verifies that file **R3WW1** only contains a root node with an empty data unit.

3.4 NBS-6 Test Cases

The access context used by the NBS-6 test cases is Flat All Data Units Access Context (FA).

Transfer Service Class Tests

Name:	NBS6-R-1
Purpose:	Read "begin" FADU in an NBS-6 file.
Procedure:	Read "begin" FADU in file **R6R** and store the result in file **A6R1**.
Pass Condition:	The initiator verifies that the contents of file **A6R1** are identical to the contents defined for file **R6R**.

Name:	NBS6-R-2
Purpose:	Read "begin" FADU in an NBS-6 file.
Procedure:	Read "begin" FADU in file **R6R2** and store the result in file **A6R2**.
Pass Condition:	The initiator verifies that the contents of file **A6R2** are identical to the contents defined for file **R6R2**.

Name:	NBS6-W-1
Purpose:	Insert an FADU at the "end" of an NBS-6 file.
Procedure:	Insert an FADU at the end of file **R6W1**. The data elements contain the following values: "Warsaw", "Poland", 294, '100100110'B, '126'H, 22.0, "inches", and FALSE.
Pass Condition:	The responder verifies that the above data unit was inserted at the end of file **R6W1**.

Name:	NBS6-W-2
Purpose:	Insert an FADU at the "end" of an NBS-6 file.
Procedure:	Insert an FADU at the end of file **R6W2**. The data elements contain the following values: "October 31, 1992 5:35 p.m.", "19921031173500.0", null, and "9210311735Z".
Pass Condition:	The responder verifies that the above data unit was inserted at the end of file **R6W2**.

Access Service Class Tests

Name: NBS6-RR-1

Purpose: Perform multiple reads of an NBS-6 file.

Procedure: The following actions operate on file **R6R**. The received contents of these operations are stored into file **A6RR1**. The actions are:

1. Locate "next" FADU.
2. Read "next" FADU.
3. Read "next" FADU.
4. Locate "first" FADU.
5. Read "next" FADU.
6. Read "first" FADU.
7. Read "begin" FADU.

Pass Condition: The initiator verifies that file **A6RR1** contains:

"London"	"England"	149	'10010101'B	'95'H	22.9	"inches"	FALSE
"Manila"	"Philippines"	49	'110001'B	'31'H	82.0	"inches"	TRUE
"London"	"England"	149	'10010101'B	'95'H	22.9	"inches"	FALSE
"Berlin"	"Germany"	187	'10111011'B	'66'H	23.1	"inches"	FALSE
"Berlin"	"Germany"	187	'10111011'B	'66'H	23.1	"inches"	FALSE
"London"	"England"	149	'10010101'B	'95'H	22.9	"inches"	FALSE
"Manila"	"Philippines"	49	'110001'B	'31'H	82.0	"inches"	TRUE
"Paris"	"France"	164	'10100100'B	'A4'H	22.3	"inches"	FALSE
"Sydney"	"Australia"	62	'111110'B	'3E'H	46.5	"inches"	FALSE
"Tokyo"	"Japan"	19	'10011'B	'13'H	61.6	"inches"	TRUE

Name: NBS6-WW-1
Purpose: Perform multiple writes to an NBS-6 file.
Procedure: The following actions operate on file **R6WW1**. The actions are:

1. Insert an FADU that contains: "Athens", "Greece", 351, '101011111'B, '15F'H, 15.8, "inches", and FALSE.
2. Insert an FADU that contains: "Rome", "Italy", 377, '101111001'B, '179'H, 29.5, "inches", and FALSE
3. Insert an FADU that contains: "Bangkok", "Thailand", 53, '110101'B, '35'H, 57.8, "inches", and TRUE.

Pass Condition: The responder verifies that the contents of file **R6WW1** is identical to the contents of file **R6R** followed by:

"Athens"	"Greece"	351	'101011111'B	'15F'H	15.8	"inches"	FALSE
"Rome"	"Italy"	377	'101111001'B	'179'H	29.5	"inches"	FALSE
"Bangkok"	"Thailand"	53	'110101'B	'35'H	57.8	"inches"	TRUE

Name: NBS6-WW-2
Purpose: Erase an NBS-6 file.
Procedure: Erase "begin" FADU of file **R6WW2**.
Pass Condition: The responder verifies that file **R6WW2** only contains a root node with an empty data unit.

Name: NBS6-RW-1

Purpose: Perform multiple reads and writes to an NBS-6 file.

Procedure: The following actions operate on file **R6RW1**. The received contents of these operations are stored into file **A6RW1**. The actions are:

1. Insert an FADU that contains: "Oslo", "Norway", 308, '100110100'B, '134'H, 26.9, "inches", and FALSE.
2. Insert an FADU that contains: "Dublin", "Ireland", 155, '10011011'B, '96'H, 29.7, "inches", and FALSE.
3. Locate "first" FADU.
4. Read "next" FADU.
5. Locate "begin" FADU.
6. Read "next" FADU.
7. Insert an FADU that contains: "Cairo", "Egypt", 381, '101111101'B, '17D'H, 1.1, "inches", and FALSE.
8. Read "begin" FADU.

Pass Condition: The initiator verifies that the contents of file **A6RW1** contains:

"London"	"England"	149	'10010101'B	'95'H	22.9	"inches"	FALSE
"Berlin"	"Germany"	187	'10111011'B	'66'H	23.1	"inches"	FALSE
"Berlin"	"Germany"	187	'10111011'B	'66'H	23.1	"inches"	FALSE
"London"	"England"	149	'10010101'B	'95'H	22.9	"inches"	FALSE
"Manila"	"Philippines"	49	'110001'B	'31'H	82.0	"inches"	TRUE
"Paris"	"France"	164	'10100100'B	'A4'H	22.3	"inches"	FALSE
"Sydney"	"Australia"	62	'111110'B	'3E'H	46.5	"inches"	FALSE
"Tokyo"	"Japan"	19	'10011'B	'13'H	61.6	"inches"	TRUE
"Oslo"	"Norway"	308	'100110100'B	'134'H	26.9	"inches"	FALSE
"Dublin"	"Ireland"	155	'10011011'B	'96'H	29.7	"inches"	FALSE
"Cairo"	"Egypt"	381	'101111101'B	'17D'H	1.1	"inches"	FALSE

3.5 NBS-7 Test Cases

The access context used by the NBS-7 test cases is Flat All Data Units Access Context (FA).

Transfer Service Class Tests

Name:	NBS7-R-1
Purpose:	Read the "begin" FADU in an NBS-7 file.
Procedure:	Read the "begin" FADU (the whole file) in file **R7R** store the received contents into file **A7R1**.
Pass Condition:	The initiator verifies that the contents of file **A7R1** is identical to the contents defined for file **R7R**.

Name:	NBS7-R-2
Purpose:	Read the "first" FADU in an NBS-7 file.
Procedure:	Read the "first" FADU in file **R7R**.
Pass Condition:	The initiator verifies that the following data unit is received: 100, "9.83s", "Ben Johnson", "Canada", "19870830151500.0", and "8708301515Z".

Name:	NBS7-R-3
Purpose:	Read the "last" FADU in an NBS-7 file.
Procedure:	Read the "last" FADU in file **R7R**.
Pass Condition:	The initiator verifies that the following data unit is received: 3000, "7m, 32.1s", "Henry Rono", "Kenya", "19780727181800.0", and "7807271818Z".

Name:	NBS7-R-4
Purpose:	Read the "current" FADU in an NBS-7 file.
Procedure:	Read "current" FADU in file **R7R** and store the received contents into file **A7R4**.
Pass Condition:	The initiator verifies that the contents of file **A7R4** is identical to the contents defined for file **R7R**.

Name:	NBS7-R-5
Purpose:	Read the "next" FADU in an NBS-7 file.
Procedure:	Read "next" FADU in file **R7R**.
Pass Condition:	The initiator verifies that the following data unit is received: 100, "9.83s", "Ben Johnson", "Canada", "19870830151500.0", and "8708301515Z".

Name:	NBS7-R-6
Purpose:	Read the "previous" FADU in an NBS-7 file.
Procedure:	Read the "previous" FADU in file **R7R**.
Pass Condition:	The initiator verifies that no characters are received.

Name:	NBS7-R-7
Purpose:	Read an FADU identified by traversal number.
Procedure:	Read the FADU whose traversal number is 3 in file **R7R**.
Pass Condition:	The initiator verifies that the following data unit is received: 400, "47.60s", "Marita Koch", "East Germany", "19851006171700.0", and "8510061717Z".

Name:	NBS7-R-8
Purpose:	Read the last FADU identified by traversal number.
Procedure:	Read the FADU whose traversal number is 5 in file **R7R**.
Pass Condition:	The initiator verifies that the following data unit is received: 3000, "7m, 32.1s", "Henry Rono", "Kenya", "19780727191900.0", and "7807271919Z".

Name:	NBS7-R-9
Purpose:	Read the "begin" FADU in an NBS-7 file.
Procedure:	Read the "begin" FADU (the whole file) in file **R7R9** store the received contents into file **A7R9**.
Pass Condition:	The initiator verifies that the contents of file **A7R9** is identical to the contents defined for file **R7R9**.

Name:	NBS7-W-1
Purpose:	Insert an FADU at the "end" of an NBS-7 file.
Procedure:	Insert a new FADU at the "end" of file **R7W1**. The data unit contents contains: 400, "43.29s", "Butch Reynolds", "United States", "19880822092500.0", and "8808220925Z".
Pass Condition:	The responder verifies that the above data unit was inserted at the end of file **R7W1**.

Name:	NBS7-W-2
Purpose:	Insert an FADU at the end of an NBS-7 file whose position is identified by traversal number.
Procedure:	Insert a new FADU into file **R7W2** after the FADU whose traversal number is 50. The data unit contents contains: 100, "10.49s", "Florence Griffith-Joyner", "United States", "19880915012300.0", and "8809150123Z".
Pass Condition:	The responder verifies that the above data unit was inserted at the end of file **R7W2**.

Name:	NBS7-W-3
Purpose:	Replace the "first" FADU in an NBS-7 file.
Procedure:	Replace the "first" FADU in file **R7W3** with a new data unit that contains: 400, "43.29s", "Butch Reynolds", "United States", "19880822092500.0", and "8808220925Z".
Pass Condition:	The responder verifies that file **R7W3** contains:

400	"43.29s"	"Butch Reynolds"	"United States"	"19880822092500.0"	"8808220925Z"
200	"19.72s"	"Pietro Mennea"	"Italy"	"19790917161600.0"	"7909171616Z"
400	"47.60s"	"Marita Koch"	"East Germany"	"19851006171700.0"	"8510061717Z"
1000	"2m, 12.18s"	"Sebastian Coe"	"Great Britain"	"19810711181800.0"	"8107111818Z"
3000	"7m, 32.1s"	"Henry Rono"	"Kenya"	"19780727191900.0"	"7807271919Z"

Name:	NBS7-W-4
Purpose:	Replace the "next" FADU in an NBS-7 file.
Procedure:	Replace the "next" FADU in file **R7W5** with a new data unit that contains: 400, "43.29s", "Butch Reynolds", "United States", "19880822092500.0", and "8808220925Z".
Pass Condition:	See pass condition for test case NBS7-W-5.

Name: NBS7-W-5

Purpose: Replace the "last" FADU in an NBS-7 file.

Procedure: Replace the "last" FADU in file **R7W5** with a new data unit that contains 1500, "3m, 29.46s", "Said Aouita", "Morocco", "19850823151500.0", and "8508231515Z".

Pass Condition: The responder verifies that file **R7W5** contains:

400	"43.29s"	"Butch Reynolds"	"United States"	"19880822092500.0"	"8808220925Z"
200	"19.72s"	"Pietro Mennea"	"Italy"	"19790917161600.0"	"7909171616Z"
400	"47.60s"	"Marita Koch"	"East Germany"	"19851006171700.0"	"8510061717Z"
1000	"2m, 12.18s"	"Sebastian Coe"	"Great Britain"	"19810711181800.0"	"8107111818Z"
1500	"3m, 29.46s"	"Said Aouita"	"Morocco"	"19850823151500.0"	"8508231515Z".

The resultant file is from running test cases NBS7-W-4 through NBS7-W-5 in any order.

Name: NBS7-W-6

Purpose: Replace the "previous" FADU in an NBS-7 file.

Procedure: Replace the "previous" FADU in file **R7W6** with a new data unit that contains: 400, "43.29s", "Butch Reynolds", "United States", "19880822092500.0", and "8808220925Z". This action should fail.

Pass Condition: The responder verifies that the contents of file **R7W6** are identical to the contents defined for file **R7R**.

Name: NBS7-W-7

Purpose: Replace the first FADU identified by traversal number.

Procedure: Replace the first FADU identified by traversal number 1 in file **R7W10** with a new data unit that contains: 400, "43.29s", "Butch Reynolds", "United States", "19880822092500.0", and "8808220925Z".

Pass Condition: See pass condition for test case NBS7-W-10.

Name: NBS7-W-8

Purpose: Replace the "current" FADU in an NBS-7 file.

Procedure: Replace the "current" FADU in file **R7W10** with a new data unit that contains: 100, "10.49s", "Florence Griffith-Joyner", "United States", "19880915012300.0", and "8809150123Z". This action should fail.

Pass Condition: See pass condition for test case NBS7-W-10.

Name: NBS7-W-9
Purpose: Replace an FADU identified by traversal number.
Procedure: Replace the FADU identified by traversal number 2 in file **R7W10** with a new data unit that contains: 1500, "3m, 29.46s", "Said Aouita", "Morocco", "19850823151500.0", and "8508231515Z".
Pass Condition: See pass condition for test case NBS7-W-10.

Name: NBS7-W-10
Purpose: Replace the last FADU identified by traversal number.
Procedure: Replace the FADU identified by traversal number 5 in file **R7W10** with a new data unit contains: 2000, "4m, 50.81s", "Said Aouita", "Morocco", "19870716202000.0", and "8707162020Z".
Pass Condition: The responder verifies that file **R7W10** contains:

400	"43.29s"	"Butch Reynolds"	"United States"	"19880822092500.0"	"8808220925Z"
1500	"3m, 29.46s"	"Said Aouita"	"Morocco"	"19850823151500.0"	"8508231515Z"
400	"47.60s"	"Marita Koch"	"East Germany"	"19851006171700.0"	"8510061717Z"
1000	"2m, 12.18s"	"Sebastian Coe"	"Great Britain"	"19810711181800.0"	"8107111818Z"
2000	"4m, 50.81s"	"Said Aouita"	"Morocco"	"19870716202000.0"	"8707162020Z"

The resultant file is from running test cases NBS7-W-7 through NBS7-W-10 in any order.

Name: NBS7-W-11
Purpose: Replace the "begin" FADU in an NBS-7 file.
Procedure: Replace the "begin" FADU (the whole file) in file **R7W11** with:

400	"47.60s"	"Marita Koch"	"East Germany"	"19851006171700.0"	"8510061717Z"
400	"43.29s"	"Butch Reynolds"	"United States"	"19880822092500.0"	"8808220925Z"
1000	"2m, 12.18s"	"Sebastian Coe"	"Great Britain"	"19810711181800.0"	"8107111818Z"

Pass Condition: The responder verifies that file **R7W11** contains:

400	"47.60s"	"Marita Koch"	"East Germany"	"19851006171700.0"	"8510061717Z"
400	"43.29s"	"Butch Reynolds"	"United States"	"19880822092500.0"	"8808220925Z"
1000	"2m, 12.18s"	"Sebastian Coe"	"Great Britain"	"19810711181800.0"	"8107111818Z"

Name: NBS7-W-12

Purpose: Insert an FADU at the "end" of an NBS-7 file.

Procedure: Insert a new FADU at the "end" of file **R7W12**. The data unit contents contains: "South America", 69, '1000101'B, '45'H, null, 11.9, 5.6 and FALSE.

Pass Condition: The responder verifies that the above data unit was inserted at the end of file **R7W12**.

Access Service Class Tests

Name: NBS7-RR-1

Purpose: Perform multiple reads of an NBS-7 file.

Procedure: The following actions operate on file **R7RR1**. The received contents of these operations are stored into file **A7RR1**. The actions are:

1. Locate "end" FADU.
2. Read "previous" FADU.
3. Locate "first" FADU.
4. Read "current" FADU.
5. Locate "next" FADU.
6. Read "current" FADU.
7. Locate "next" FADU.
8. Locate "next" FADU.
9. Read "next" FADU.
10. Locate "last" FADU.
11. Read "current" FADU.
12. Locate "previous" FADU.
13. Read "current" FADU.

Pass Condition: The initiator verifies that the contents of file **A7RR1** contains:

"Spanish"	null	320
"Arabic"	null	187
"English"	null	431
"German"	null	118
"Spanish"	null	320
"Russian"	null	289

Name: NBS7-RR-2

Purpose: Perform multiple reads of an NBS-7 file.

Procedure: The following actions operate on file **R7RR1**. The received contents of these operations are stored into file **A7RR2**. The actions are:

1. Locate FADU whose traversal number is 7.
2. Read "current" FADU.
3. Locate "previous" FADU.
4. Read "current" FADU.
5. Locate FADU whose traversal number is 1.
6. Read "current" FADU.
7. Locate "next" FADU.
8. Read "current" FADU.
9. Locate FADU whose traversal number is 12.
10. Read "current" FADU.
11. Locate "current" FADU.

Pass Condition: The initiator verifies that the contents of file **A7RR2** contains:

"Japanese"	null	124
"Italian"	null	63
"Arabic"	null	187
"English"	null	431
"Spanish"	null	320

Name: NBS7-WW-1

Purpose: Perform multiple writes to an NBS-7 file.

Procedure: The following actions operate on file **R7WW1**. The actions are:

1. Insert an FADU that contains: Polish, null, 42.
2. Insert an FADU that contains: Tagalog, null, 33.
3. Insert an FADU that contains: Min, null, 45.
4. Locate "first" FADU.
5. Replace current FADU with an FADU that contains: Hebrew, null, 4.
6. Locate "next" FADU.
7. Replace current FADU with an FADU that contains: Swati, null, 1.
8. Locate "last" FADU.
9. Replace current FADU with an FADU that contains: Swedish, null, 9.
10. Locate "previous" FADU.
11. Replace current FADU with an FADU that contains: Hindi, null, 325.
12. Locate FADU whose traversal number is 6.
13. Replace current FADU with an FADU that contains: Bengali, null, 178.
14. Locate FADU whose traversal number is 100.
15. Insert an FADU that contains: Thai, null, 46.

Pass Condition: The responder verifies that the contents of file **R7WW1** contains:

"Hebrew"	null	4
"Swati"	null	1
"Esperanto"	null	2
"French"	null	117
"German"	null	118
"Bengali"	null	178
"Japanese"	null	124
"Korean"	null	68
"Mandarin"	null	825
"Portuguese"	null	169
"Russian"	null	289
"Spanish"	null	320
"Polish"	null	42
"Hindi"	null	325
"Swedish"	null	9
"Thai"	null	46

Name: NBS7-WW-2
Purpose: Erase an NBS-7 file.
Procedure: Erase "begin" FADU of file **R7WW2**.
Pass Condition: The responder verifies that file **R7WW2** only contains a root node with an empty data unit.

Name: NBS7-WW-3
Purpose: Replace an entire NBS-7 file.
Procedure: Locate "begin" of file **R7WW3**. Replace file **R7WW3** with:

"Japanese"	null	124
"Korean"	null	68
"Mandarin"	null	825

Pass Condition: The responder verifies that file **R7WW3** contains:

"Japanese"	null	124
"Korean"	null	68
"Mandarin"	null	825

Name: NBS7-RW-1

Purpose: Perform multiple reads and writes to an NBS-7 file.

Procedure: The following actions operate on file **R7RW1**. The received contents of these operations are stored into file **A7RW1**. The actions are:

1. Locate FADU whose traversal number is 7.
2. Read "current" FADU.
3. Insert an FADU that contains: Finnish, null, 5.
4. Insert an FADU that contains: Greek, null, 11.
5. Read "previous" FADU.
6. Locate "first" FADU.
7. Read "current" FADU.
8. Insert an FADU that contains: Zulu, null, 7.
9. Insert an FADU that contains: Rundi, null, 5.
10. Locate "previous" FADU.
11. Read "previous" FADU.
12. Read "begin" FADU.

Pass Condition: The initiator verifies that the contents of file **A7RW1** contains:

"Japanese"	null	124
"Greek"	null	11
"Arabic"	null	187
"Zulu"	null	7
"Arabic"	null	187
"English"	null	431
"Esperanto"	null	2
"French"	null	117
"German"	null	118
"Italian"	null	63
"Japanese"	null	124
"Korean"	null	68
"Mandarin"	null	825
"Portuguese"	null	169
"Russian"	null	289
"Spanish"	null	320
"Finish"	null	5
"Greek"	null	11
"Zulu"	null	7
"Rundi"	null	5

Name: NBS7-RW-2

Purpose: Create holes in an NBS-7 file.

Procedure: The following actions operate on file **R7RW2**. The received contents of these operations are stored into file **A7RW2**. The actions are:

1. Locate FADU whose traversal number is 3.
2. Replace current FADU with an FADU with a data unit of length 0 (or without a data unit).
3. Locate FADU whose traversal number is 8.
4. Replace current FADU with an FADU with a data unit of length 0 (or without a data unit).
5. Locate FADU whose traversal number is 12.
6. Replace current FADU with an FADU with a data unit of length 0 (or without a data unit).
7. Read "begin" FADU.

Pass Condition: The initiator verifies that the contents of file **A7RW2** contains:

"Arabic"	null	187
"English"	null	431
"French"	null	117
"German"	null	118
"Italian"	null	63
"Japanese"	null	124
"Mandarin"	null	825
"Portuguese"	null	169
"Russian"	null	289

Name: NBS7-RW-3

Purpose: Fill holes in an NBS-7 file.

Procedure: The following actions operate on file **R7RW3**. The received contents of these operations are stored into file **A7RW3**. The actions are:

1. Locate FADU whose traversal number is 3.
2. Replace current FADU with an FADU that contains: Nepali, null, 12.
3. Locate FADU whose traversal number is 7.
4. Replace "next" FADU with an FADU that contains: Persian, null, 31.
5. Read "begin" FADU.

Pass Condition: The initiator verifies that the contents of file **A7RW3** contains:

"Arabic"	null	187
"English"	null	431
"Nepali"	null	12
"French"	null	117
"German"	null	118
"Italian"	null	63
"Japanese"	null	124
"Persian"	null	31
"Mandarin"	null	825
"Portuguese"	null	169
"Russian"	null	289

3.6 NBS-8 Test Cases

The access context used by the NBS-8 test cases is Flat All Data Units Access Context (FA).

Transfer Service Class Tests

Name:	NBS8-R-1
Purpose:	Read the "begin" FADU.
Procedure:	Read the "begin" FADU (the whole file) in file **R8R** and store the received contents into file **A8R1**.
Pass Condition:	The initiator verifies that the contents of file **A8R1** are identical to the contents defined for file **R8R**.

Name:	NBS8-R-2
Purpose:	Read the "current" FADU.
Procedure:	Read "current" FADU in file **R8R**.
Pass Condition:	The initiator verifies that no data elements are received.

Name:	NBS8-R-3
Purpose:	Read the "next" FADU.
Procedure:	Read "next" FADU in **R8R**.
Pass Condition:	The initiator verifies that the following data elements are received: "U.S.", "9638", "10714", and 8682.

Name:	NBS8-R-4
Purpose:	Read the "previous" FADU.
Procedure:	Read "previous" FADU in file **R8R**.
Pass Condition:	The initiator verifies that no data elements are received.

Name:	NBS8-R-5
Purpose:	Read an FADU by its node sequence.
Procedure:	Read FADU "Sydney" in file **R8R** by its node name.
Pass Condition:	The initiator verifies that the following data elements are received: "Australia", "3396", "3708", and 10047.

Name:	NBS8-R-6
Purpose:	Read the first FADU by its node sequence.
Procedure:	Read FADU "Los Angeles" in file **R8R** by its node name.
Pass Condition:	The initiator verifies that the following data elements are received: "U.S.", "9638", "10714", and 8682.

Name:	NBS8-R-7
Purpose:	Read the last FADU by its node sequence.
Procedure:	Read FADU "Tokyo-Yokahama" in file **R8R** by its node name.
Pass Condition:	The initiator verifies that the following data elements are received : "Japan", "25434", "29971", and 23356.

Name:	NBS8-R-8
Purpose:	Read the "begin" FADU.
Procedure:	Read the "begin" FADU (the whole file) in file **R8R8** and store the received contents into file **A8R8**.
Pass Condition:	The initiator verifies that the contents of file **A8R8** are identical to the contents defined for file **R8R8**.

Name:	NBS8-W-1
Purpose:	Insert a new FADU.
Procedure:	Insert a new FADU in file **R8W4** identified as node "New York" and whose data elements contain: "U.S.", "14598", "14648", and 11458.
Pass Condition:	See pass condition for test case NBS8-W-4.

Name:	NBS8-W-2
Purpose:	Insert a new FADU at the beginning.
Procedure:	Insert a new FADU in file **R8W4** identified as node "Hong Kong" and whose data elements contain: "British Colony", "5415", "5956", and 270750.
Pass Condition:	See pass condition for test case NBS8-W-4.

Name:	NBS8-W-3
Purpose:	Insert a new FADU at the end.
Procedure:	Insert a new FADU in file **R8W4** identified as node "Toronto" and whose data elements contain: "Canada", "2972", "3296", and 19300.
Pass Condition:	See pass condition for test case NBS8-W-4.

Name: NBS8-W-4

Purpose: Insert two FADUs with the same FADU identifier.

Procedure: Insert the following two FADUs into file **R8W4**: a new FADU identified as node "Moscow" and whose data elements contain "USSR", "9873", "11121", and 26050, and a new FADU which is also identified as node "Moscow" and whose data elements contain: "Soviet", "Union", "1234", and 100.

Pass Condition: The responder verifies that file **R8W4** contains:

Node Name	Contents			
"Hong Kong"	"British Colony"	"5415"	"5956"	270750
"Los Angeles"	"U.S."	"9638"	"10714"	8682
"Manchester"	"U.K."	"4151"	"3827"	11627
"Mexico City"	"Mexico"	"16900"	"27872"	32377
"Moscow"	"USSR"	"9873"	"11121"	26050
"Moscow"	"Soviet"	"Union"	"1234"	100
"New York"	"U.S."	"14598"	"14648"	11458
"Singapore"	"Singapore"	"2556"	"2913"	32769
"Sydney"	"Australia"	"3396"	"3708"	10047
"Tokyo-Yokahama"	"Japan"	"25434"	"29971"	23356
"Toronto"	"Canada"	"2972"	"3296"	19300

The resultant file is from running test cases NBS8-W-1 through NBS8-W-4 in any order.

Name: NBS8-W-5

Purpose: Replace the "begin" FADU.

Procedure: Replace the "begin" FADU of file **R8W5** (the complete file) with a new file whose FADU node names are: "Bombay", "Cairo", "Lima", "Lima", and "Milan". The contents of this file contains the following data elements:

"India"	"10137"	"15357"	10670
"Egypt"	"8595"	"12500"	82644
"Peru"	"5447"	"9241"	45392
"Inca"	"Empire"	"Andes"	100
"Italy"	"4635"	"4839"	13474

Pass Condition: The responder verifies that file **R8W5** contains:

Node Name	Contents			
"Bombay"	"India"	"10137"	"15357"	10670
"Cairo"	"Egypt"	"8595"	"12500"	82644
"Lima"	"Peru"	"5447"	"9241"	45392
"Lima"	"Inca"	"Empire"	"Andes"	100
"Milan"	"Italy"	"4635"	"4839"	13474

Name: NBS8-W-6

Purpose: Replace "current" FADU.

Procedure: Replace the "current" FADU in file **R8W6** with a new FADU identified as node "Sao Paolo" and whose data elements contain: "Brazil", "14910", "25354", and 33060.

Pass Condition: The responder verifies that file **R8W6** contains:

Node Name	Contents			
"Los Angeles"	"U.S."	"9638"	"10714"	8682
"Manchester"	"U.K."	"4151"	"3827"	11627
"Mexico City"	"Mexico"	"16900"	"27872"	32377
"Sao Paolo"	"Brazil"	"14910"	"25354"	33060
"Singapore"	"Singapore"	"2556"	"2913"	32769
"Sydney"	"Australia"	"3396"	"3708"	10047
"Tokyo-Yokahama"	"Japan"	"29971"	"29971"	23356

Name: NBS8-W-7

Purpose: Replace "next" FADU's data contents only.

Procedure: Replace the data unit contents of the "next" FADU in file **R8W7**. The new data elements contain: "California", "United", "States", and 100.

Pass Condition: The responder verifies that file **R8W7** contains:

Node Name	Contents			
"Los Angeles"	"California"	"United"	"States"	100
"Manchester"	"U.K."	"4151"	"3827"	11627
"Mexico City"	"Mexico"	"16900"	"27872"	32377
"Singapore"	"Singapore"	"2556"	"2913"	32769
"Sydney"	"Australia"	"3396"	"3708"	10047
"Tokyo-Yokahama"	"Japan"	"29971"	"29971"	23356

Name: NBS8-W-8

Purpose: Replace "previous" FADU.

Procedure: Replace the "current" FADU in file **R8W8** with a new FADU identified as node "Rangoon" and whose data elements contain: "Burma", "2558", "3332", and 54426. This action should fail.

Pass Condition: The responder verifies that file **R8W8** contains:

Node Name	Contents			
"Los Angeles"	"U.S."	"9638"	"10714"	8682
"Manchester"	"U.K."	"4151"	"3827"	11627
"Mexico City"	"Mexico"	"16900"	"27872"	32377
"Singapore"	"Singapore"	"2556"	"2913"	32769
"Sydney"	"Australia"	"3396"	"3708"	10047
"Tokyo-Yokahama"	"Japan"	"29971"	"29971"	23356

Name: NBS8-W-9

Purpose: Replace FADU whose node sequence matches an existing node.

Procedure: Replace an FADU in file **R8W9** identified as node "Singapore" and whose data elements contain: "Sir Thomas", "Stamford", "Raffles", and 1819.

Pass Condition: The responder verifies that file **R8W9** contains:

Node Name	Contents			
"Los Angeles"	"U.S."	"9638"	"10714"	8682
"Manchester"	"U.K."	"4151"	"3827"	11627
"Mexico City"	"Mexico"	"16900"	"27872"	32377
"Singapore"	"Sir Thomas"	"Stamford"	"Raffles"	1819
"Sydney"	"Australia"	"3396"	"3708"	10047
"Tokyo-Yokahama"	"Japan"	"29971"	"29971"	23356

Name: NBS8-W-10

Purpose: Replace FADU whose node sequence does not match an existing node.

Procedure: Replace an FADU in file **R8W10** identified as node "Seoul" and whose data elements contain: "South Korea", "13665", "21976", and 39956.

Pass Condition: The responder verifies that file **R8W10** contains:

Node Name	Contents			
"Los Angeles"	"U.S."	"9638"	"10714"	8682
"Manchester"	"U.K."	"4151"	"3827"	11627
"Mexico City"	"Mexico"	"16900"	"27872"	32377
"Singapore"	"Singapore"	"2556"	"2913"	32769
"Seoul"	"South Korea"	"13665"	"21976"	39956
"Sydney"	"Australia"	"3396"	"3708"	10047
"Tokyo-Yokahama"	"Japan"	"29971"	"29971"	23356

Name: NBS8-W-11

Purpose: Insert a new FADU.

Procedure: Insert a new FADU in file **R8W11** identified as node "Adoration of the Magi" and whose data elements contain: "Mantegna", 10.4, FALSE, "8512121530-0600", null, 'A'H, and '1010'B.

Pass Condition: The responder verifies that the above data unit was inserted at the beginning of file **R8W11**.

Access Service Class Tests

Name: NBS8-RR-1

Purpose: Perform multiple reads of an NBS-8 file.

Procedure: The following actions operate on file **R8R**. The received contents of these operations are stored into file **A8RR1**. The actions are:

1. Locate FADU whose node name is "Manchester".
2. Read "current" FADU.
3. Locate "next" FADU.
4. Read "next" FADU.
5. Locate "end" FADU.
6. Read "previous" FADU.
7. Locate "previous" FADU.
8. Read "current" FADU.

Pass Condition: The initiator verifies that the contents of file **A8RR1** contains:

Node Name	Contents			
"Manchester"	"U.K."	"4151"	"3827"	11627
"Singapore"	"Singapore"	"2556"	"2913"	32769
"Tokyo-Yokahama"	"Japan"	"29971"	"29971"	23356
"Sydney"	"Australia"	"3396"	"3708"	10047

Name: NBS8-RR-2

Purpose: Perform multiple reads of an NBS-8 file.

Procedure: The following actions operate on file **R8R**. The received contents of these operations are stored into file **A8RR2**. The actions are:

1. Locate FADU whose node name is "Los Angeles".
2. Read "current" FADU.
3. Locate FADU whose node name is "Sydney".
4. Read "current" FADU.
5. Locate "previous" FADU.
6. Read "previous" FADU.
7. Locate "next" FADU.
8. Read "current" FADU.
9. Locate FADU whose node name is "Tokyo-Yokahama".
10. Read "current" FADU.
11. Locate "current" FADU.

Pass Condition: The initiator verifies that the contents of file **A8RR2** contains:

Node Name	Contents			
"Los Angeles"	"U.S."	"9638"	"10714"	8682
"Sydney"	"Australia"	"3396"	"3708"	10047
"Mexico City"	"Mexico"	"16900"	"27872"	32377
"Singapore"	"Singapore"	"2556"	"2913"	32769
"Tokyo-Yokahama"	"Japan"	"29971"	"29971"	23356

Name: NBS8-WW-1

Purpose: Perform multiple writes to an NBS-8 file.

Procedure: The following actions operate on file **R8WW1**. The actions are:

1. Insert an FADU whose FADU identifier is "Paris" and whose data elements contain: "France", "8633", "8800", 19983.
2. Locate FADU whose node name is Sydney.
3. Replace FADU "Sydney" with a new FADU whose identifier is "Sydney" and whose data elements contain: "Is A" "Lively", "City", 12345.
4. Replace the "next" FADU's data elements contents with "Is An", "Exciting", "City", 500.
5. Locate "begin" FADU.
6. Insert an FADU whose FADU identifier is "Kiev" and whose data elements contain: "USSR", "2489", "3237", 40145.
8. Locate FADU whose node name is "Tokyo-Yokahama".
9. Replace the "previous" FADU's data unit contents with: "Is Still A", "Lively" "City", 100.

Pass Condition: The responder verifies that the contents of file **R8WW1** contains:

Node Name	Contents			
"Kiev"	"USSR"	"2489"	"3237"	40145
"Los Angeles"	"U.S."	"9638"	"10714"	8682
"Manchester"	"U.K."	"4151"	"3827"	11627
"Mexico City"	"Mexico"	"16900"	"27872"	32377
"Paris"	"France"	"8633"	"8800"	19983
"Singapore"	"Singapore"	"2556"	"2913"	32769
"Sydney"	"Is Still A"	"Lively"	"City"	100
"Tokyo-Yokahama"	"Is An"	"Exciting"	"City"	500

Name: NBS8-WW-2

Purpose: Erase an NBS-8 file.

Procedure: Locate "begin" of file **R8WW2**. Erase file **R8WW2**.

Pass Condition: The responder verifies that file **R8WW2** only contains a root node with an empty data unit.

Name: NBS8-RW-1

Purpose: Perform multiple reads and writes to an NBS-8 file.

Procedure: The following actions operate on file **R8RW1**. The received contents of these operations are stored into file **A8RW1**. The actions are:

1. Locate FADU whose node name is "Mexico City".
2. Read "current" FADU.
3. Locate "begin" FADU.
4. Insert a new FADU whose node name is "Rome" and whose data elements contain: "Italy", "2944", "3129", and 42667.
5. Insert a new FADU whose node name is "Chicago" and whose data elements contain: "U.S.", "6500", "6568", and 8544.
6. Insert a new FADU whose node name is "Washington D.C." and whose data elements contain: "U.S.", "2456", "2700", and 6880.
7. Locate FADU whose node name is "Washington D.C.".
8. Read "current" FADU.
9. Locate "previous" FADU.
10. Read "current" FADU.
11. Locate "begin" FADU.
12. Read "current" FADU.

Pass Condition: The initiator verifies that the contents of file **A8RW1** contains:

Node Name	Contents			
"Mexico City"	"Mexico"	"16900"	"27872"	32377
"Washington D.C."	"U.S."	"2456"	"2700"	6880
"Tokyo-Yokahama"	"Japan"	"29971"	"29971"	23356
"Chicago"	"U.S."	"6500"	"6568"	8544
"Los Angeles"	"U.S."	"9638"	"10714"	8682
"Manchester"	"U.K."	"4151"	"3827"	11627
"Mexico City"	"Mexico"	"16900"	"27872"	32377
"Rome"	"Italy"	"2944"	"3129"	42667
"Singapore"	"Singapore"	"2556"	"2913"	32769
"Sydney"	"Australia"	"3396"	"3708"	10047
"Tokyo-Yokahama"	"Japan"	"29971"	"29971"	23356
"Washington D.C."	"U.S."	"2456"	"2700"	6880

Name: NBS8-RW-2

Purpose: Erase FADUs in an NBS-8 file.

Procedure: The following actions operate on file **R8RW2**. The received contents of these operations are stored into file **A8RW2**. The actions are:

1. Locate FADU whose node name is "Singapore".
2. Erase "previous" FADU.
3. Erase "current" FADU.
4. Erase "next" FADU.
5. Erase FADU whose node name is "Manchester".
6. Erase FADU whose node name is "Los Angeles".
7. Read "begin" FADU.

Pass Condition: The initiator verifies that the contents of file **A8RW2** contains:

Node Name	Contents			
"Sydney"	"Australia"	"3396"	"3708"	10047

Name: NBS8-RW-3

Purpose: Read first FADU, then erase the "previous" FADU in an NBS-8 file.

Procedure: The following actions operate on file **R8RW3**. The actions are:

1. Read FADU whose node name is "Los Angeles".
2. Erase "previous" FADU. This action should fail.

Pass Condition: The responder verifies that file **R8RW3** contains:

Node Name	Contents			
"Los Angeles"	"U.S."	"9638"	"10714"	8682
"Manchester"	"U.K."	"4151"	"3827"	11627
"Mexico City"	"Mexico"	"16900"	"27872"	32377
"Singapore"	"Singapore"	"2556"	"2913"	32769
"Sydney"	"Australia"	"3396"	"3708"	10047
"Tokyo-Yokahama"	"Japan"	"29971"	"29971"	23356

3.7 NBS-9 Test Cases

Name:	NBS9-R-1
Purpose:	Read the virtual NBS-9 file.
Procedure:	Read the virtual NBS-9 file called **DIRLIS**.
Pass Condition:	The initiator verifies with the responder that the file directory entries returned by reading the virtual file **DIRLIS** contains the current file directory entries found on the responder's filestore. **DIRLIS** does not need to be a physical file stored on the responder's filestore.

3.8 Limited File Management Test Cases

Name:	LFM-1
Purpose:	Create a file when the file does not exist.
Procedure:	Create file **RLFM1** with contents type FTAM-3 and permitted actions of read, read attributes, extend, delete file, and replace. The override parameter has no effect since the file will not exist on the responder's filestore. A new file is created with the new, specified attributes.
Pass Condition:	The responder verifies the creation of file **RLFM1** with the above attributes.

Name:	LFM-2
Purpose:	Create a file with override set to "delete and create with new attributes".
Procedure:	Create file **RLFM2** with contents type FTAM-3 and permitted actions of read, read attributes, and delete file. The override parameter is set such that the file is deleted if it already exists and a new file is created with the new, specified attributes.
Pass Condition:	The responder verifies the creation of file **RLFM2** with the above attributes, and the deletion of the previous file by the same name.

Name:	LFM-3
Purpose:	Create a file with override set to "delete and create with old attributes".
Procedure:	Create file **RLFM3** with contents type FTAM-3 and permitted actions of read, read attributes, and delete file. The override parameter is set such that the file is deleted if it already exists and a new file is created with the old, previous attributes.
Pass Condition:	The responder verifies that file **RLFM3** was deleted and recreated with the file's previously existing attributes.

Name:	LFM-4
Purpose:	Create a file with override set to "select old file".
Procedure:	Create file **RLFM4** on the responder's virtual filestore. The override parameter is set such that the file is selected if it already exists. The file **RLFM4** exists in the responder's filestore and therefore will be selected. The initiator replaces the contents of file **RLFM4** with the contents of file **ILFM4**.
Pass Condition:	The responder verifies that the contents of file **RLFM4** is identical to the contents defined for file **ILFM4**. The supported file attributes should be checked.

Name:	LFM-5
Purpose:	Create a file with override set to "create failure".
Procedure	Create file **RLFM5** on the responder's virtual filestore. The override parameter is set such that the create fails if the file already exists.
Pass Condition:	The initiator verifies that the responder rejected the creation request with a suitable diagnostic. The responder verifies that file **RLFM5** has not changed.

Name:	LFM-6
Purpose:	Read all supported file attributes.
Procedure:	Read all file attributes supported for file **RLFM6**.
Pass Condition:	The initiator verifies that at a minimum the following attributes are returned: 1. Filename 2. Permitted Actions 3. Contents Type The initiator verifies with the responder that all values returned match the actual values defined for file **RLFM6**.

Name:	LFM-7
Purpose:	Permitted actions prevent reading file attributes.
Procedure:	Read all file attributes for file **RLFM7**.
Pass Condition:	The initiator verifies that an error indicating that the file attributes can not be read is returned.

Name:	LFM-8
Purpose:	Delete a file.
Procedure:	Delete file **RLFM8**.
Pass Condition:	The responder verifies that file **RLFM8** was deleted.

Name:	LFM-9
Purpose:	Delete a non-existent file.
Procedure:	Delete file **RXYZ**. File **RXYZ** must not exist on the responder's filestore.
Pass Condition:	The initiator verifies that an error indicating that the file does not exist is returned.

Name:	LFM-10
Purpose:	Permitted actions prevent deleting a file.
Procedure:	Delete file **RLFM10**.
Pass Condition:	The initiator verifies that an error indicating that the file can not be deleted is returned.

Name:	LFM-11
Purpose:	Read the storage group file attributes.
Procedure:	Read at least the storage group attributes of file **RLFM11**.
Pass Condition:	The initiator verifies that at a minimum the following attributes in the storage group are returned:

1. File Availability
2. Filesize

The initiator verifies that all values returned match the actual values defined for file **RLFM11**. The value for the filesize attribute is based on the real filestore's representation and will not necessarily be 2048 octets.

Name: LFM-12

Purpose: Check storage file attribute values after reading a file.

Procedure: Read at least the following file attributes for file **RLFM12**:

1. Date and time of last modification
2. Date and time of last read access
3. Date and time of last attribute modification
4. Identity of creator
5. Identity of last modifier
6. Identity of last reader
7. Identity of last attribute modifier

Read file **RLMF12**. Read file attributes listed above again.

Pass Condition: The initiator verifies that the following file attributes have been updated:

1. Date and time of last read access
2. Identity of last reader

The initiator also verifies that the following file attributes did not change:

1. Date and time of last modification
2. Date and time of last attribute modification
3. Identity of creator
4. Identity of last modifier
5. Identity of last attribute modifier

Name: LFM-13

Purpose: Check storage attribute values after extending a file.

Procedure: Read at least the following file attributes for file **RLFM13**:

1. Date and time of last modification
2. Date and time of last read access
3. Date and time of last attribute modification
4. Identity of creator
5. Identity of last modifier
6. Identity of last reader
7. Identity of last attribute modifier

Extend file **RLMF13** with the contents of file **ILFM13**. Read the file attributes listed above again.

Pass Condition: The initiator verifies that the following file attributes have been updated:

1. Date and time of last modification
2. Identity of last modifier

The initiator also verifies that the following file attributes did not change:

1. Date and time of last read access
2. Date and time of last attribute modification
3. Identity of creator
4. Identity of last reader
5. Identity of last attribute modifier

Name: LFM-14

Purpose: Read the security group file attributes.

Procedure: Read at least the security group attributes of file **RLFM14**.

Pass Condition: The initiator verifies that at a minimum the following attributes in the security group are returned:

1. Access Control

The initiator verifies that all values returned match the actual values defined for file **RLFM14**.

3.9 Enhanced File Management Test Cases

Name: EFM-1
Purpose: Change kernel group file attributes.
Procedure: The following kernel group file attributes can be changed:

1. Filename

For file **REFM1**, change its filename attribute from REFM1 to RNAMECHG.
Pass Condition: The responder verifies that the filename attribute was changed to RNAMECHG.

Name: EFM-2
Purpose: Change filename to a filename that already exists.
Procedure: For file **REFM2**, change its filename attribute from REFM2 to REFM2A.
Pass Condition: The initiator verifies that an error is returned. Possible errors are "File Already Exists" or "Bad Attribute Value".

Name: EFM-3
Purpose: Attempt to change filename attribute for a non-existent file.
Procedure: For file **RXYZ**, change its filename attribute from RXYZ to RXYZABC. File **RXYZ** must not exist on the responder's filestore.
Pass Condition: The initiator verifies that an error indicating that the file does not exist is returned.

Name: EFM-4
Purpose: Permitted actions prevent changing file attributes.
Procedure: For file **REFM4**, change its filename attribute from REFM4 to REFM4A.
Pass Condition: The initiator verifies that an error indicating that the file attributes can not be changed is returned.

Name: EFM-5

Purpose: Change storage group file attributes.

Procedure: The following storage group file attributes can be changed:

1. Storage Account
2. File Availability
3. Future Filesize

For file **REFM5**, change each of the supported storage group file attributes. At a minimum, read the file availability attribute and change its value from "immediate availability" to "deferred availability". (The file availability attribute for file **REFM5** is initially set to "immediate availability".) Read the file availability attribute again. (Note: The initiator and responder must coordinate attribute values for the storage account file attribute since its representation may be system dependent.)

Pass Condition: The initiator verifies that the changed storage group attributes for file **REFM5** match those specified by the initiator. At a minimum, the initiator verifies that the file availability attribute was initially "immediately available" and then was changed to "deferred availability". The responder also verifies that the storage group attributes for file **REFM5** were changed to those values specified by the initiator. At a minimum, the responder verifies that the file availability attribute was changed to "deferred availability".

Name: EFM-6

Purpose: Check storage file attribute values after changing filename attribute.

Procedure: Read at least the following file attributes for file **REFM6**:

1. Date and time of last modification
2. Date and time of last read access
3. Date and time of last attribute modification
4. Identity of creator
5. Identity of last modifier
6. Identity of last reader
7. Identity of last attribute modifier

Change the filename attribute for file **REMF6** from REFM6 to REFM6A. Read file attributes listed above again.

Pass Condition: The initiator verifies that the following file attributes have been updated:

1. Date and time of last attribute modification
2. Identity of last attribute modifier

The initiator also verifies that the following file attributes did not change:

1. Date and time of last modification
2. Date and time of last read access
3. Identity of creator
4. Identity of last modifier
5. Identity of last reader

Name: EFM-7

Purpose: Change security group file attributes.

Procedure: The following security group file attributes can be changed:

1. Access Control
2. Legal Qualifications

For file **REFM7**, change each of the supported security group file attributes. At a minimum, the access control attribute must be changed. (Note: The initiator and responder must coordinate attribute values for the security file attributes since their representation may be system dependent.)

Pass Condition: The initiator verifies that the changed security group attributes for file **REFM7** match those specified by the initiator. The responder also verifies that the security group attributes for file **REFM7** were changed to those values specified by the initiator.

4. Test Files And Their Contents

Section 3 defines the test files used by the FTAM Interoperability Tests. For each test file, the kernel file attributes are assumed to exist and, except where noted, the permitted actions supported for a file are:

a) Actions Supported

Read	
Insert	(for FTAM-2, NBS-6, NBS-7, and NBS-8 files)
Replace	
Extend	
Erase	(for access service class tests)
Read Attributes	(for limited file management tests)
Change Attributes	(for enhanced file management tests)
Delete File	(for limited file management tests)

b) FADU-identity Groups Supported

Traversal	(for NBS-7 files)
Reverse Traversal	(for NBS-7 files)
Random Order	(for NBS-8 files)

The contents type attribute is specified by the file's document type.

Each test file is defined by its filename, document type, and file contents. In defining the FTAM–1, FTAM-2, and FTAM-3 test files, the document type field states the document type name, followed by values for the document type parameters in the sequence described by the document type definitions. For example, the document type field for test file R1R1 is:

Document Type: FTAM-1, 22, 134, 'not significant'

This states that file R1R1 is an FTAM-1 file, and has a universal class number of 22, a maximum string length of 134 characters, and a string significance of 'not significant'. This order follows the parameter sequence for the FTAM-1 document type defined in ISO 8571-2 as:

```
PARAMETERS ::= SEQUENCE {
    universal-class-number  [0] IMPLICIT INTEGER OPTIONAL,
    maximum-string-length  [1] IMPLICIT INTEGER OPTIONAL,
    string-significance  [2] IMPLICIT INTEGER
    {variable(0), fixed(1), not-significant(2)} OPTIONAL }
```

4.1 FILES USED BY FTAM-1 TEST CASES

TEST NAME: **FTAM1-R-1**

Filename: R1R1

Document Type: FTAM-1, 22, 134, 'not-significant'

File Contents: This file contains 50 strings. Each string consists of the name "FTAM1A" followed by the 128 printable characters (in ascending order) taken from the ISO 646 G0 character set starting with position 2/0 and ending with position 7/14 (95 characters) then beginning again with 2/0 and ending with 4/0 (33 characters). This results in a file that contains 6700 characters.

Filename: A1R1

Document Type: FTAM-1, 22, 134, 'not-significant'

File Contents: Generated by running test.

TEST NAME: **FTAM1-R-2**

Filename: R1R2

Document Type: FTAM-1, 27, 134, 'not-significant'

File Contents: This file contains 50 strings. The first 25 strings consists of the name "FTAM1C" followed by the 128 printable characters (in ascending order) taken from the ISO 8859-1 G0 and G1 character sets starting with position 2/0 and ending with position 7/14 (95 characters from G0) followed by the characters 10/0 through 12/0 (33 characters from G1). The second 25 strings consists of the name "FTAM1C" followed by the 128 printable characters (in ascending order) taken from the ISO 8859-1 G0 and G1 character sets starting with position 11/15 and ending with position 15/15 (65 characters from G1) followed by the characters 2/0 through 5/14 (63 characters from G0). This results in a file that contains 6700 characters.

Filename: A1R2

Document Type: FTAM-1, 27, 134, 'not-significant'

File Contents: Generated by running test.

TEST NAME: **FTAM1-R-3**

Filename: A1R3
Document Type: FTAM-1, 27, 134, 'not-significant'
File Contents: Generated by running test.

TEST NAME: **FTAM1-W-1**

Filename: R1W1
Document Type: FTAM-1, 22, 134, 'not-significant'
File Contents: Same as file R1R1.

File Name: I1W1
Document Type: FTAM-1, 22, 134, 'not-significant'
File Contents: This file contains 50 strings. Each string consists of the name "FTAM1B" followed by the 128 printable characters (in descending order) taken from the ISO 646 G0 character set starting with position 7/14 and ending with position 2/0 (95 characters) then beginning again with 7/14 and ending with 5/14 (33 characters). This results in a file that contains 6700 characters.

TEST NAME: **FTAM1-W-2**

Filename: R1W2
Document Type: FTAM-1, 27, 134, 'not-significant'
File Contents: Same as file R1R2.

Filename: I1W2
Document Type FTAM-1, 27, 134, 'not-significant'
File Contents: This file contains 50 strings. The first 25 strings consists of the name "FTAM1D" followed by the 128 printable characters (in descending order) taken from the ISO 8859-1 G0 and G1 character sets starting with position 15/15 and ending with position 10/0 (96 characters from G1) followed by the characters 7/14 through 5/15 (32 characters from G0). The second 25 strings consists of the name "FTAM1D" followed by the 128 printable characters (in descending order) taken from the ISO 8859-1 G0 and G1 character sets starting with position 6/0 and ending with position 2/0 (65 characters from G0) followed by the characters 15/15 through 12/1 (63 characters from G1). This results in a file that contains 6700 characters.

TEST NAME: **FTAM1-W-3**

Filename: R1W3
Document Type: FTAM-1, 22, 134, 'not-significant'
File Contents: Same as file R1R1.

Filename: I1W3
Document Type: FTAM-1, 22, 134, 'not-significant'
File Contents: Same as file I1W1.

TEST NAME: **FTAM1-W-4**

Filename: R1W4
Document Type: FTAM-1, 27, 134, 'not-significant'
File Contents: Same as file R1R2.

Filename: I1W4
Document Type: FTAM-1, 27, 134, 'not-significant'
File Contents: Same as file I1W2.

TEST NAME: **FTAM1-WW-1**

Filename: R1WW1
Document Type: FTAM-1, 22, 134, 'not-significant'
File Contents: Same as file R1R1.

4.2 FILES USED BY FTAM-2 TEST CASES

TEST NAMES: **All tests prefixed by FTAM2-R.**

Filename: R2R

Document Type: FTAM-2, 25, 132, 'not-significant'

File Contents: This file contains 12 FADUs, each containing one string. The strings for the data units are:

10 'O' characters,
20 'S' characters,
30 'I' characters,
40 'F' characters,
50 'T' characters,
60 'A' characters,
70 'M' characters,
60 'T' characters,
50 'E' characters,
40 'S' characters,
30 'T' characters, and
20 'S' characters.

TEST NAME: **FTAM2-R-1**

Filename: A2R1

Document Type: FTAM-2, 25, 132, 'not-significant'

File Contents: Generated by running test.

TEST NAME: **FTAM2-W-1**

Filename: R2W1

Document Type: FTAM-2, 25, 132, 'not-significant'

File Contents: Same as file R2R.

TEST NAME: **FTAM2-RR-1**

Filename: A2RR1

Document Type: FTAM-2, 25, 132, 'not-significant'

File Contents: Generated by running test.

TEST NAME: **FTAM2-WW-1**

Filename: R2WW1
Document Type: FTAM-2, 25, 132, 'not-significant'
File Contents: Same as file R2R.

TEST NAME: **FTAM2-WW-2**

Filename: R2WW2
Document Type: FTAM-2, 25, 132, 'not-significant'
File Contents: Same as file R2R.

TEST NAME: **FTAM2-RW-1**

Filename: R2RW1
Document Type: FTAM-2, 25, 132, 'not-significant'
File Contents: Same as file R2R.

Filename: A2RW1
Document Type: FTAM-2, 25, 132, 'not-significant'
File Contents: Generated by running test.

4.3 FILES USED BY FTAM-3 TEST CASES

TEST NAME: **FTAM3-R-1**

Filename: R3R1
Document Type: FTAM-3, 512, 'not-significant'
File Contents: This file contains 4 strings. Each string consists of the name "FTAM3A" followed by the 256 8-bit combinations in descending order from 255 to 0 followed by the 250 8-bit combinations in descending order from 249 to 0. This results in a file of 2048 characters.

Filename: A3R1
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Generated by running test.

TEST NAME: **FTAM3-R-2**

Filename: R3R2
Document Type: FTAM-3, 512, 'not-significant'
File Contents: This file contains 1540 strings. Each string consists of the name "FTAM3B" followed by the 256 8-bit combinations in ascending order from 0 to 255 followed by the 250 8-bit combinations in ascending order from 0 to 249. This results in a file of 788,480 characters.

Filename: A3R2
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Generated by running test.

TEST NAME: **FTAM3-R-3**

Filename: R1234567
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

Filename: A1234567
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Generated by running test.

TEST NAME: **FTAM3-R-4**

Filename: RXYZ
Document Type: FTAM-3, 512, 'not-significant'
File Contents: File RXYZ must not exist on the responder's filestore.

Filename: AXYZ
Document Type: FTAM-3, 512, 'not-significant'
File Contents: The file AXYZ, if created, should be empty.

TEST NAME: **FTAM3-R-7**

Filename: R3R7
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1. The permitted actions for file R3R7 are replace and extend. The read permitted action is not supported for file R3R7.

Filename: A3R7
Document Type: FTAM-3, 512, 'not-significant'
File Contents: The file A3R7, if created, should be empty.

TEST NAME: **FTAM3-W-1**

Filename: R3W1
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

Filename: I3W1
Document Type: FTAM-3, 512, 'not-significant'
File Contents: This file contains 4 strings. Each string consists of the filename "FTAM3C" followed by the 256 8-bit combinations in ascending order from 0 to 255 followed by the 250 8-bit combinations in ascending order from 0 to 249. This results in a file of 2048 characters.

TEST NAME: **FTAM3-W-2**

Filename: R3W2
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

Filename: I3W2
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R2.

TEST NAME: **FTAM3-W-3**

Filename: R3W3
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R2.

Filename: I3W3
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

TEST NAME: **FTAM3-W-4**

Filename: R3W4
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

Filename: I3W4
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file I3W1.

TEST NAME: **FTAM3-W-5**

Filename: R3W5
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1. Permitted actions for file R3W5 are read and extend. The replace permitted action is not supported for file R3W5.

Filename: I3W5
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file I3W1.

TEST NAME: **FTAM3-WW-1**

Filename: R3WW1

Document Type: FTAM-3, 512, 'not-significant'

File Contents: Same as file R3R1.

4.4 FILES USED BY NBS-6 TEST CASES

TEST NAMES: **All tests prefixed by NBS6-R**

Filename: R6R

Document Type: NBS-6

File Contents: The data elements of this file are represented by:

```
City-Elevation-and-Precipitation ::= SEQUENCE {
    city                    { general, 15 },
    country                 { ia5, 15 },
    elevation               { int, 2 },
    elevation-in-binary     { bit, 2 },
    elevation-in-octets     { octet, 2 },
    annual-precipitation    { float, 16, 8 },
    unit-of-measurement     { graphic, 10 },
    precipitation-over-50   boolean
}
```

The data element values for the data units are:

"Berlin"	"Germany"	187	'10111011'B	'66'H	23.1	"inches"	FALSE
"London"	"England"	149	'10010101'B	'95'H	22.9	"inches"	FALSE
"Manila"	"Philippines"	49	'110001'B	'31'H	82.0	"inches"	TRUE
"Paris"	"France"	164	'10100100'B	'A4'H	22.3	"inches"	FALSE
"Sydney"	"Australia"	62	'111110'B	'3E'H	46.5	"inches"	FALSE
"Tokyo"	"Japan"	19	'10011'B	'13'H	61.6	"inches"	TRUE

TEST NAME: **NBS6-R-1**

Filename: A6R1

Document Type: NBS-6

File Contents: Generated by running test.

TEST NAME: **NBS6-R-2**

Filename: R6R2
Document Type: NBS-6
File Contents: The data elements of this file are represented by:

```
Dates-and-Times ::= SEQUENCE {
    date-and-time              { general, 30 },
    date-time-generalized      gen-time,
    place-holder               null,
    date-time-utc              univer-time
}
```

The data element values for the data units are:

"February 29, 1992 8:45 a.m."	"19920229084500.0"	null	"9202290845Z"
"December 25, 1980 11:30 p.m."	"19801225113000.0"	null	"8012251130Z"
"January 15, 1995 2:15 p.m."	"19950115141500.0-0600"	null	"9501151415-0600"

Filename: A6R2
Document Type: NBS-6
File Contents: Generated by running test.

TEST NAME: **NBS6-W-1**

Filename: R6W1
Document Type: NBS-6
File Contents: Same as file R6R.

TEST NAME: **NBS6-W-2**

Filename: R6W2
Document Type: NBS-6
File Contents: Same as file R6R2.

TEST NAME: **NBS6-RR-1**

Filename: A6RR1
Document Type: NBS-6
File Contents: Generated by running test.

TEST NAME: **NBS6-WW-1**

Filename: R6WW1
Document Type: NBS-6
File Contents: Same as file R6R.

TEST NAME: **NBS6-WW-2**

Filename: R6WW2
Document Type: NBS-6
File Contents: Same as file R6R.

TEST NAME: **NBS6-RW-1**

Filename: R6RW1
Document Type: NBS-6
File Contents: Same as file R6R.

Filename: A6RW1
Document Type: NBS-6
File Contents: Generated by running test.

4.5 FILES USED BY NBS-7 TEST CASES

TEST NAMES: **All tests prefixed by NBS7-R**

Filename: R7R
Document Type: NBS-7
File Contents: The data elements of this file are represented by:

```
Men-and-Woman-Running-Records ::= SEQUENCE {
    event-in-meters         { int, 2 },
    world-record-time       { ia5, 15 },
    holder                  { graphic, 25 },
    country                 { general, 15 },
    date-record-set         gen-time,
    date-utctime            univer-time
}
```

The data element values for the data units are:

100	"9.83s"	"Ben Johnson"	"Canada"	"19870830151500.0"	"8708301515Z"
200	"19.72s"	"Pietro Mennea"	"Italy"	"19790917161600.0"	"7909171616Z"
400	"47.60s"	"Marita Koch"	"East Germany"	"19851006171700.0"	"8510061717Z"
1000	"2m, 12.18s"	"Sebastian Coe"	"Great Britain"	"19810711181800.0"	"8107111818Z"
3000	"7m, 32.1s"	"Henry Rono"	"Kenya"	"19780727191900.0"	"7807271919Z"

TEST NAME: **NBS7-R-1**

Filename: A7R1
Document Type: NBS-7
File Contents: Generated by running test.

TEST NAME: **NBS7-R-4**

Filename: A7R4
Document Type: NBS-7
File Contents: Generated by running test.

TEST NAME: **NBS7-R-9**

Filename: R7R9
Document Type: NBS-7
File Contents: The data elements of this file are represented by:

```
Area-Of-The-World ::= SEQUENCE {
    continent                   { ia5, 20 },
    area-in-square-miles        { int, 1 },   -- 100,000 square miles
    area-in-binary              { bit, 1 },
    area-in-octal               { octet, 1 },
    place-holder                null,
    percent-of-earth            { float, 16, 8 },
    percent-of-population       { float, 16, 8 },
    population-over-50-percent  boolean
}
```

The data element values for the data units are:

"Europe"	38	'100110'B	'26'H	null	6.6	13.5	FALSE
"Asia"	173	'10101101'B	'AD'H	null	29.9	60.0	TRUE
"Africa"	117	'1110101'B	'75'H	null	20.2	12.2	FALSE

Filename: A7R9
Document Type: NBS-7
File Contents: Generated by running test.

TEST NAME: **NBS7-W-1**

Filename: R7W1
Document Type: NBS-7
File Contents: Same as file R7R.

TEST NAME: **NBS7-W-2**

Filename: R7W2
Document Type: NBS-7
File Contents: Same as file R7R.

TEST NAME: **NBS7-W-3**

Filename: R7W3
Document Type: NBS-7
File Contents: Same as file R7R.

TEST NAMES: **NBS7-W-4 through NBS7-W-5**

Filename: R7W5
Document Type: NBS-7
File Contents: Same as file R7R.

TEST NAME: **NBS7-W-6**

Filename: R7W6
Document Type: NBS-7
File Contents: Same as file R7R.

TEST NAMES: **NBS7-W-7 through NBS7-W-10**

Filename: R7W10
Document Type: NBS-7
File Contents: Same as file R7R.

TEST NAME: **NBS7-W-11**

Filename: R7W11
Document Type: NBS-7
File Contents: Same as file R7R.

TEST NAME: **NBS7-W-12**

Filename: R7W12
Document Type: NBS-7
File Contents: Same as file R7R9.

TEST NAME: **NBS7-RR-1**

Filename: R7RR1
Document Type: NBS-7
File Contents: The data elements of this file are represented by:

```
Languages-of-the-World ::= SEQUENCE {
    language                { ia5, 15 },
    place-holder            null,
    spoken-by-in-millions   { int, 2 }
}
```

The data element values for the data units are:

"Arabic"	null	187
"English"	null	431
"Esperanto"	null	2
"French"	null	117
"German"	null	118
"Italian"	null	63
"Japanese"	null	124
"Korean"	null	68
"Mandarin"	null	825
"Portuguese"	null	169
"Russian"	null	289
"Spanish"	null	320

Filename: A7RR1
Document Type: NBS-7
File Contents: Generated by running test.

TEST NAME: **NBS7-RR-2**

Filename: A7RR2
Document Type: NBS-7
File Contents: Generated by running test.

TEST NAME: **NBS7-WW-1**

Filename: R7WW1
Document Type: NBS-7
File Contents: Same as file R7RR1.

TEST NAME: **NBS7-WW-2**

Filename: R7WW2
Document Type: NBS-7
File Contents: Same as file R7RR1.

TEST NAME: **NBS7-WW-3**

Filename: R7WW3
Document Type: NBS-7
File Contents: Same as file R7RR1.

TEST NAME: **NBS7-RW-1**

Filename: R7RW1
Document Type: NBS-7
File Contents: Same as file R7RR1.

Filename: A7RW1
Document Type: NBS-7
File Contents: Generated by running test.

TEST NAME: **NBS7-RW-2**

Filename: R7RW2
Document Type: NBS-7
File Contents: Same as file R7RR1.

Filename: A7RW2
Document Type: NBS-7
File Contents: Generated by running test.

TEST NAME: **NBS7-RW-3**

Filename: R7RW3
Document Type: NBS-7
File Contents: Same as file R7RW2 after running test case NBS7-RW-2.

Filename: A7RW3
Document Type: NBS-7
File Contents: Generated by running test.

4.6 FILES USED BY NBS-8 TEST CASES

TEST NAMES: **All tests prefixed by NBS8-R**

Filename: R8R
Document Type: NBS-8
File Contents: The data elements of this file are represented by:

```
Population-of-Cities ::= SEQUENCE {
    country                              { ia5, 15 },
    year-1985-in-thousands               { graphic, 10 },
    year-2000-projected-in-thousands     { general, 10 },
    1985-pop-density-per-square-mile     { int, 4 }
}
Key-City-Name                            { ia5, 20 }
```

The data element values for the data units are:

Node Name	Contents			
"Los Angeles"	"U.S."	"9638"	"10714"	8682
"Manchester"	"U.K."	"4151"	"3827"	11627
"Mexico City"	"Mexico"	"16900"	"27872"	32377
"Singapore"	"Singapore"	"2556"	"2913"	32769
"Sydney"	"Australia"	"3396"	"3708"	10047
"Tokyo-Yokahama"	"Japan"	"25434"	"29971"	23356

TEST NAME: **NBS8-R-1**

Filename: A8R1
Document Type: NBS-8
File Contents: Generated by running test.

TEST NAME: **NBS8-R-8**

Filename: R8R8
Document Type: NBS-8
File Contents: The data elements of this file are represented by:

```
Top-Prices-For-World-Art ::= SEQUENCE {
    artist                  { ia5, 15 },
    price-in-millions       { float, 16, 8 },
    over-30-million         boolean,
    date-sold-utc-time      univer-time,
    place-holder            null,
    price-in-octets         { octet, 1 },     -- price rounded
    price-in-binary         { bit, 1 }
}
Key-Work-Of-Art             { general, 25 }
```

The data element values for the data units are:

Node Name	Contents						
"Irises"	"van Gogh"	53.9	TRUE	"8712101515Z"	null	'3C'H	'111100'B
"Sunflowers"	"van Gogh"	39.9	TRUE	"8702181616Z"	null	'28'H	'101000'B
"Woman Reading"	"Braque"	9.5	FALSE	"8604301717Z"	null	'A'H	'1010'B

Filename: A8R8
Document Type: NBS-8
File Contents: Generated by running test.

TEST NAME: **NBS8-W-1 through NBS8-W-4**

Filename: R8W4
Document Type: NBS-8
File Contents: Same as file R8R.

TEST NAME: **NBS8-W-5**

Filename: R8W5
Document Type: NBS-8
File Contents: Same as file R8R.

TEST NAME: **NBS8-W-6**
Filename: R8W6
Document Type: NBS-8
File Contents: Same as file R8R.

TEST NAME: **NBS8-W-7**
Filename: R8W7
Document Type: NBS-8
File Contents: Same as file R8R.

TEST NAME: **NBS8-W-8**
Filename: R8W8
Document Type: NBS-8
File Contents: Same as file R8R.

TEST NAME: **NBS8-W-9**
Filename: R8W9
Document Type: NBS-8
File Contents: Same as file R8R.

TEST NAME: **NBS8-W-10**
Filename: R8W10
Document Type: NBS-8
File Contents: Same as file R8R.

TEST NAME: **NBS8-W-11**
Filename: R8W11
Document Type: NBS-8
File Contents: Same as file R8R8.

TEST NAME: **NBS8-RR-1**
Filename: A8RR1
Document Type: NBS-8
File Contents: Same as file R8R.

TEST NAME: **NBS8-RR-2**

Filename: A8RR2
Document Type: NBS-8
File Contents: Same as file R8R.

TEST NAME: **NBS8-WW-1**

Filename: R8WW1
Document Type: NBS-8
File Contents: Same as file R8R.

TEST NAME: **NBS8-WW-2**

Filename: R8WW2
Document Type: NBS-8
File Contents: Same as file R8R.

TEST NAME: **NBS8-RW-1**

Filename: R8RW1
Document Type: NBS-8
File Contents: Same as file R8R.

Filename: A8RW1
Document Type: NBS-8
File Contents: Generated by running test.

TEST NAME: **NBS8-RW-2**

Filename: R8RW2
Document Type: NBS-8
File Contents: Same as file R8R.

Filename: A8RW2
Document Type: NBS-8
File Contents: Generated by running test.

TEST NAME: **NBS8-RW-3**

Filename: R8RW3

Document Type: NBS-8

File Contents: Same as file R8R.

4.7 FILES USED BY LIMITED FILE MANAGEMENT TEST CASES

TEST NAME: **LFM-1**

Filename: RLFM1
Document Type: FTAM-3, 512, 'not-significant'
File Contents: File RLFM1 is created.

TEST NAME: **LFM-2**

Filename: RLFM2
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

TEST NAME: **LFM-3**

Filename: RLFM3
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

TEST NAME: **LFM-4**

Filename: RLFM4
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

Filename: ILFM4
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file I3W1.

TEST NAME: **LFM-5**

Filename: RLFM5
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

TEST NAME: **LFM-6**

Filename: RLFM6
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

TEST NAME: **LFM-7**

Filename: RLFM7
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1. Permitted actions allowed are read, replace, extend, and delete file. The read attribute permitted action can not be performed on file RLFM7.

TEST NAME: **LFM-8**

Filename: RLFM8
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

TEST NAME: **LFM-10**

Filename: RLFM10
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1. Permitted actions allowed are read, replace, and extend. The delete file permitted action is not supported for file RLFM10.

TEST NAME: **LFM-11**

Filename: RLFM11
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

TEST NAME: **LFM-12**

Filename: RLFM12
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

TEST NAME: **LFM-13**

Filename: RLFM13
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

Filename: ILFM13
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file I3W1.

TEST NAME: **LFM-14**

Filename: RLFM14

Document Type: FTAM-3, 512, 'not-significant'

File Contents: Same as file R3R1.

4.8 FILES USED BY ENHANCED FILE MANAGEMENT TEST CASES

TEST NAME: **EFM-1**

Filename: REFM1
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

TEST NAME: **EFM-2**

Filename: REFM2
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

Filename: REFM2A
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file I3W1.

TEST NAME: **EFM-4**

Filename: REFM4
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1. Permitted actions allowed are read, replace, extend, read attribute and delete file. The change attribute permitted action can not be performed on file REFM4.

TEST NAME: **EFM-5**

Filename: REFM5
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1. The file availability storage attribute should be set to "immediate availability".

TEST NAME: **EFM-6**

Filename: REFM6
Document Type: FTAM-3, 512, 'not-significant'
File Contents: Same as file R3R1.

TEST NAME: **EFM-7**

Filename: REFM7

Document Type: FTAM-3, 512, 'not-significant'

File Contents: Same as file R3R1.

5. Cross Reference Table

The following cross reference table lists the document types and management attributes, and the possible actions that can be performed on them. For each action, the name of the corresponding test case defined in the *FTAM Interoperability Tests* is listed.

FTAM-1 Unstructured Constraint Set Actions (see Table 6 in ISO 8571-2)[1]

Read	FTAM1-R-1, 2
Replace	FTAM1-W-1, 2
Extend	FTAM1-W-3, 4
Erase	FTAM1-WW-1

FTAM-2 Sequential Flat Constraint Set Actions (see Table 7 and 8 in ISO 8571-2)

Locate			
Begin			FTAM2-RW-1
First			FTAM2-RR-1, -RW-1
Next			FTAM2-RR-1, -RW-1
Read			
Begin	-	whole	FTAM2-R-1, -RR-1, -RW-1
First	-	leaf	FTAM2-RR-1
Next	-	leaf	FTAM2-RR-1, -RW-1
Insert			
End	-	leaf	FTAM2-W-1, -WW-1, -RW-1
Erase			
Begin	-	whole	FTAM2-WW-2

FTAM-3 Unstructured Constraint Set Actions (see Table 6 in ISO 8571-2)

Read	FTAM3-R-1, 2, 3, 4, 5, 6, 7
Replace	FTAM3-W-1, 2, 3, 5
Extend	FTAM3-W-4, 5
Erase	FTAM3-WW-1

[1] Test case FTAM1-R-3 tests for structural simplification, character relaxation, and string length relaxation of an FTAM-2 file.

NBS-6 Sequential Flat Constraint Set Actions (see Table 7 and 8 in ISO 8571-2)

Locate			
Begin			NBS6-RW-1
End			NBS6-WW-1, -RW-1
First			NBS6-RR-1, -RW-1
Next			NBS6-RR-1
Read			
Begin	-	whole	NBS6-R-1, -RR-1, -RW-1
First	-	leaf	NBS6-RR-1
Next	-	leaf	NBS6-RR-1, -RW-1
Insert			
End	-	leaf	NBS6-W-1, -WW-1, -RW-1
Erase			
Begin	-	whole	NBS6-WW-2

NBS-7 NBS Ordered Flat Constraint Set Actions (see Table 6.14 & 6.15 in NIST Stable Agreements Document)

Locate			
Begin			NBS7-WW-3, -RW-1, 2, 3
End			NBS7-RR-1, -WW-1, -RW-1
First			NBS7-RR-1, -WW-1, -RW-1
Last			NBS7-RR-1, -WW-1
Current			NBS7-RR-2
Next			NBS7-RR-1, 2, -WW-1
Previous			NBS7-RR-1, 2, -WW-1, -RW-1
Traversal			NBS7-RR-2, -RW-1, -RW-2, 3
Read			
Begin	-	whole	NBS7-R-1
First	-	leaf	NBS7-R-2
Last	-	leaf	NBS7-R-3
Current	-	leaf	NBS7-R-4, -RR-1, 2, -RW-1, 2, 3
Next	-	leaf	NBS7-R-5
Previous	-	leaf	NBS7-R-6, -RR-1, -RW-1
Traversal	-	leaf	NBS7-R-7, 8
Insert			
End	-	leaf	NBS7-W-1, -WW-1, -RW-1
Traversal	-	leaf	NBS7-W-2
Erase			
Begin	-	whole	NBS7-WW-2
Replace			
Begin	-	whole	NBS7-W-11, -WW-3
First	-	leaf	NBS7-W-3
Last	-	leaf	NBS7-W-5
Current	-	leaf	NBS7-W-8, -WW-1, -RW-3
Next	-	leaf	NBS7-W-4, -RW-3
Previous	-	leaf	NBS7-W-6
Traversal	-	leaf	NBS7-W-7, 9, 10, -RW-2

NBS-8 NBS Ordered Flat Constraint Set Actions[2] (see Table 9 & 10 in ISO 8571-2)

Locate			
Begin			NBS8-WW-1, -RW-1
End			NBS8-RR-1
Current			NBS8-RR-2
Next			NBS8-RR-1, 2
Previous			NBS8-RR-1, 2, -RW-1
NodeSeq			NBS8-RR-1, 2, -WW-1, -RW-1, 2
Read			
Begin	-	whole	NBS8-R-1, -RW-2
Current	-	leaf	NBS8-R-2, -RR-1, 2, -RW-1
Next	-	leaf	NBS8-R-3, -RR-1
Previous	-	leaf	NBS8-R-4, -RR-1, 2
NodeSeq	-	leaf	NBS8-R-5, 6, 7, -RW-3
Insert			
Begin	-	leaf	NBS8-W-1, 2, 3, 4, -WW-1, -RW-1
Replace			
Begin	-	whole	NBS8-W-5
Current	-	leaf	NBS8-W-6
Next	-	leaf	NBS8-W-7, -WW-1
Previous	-	leaf	NBS8-W-8, -WW-1
NodeSeq	-	leaf	NBS8-W-9, 10, -WW-1
Erase			
Begin	-	whole	NBS8-WW-
Current	-	leaf	NBS8-RW-2
Next	-	leaf	NBS8-RW-2
Previous	-	leaf	NBS8-RW-2, -RW-3
NodeSeq	-	leaf	NBS8-RW-2

NBS-9 Unstructured Constraint Set Actions

Read	NBS9-R-1

Limited File Management

Create File	LFM-1, 2, 3, 4, 5
Read Attributes	LFM-6, 7, 11, 12, 13, 14
Delete File	LFM-8, 9, 10

Enhanced File Management

Change Attributes	EFM-1, 2, 3, 4, 5, 6, 7

[2] The FADU identities "first", "last", and "traversal" are not required for conformant implementations.

6. Programs Substantiating The FTAM-1 and FTAM-3 Test Files

The following short programs substantiate the FTAM-1 test files (R1R1, R1R2, R1W1 and R1W2) and the FTAM-3 test files (R3R1, R3R2 and I3W1). The programs are written in the 'C' Programming Language.

```
/*
Filename:          R1R1
Document Type:     FTAM-1, 22, 134, 'not-significant'
File Contents:     This file contains 50 strings. Each string consists of the "FTAM1A"
                   followed by the 128 printable characters (in ascending order) taken from the
                   ISO 646 G0 character set starting with position 2/0 and ending with position
                   7/14 (95 characters) then beginning again with 2/0 and ending with 4/0 (33
                   characters). This results in a file that contains 6700 characters.
*/

#include <stdio.h>

#define PERM   0644

main()
{
     char   i;
     char   j;
     int    test_file;

/* Create test file R1R1. */

     if (( test_file = creat("R1R1", PERM)) == -1 ){
       printf("Can not create file R1R1.\n");
       exit(1);
      }

/* Begin writing file R1R1 out. */

     for (i=1; i<=50; i++) {

       /* Write out "FTAM1A". */

       if (write(test_file, "FTAM1A", 6) < 6){
```

```
        printf("Can not write out FTAM1A.\n");
        exit(1);
      }

      /* Write out G0 2/0 to 7/14 (95 characters). */

      for (j=32; j<=126; j++)
        if (write( test_file, &j, 1 ) < 1){
          printf("Can not write G0 2/0 to 7/14 (95 characters).\n");
          exit(1);
        }

      /* Write out G0 2/0 to 4/0 (33 characters). */

      for (j=32; j<=64; j++)
       if (write( test_file, &j, 1 ) < 1){
          printf("Can not write G0 2/0 to 4/0 (33 characters).\n");
          exit(1);
        }

    }

    close( test_file );
}
```

```
/*
Filename:         R1R2
Document Type:    FTAM-1, 27, 134, 'not-significant'
File Contents:    This file contains 50 strings. The first 25 strings consists of the name
                  "FTAM1C" followed by the 128 printable characters (in ascending order)
                  taken from the ISO 8859-1 G0 and G1 character sets starting with position
                  2/0 and ending with position 7/14 (95 characters from G0) followed by the
                  characters 10/0 through 12/0 (33 characters from G1). The second 25 strings
                  consists of the name "FTAM1C" followed by the 128 printable characters (in
                  ascending order) taken from the ISO 8859-1 G0 and G1 character sets
                  starting with position 11/15 and ending with position 15/15 (65 characters
                  from G1) followed by the characters 2/0 through 5/14 (63 characters from
                  G0). This results in a file that contains 6700 characters.
*/

#include <stdio.h>

#define PERM    0644

main()
{
      unsigned char   i;
      unsigned char   j;
      int     test_file;

/* Create test file R1R2. */

      if (( test_file = creat("R1R2", PERM)) == -1 ){
        printf("Can not create file R1R2.\n");
        exit(1);
       }

/* Begin writing file R1R2 out. */

      for (i=1; i<=25; i++) {

        /* Write out "FTAM1C". */

        if (write(test_file, "FTAM1C", 6) < 6){
          printf("Can not write out FTAM1C.\n");
          exit(1);
```

```
        }

        /* Write out G0 2/0 to 7/14 (95 characters from G0). */

        for (j=32; j<=126; j++)
          if (write( test_file, &j, 1 ) < 1){
            printf("Can not write G0 2/0 to 7/14 (95 characters from G0).\n");
            exit(1);
           }

        /* Write out G1 10/0 to 12/0 (33 characters from G1). */

        for (j=160; j<=192; j++)
         if (write( test_file, &j, 1 ) < 1){
           printf("Can not write G1 10/0 to 12/0 (33 characters from G1).\n");
           exit(1);
          }

      }

/* Write out the next 25 strings. */

      for (i=1; i<=25; i++) {

        /* Write out "FTAM1C". */

        if (write(test_file, "FTAM1C", 6) < 6){
          printf("Can not write out FTAM1C.\n");
          exit(1);
        }

        /* Write out G1 11/15 to 15/15 (65 characters from G1). */

        for (j=191; j<=254; j++)
          if (write( test_file, &j, 1 ) < 1){
            printf("Can not write G1 11/15 to 15/15 (64 characters from G1).\n");
            exit(1);
          }

        j = 255;
        if (write( test_file, &j, 1 ) < 1){
```

```
            printf("Can not write G1 15/15 (1 character from G1).\n");
            exit(1);
          }

        /* Write out G0 2/0 to 5/14 (63 characters from G0). */

        for (j=32; j<=94; j++)
          if (write( test_file, &j, 1 ) < 1){
            printf("Can not write G0 2/0 to 5/14 (63 characters from G0).\n");
            exit(1);
          }

      }

    close( test_file );
}
```

```
/*
Filename:          I1W1
Document Type:     FTAM-1, 22, 134, 'not-significant'
File Contents:     This file contains 50 strings. Each string consists of the name "FTAM1B"
                   followed by the 128 printable characters (in descending order) taken from the
                   ISO 646 G0 character set starting with position 7/14 and ending with
                   position 2/0(95 characters) then beginning again with 7/14 and ending with
                   5/14 (33 characters). This results in a file that contains 6700 characters.
*/

#include <stdio.h>

#define PERM   0644

main()
{
     char   i;
     char   j;
     int    test_file;

/* Create test file I1W1. */

     if (( test_file = creat("I1W1", PERM)) == -1 ){
       printf("Can not create file I1W1.\n");
       exit(1);
      }

/* Begin writing file I1W1 out. */

     for (i=1; i<=50; i++) {

       /* Write out "FTAM1B". */

       if (write(test_file, "FTAM1B", 6) < 6){
         printf("Can not write out FTAM1B.\n");
         exit(1);
        }

       /* Write out G0 7/14 to 2/0 (95 characters). */

       for (j=126; j>=32; j--)
```

```
        if (write( test_file, &j, 1 ) < 1){
          printf("Can not write G0 2/0 to 7/14 (95 characters).\n");
          exit(1);
        }

      /* Write out G0 7/14 to 5/14 (33 characters). */

      for (j=126; j>=94; j--)
        if (write( test_file, &j, 1 ) < 1){
          printf("Can not write G0 7/14 to 5/14 (33 characters).\n");
          exit(1);
        }

    }

    close( test_file );
}
```

```
/*
Filename:          I1W2
Document Type:     FTAM-1, 27, 134, 'not-significant'
File Contents:     This file contains 50 strings. The first 25 strings consists of the name
                   "FTAM1D" followed by the 128 printable characters (in descending order)
                   taken from the ISO 8859-1 G0 and G1 character sets starting with position
                   15/15 and ending with position 10/0 (96 characters from G1) followed by the
                   characters 7/14 through 5/15 (32 characters from G0). The second 25
                   strings consists of the name "FTAM1D" followed by the 128 printable
                   characters (in descending order) taken from the ISO 8859-1 G0 and G1
                   character sets starting with position 6/0 and ending with position 2/0 (65
                   characters from G0) followed by the characters 15/15 through 2/1 (63
                   characters from G1). This results in a file that contains 6700 characters.
*/

#include <stdio.h>

#define PERM    0644

main()
{
      unsigned char   i;
      unsigned char   j;
      int     test_file;

/* Create test file I1W2. */

      if (( test_file = creat("I1W2", PERM)) == -1 ){
        printf("Can not create file I1W2.\n");
        exit(1);
       }

/* Begin writing file I1W2 out. */

      for (i=1; i<=25; i++) {

        /* Write out "FTAM1D". */

        if (write(test_file, "FTAM1D", 6) < 6){
          printf("Can not write out FTAM1D.\n");
          exit(1);
```

```
        }

        /* Write out G1 15/15 to 10/0 (96 characters from G1). */

        j = 255;
        if (write( test_file, &j, 1 ) < 1){
          printf("Can not write G1 15/15 (1 character from G1).\n");
          exit(1);
        }
        for (j=254; j>=160; j--)
          if (write( test_file, &j, 1 ) < 1){
            printf("Can not write G1 15/14 to 10/0 (95 characters from G1).\n");
            exit(1);
          }

        /* Write out G1 7/14 to 5/15 (32 characters from G0). */

        for (j=126; j>=95; j--)
          if (write( test_file, &j, 1 ) < 1){
            printf("Can not write G1 7/14 to 5/15 (32 characters from G0).\n");
            exit(1);
          }

      }

/* Write out the next 25 strings. */

      for (i=1; i<=25; i++) {

        /* Write out "FTAM1D". */

        if (write(test_file, "FTAM1D", 6) < 6){
          printf("Can not write out FTAM1D.\n");
          exit(1);
        }

        /* Write out G0 6/0 to 2/0 (65 characters from G0). */

        for (j=96; j>=32; j--)
          if (write( test_file, &j, 1 ) < 1){
            printf("Can not write G0 6/0 to 2/0 (65 characters from G0).\n");
```

```
            exit(1);
          }

        /* Write out G1 15/15 to 12/1 (63 characters from G1). */

        j = 255;
        if (write( test_file, &j, 1 ) < 1){
          printf("Can not write G1 15/15 (1 character from G1).\n");
          exit(1);
        }

        for (j=254; j>=193; j--)
          if (write( test_file, &j, 1 ) < 1){
            printf("Can not write G1 15/14 to 12/1 (62 characters from G1).\n");
            exit(1);
          }

      }

      close( test_file );
}
```

```
/*
Filename:          R3R1
Document Type:     FTAM-3, 512, 'not-significant'
File Contents:     This file contains 4 strings. Each string consists of the name "FTAM3A"
                   followed by the 256 8-bit combinations in descending order from 255 to 0
                   followed by the 250 8-bit combinations in descending order from 249 to 0.
                   This results in a file of 2048 characters.
*/

#include <stdio.h>

#define PERM   0644

main()
{
     char  ch;
     int   i;
     int   j;
     int   test_file;

/* Create test file R3R1. */

     if (( test_file = creat("R3R1", PERM)) == -1 ){
       printf("Can not create file R3R1.\n");
       exit(1);
      }

/* Begin writing file R3R1 out. */

     for (i=1; i<=4; i++) {

       /* Write out "FTAM3A". */

       if (write(test_file, "FTAM3A", 6) < 6){
         printf("Can not write out FTAM3A.\n");
         exit(1);
        }

       /* Write from position 255 to position 0. */

       for (j=255; j>=0; j--){
```

```
            ch = (char) j;
            if (write( test_file, &ch, 1 ) < 1){
              printf("Can not write from position 255 to position 0.\n");
              exit(1);
            }
          }

          /* Write from position 249 to position 0. */

          for (j=249; j>=0; j--){
            ch = (char) j;
            if (write( test_file, &ch, 1 ) < 1){
              printf("Can not write from position 249 to position 0.\n");
              exit(1);
            }
          }

        }

        close( test_file );
}
```

```
/*
Filename:          R3R3
Document Type:     FTAM-3, 512, 'not-significant'
File Contents:     This file contains 1540 strings.  Each string consists of the name "FTAM3B"
                   followed by the 256 8-bit combinations in ascending order from 0 to 255
                   followed by the 250 8-bit combinations in ascending order from 0 to 249.
                   This results in a file of 788,480 characters.
*/

#include <stdio.h>

#define PERM    0644

main()
{
      char    ch;
      register int    i;
      register int    j;
      int     test_file;

/* Create test file R3R2. */

      if (( test_file = creat("R3R2", PERM)) == -1 ){
        printf("Can not create file R3R2.\n");
        exit(1);
       }

/* Begin writing file R3R2 out. */

      for (i=1; i<=1540; i++) {

        /* Write out "FTAM3B". */

        if (write(test_file, "FTAM3B", 6) < 6){
          printf("Can not write out FTAM3B.\n");
          exit(1);
         }

        /* Write from position 0 to position 255. */

        for (j=0; j<=255; j++){
```

```
            ch = (char) j;
            if (write( test_file, &ch, 1 ) < 1){
              printf("Can not write from position 0 to position 255.\n");
              exit(1);
            }
          }

          /* Write from position 0 to position 249. */

          for (j=0; j<=249; j++){
            ch = (char) j;
            if (write( test_file, &ch, 1 ) < 1){
              printf("Can not write from position 0 to position 249.\n");
              exit(1);
            }
          }

        }

        close( test_file );
}
```

```
/*
Filename:          I3W1
Document Type:     FTAM-3, 512, 'not-significant'
File Contents:     This file contains 4 strings. Each string consists of the name "FTAM3C"
                   followed by the 256 8-bit combinations in ascending order from 0 to 255
                   followed by the 250 8-bit combinations in ascending order from 0 to 249.
                   This results in a file of 2048 characters.
*/

#include <stdio.h>

#define PERM   0644

main()
{
     char   ch;
     register int   i;
     register int   j;
     int    test_file;

/* Create test file I3W1. */

     if (( test_file = creat("I3W1", PERM)) == -1 ){
       printf("Can not create file I3W1.\n");
       exit(1);
      }

/* Begin writing file I3W1 out. */

     for (i=1; i<=4; i++) {

       /* Write out "FTAM3C". */

       if (write(test_file, "FTAM3C", 6) < 6){
         printf("Can not write out FTAM3C.\n");
         exit(1);
        }

       /* Write from position 0 to position 255. */

       for (j=0; j<=255; j++){
```

```
            ch = (char) j;
            if (write( test_file, &ch, 1 ) < 1){
              printf("Can not write from position 0 to position 255.\n");
              exit(1);
            }
          }

          /* Write from position 0 to position 249. */

          for (j=0; j<=249; j++){
            ch = (char) j;
            if (write( test_file, &ch, 1 ) < 1){
              printf("Can not write from position 0 to position 249.\n");
              exit(1);
            }
          }

        }

        close( test_file );
}
```

Appendix A. FTAM Interoperability Requirements for GOSIP

The FTAM tests are grouped according to type of system (limited-purpose (LP) or full-purpose (FP)) and implementation roles (initiator-sender, responder-receiver, initiator-receiver or responder-sender). Since there are two types of systems and initiator-senders will test with responder-receivers (Group 1) and initiator-receivers will test with responder-senders (Group 2), the tests are divided into four tables. Table 1 gives the tests for FP systems - Group 1, Table 2 gives the for LP systems - Group 2, Table 3 gives the tests for FP systems - Group 1, and Table 4 gives the tests for FP systems - Group 2. Within each table, tests are organized in terms of the implementation profiles specified in Section 4.2.7.2 of GOSIP and Sections 9.18 and 9.19 of the Workshop Agreements.

A limited-purpose system implementing all of the roles must pass all of the tests in Table 1 and 2. A limited-purpose system implementing a subset of the roles must pass all of the tests in Table (1 or 2) corresponding to the roles that were implemented. For each role in Table 1 and/or 2 implemented, the appropriate tests must be passed for that role. A full-purpose system implementing all of the roles must, in addition, pass all of the tests in Table 3 and 4 as indicated. A full-purpose system implementing a subset of the roles must, in addition, pass all of the tests in the table (3 or 4) corresponding to the roles that were implemented. For each role in Table 3 and/or 4 implemented, the appropriate tests must be passed for that role. Full-purpose systems must pass limited-purpose tests for each defined role. This is assumed by the tables.

LP SYSTEMS - GROUP 1 TESTS
TABLE 1

TEST	IMPLEMENTATION PROFILE
FTAM1-W-1	T1
FTAM1-W-2	T1
FTAM1-W-3	T1
FTAM1-W-4	T1
FTAM3-W-1	T1
FTAM3-W-2	T1
FTAM3-W-3	T1
FTAM3-W-4	T1
*LFM-1	M1
*LFM-2	M1
*LFM-3	M1
*LFM-4	M1
*LFM-5	M1
*LFM-6	M1
*LFM-8	M1
*EFM-1	M1
*EFM-2	M1

* Duplicate of test in Table 2, skip (for initiator) if this test has been previously passed (as initiator). Skip (for responder) if this test has been passed as responder.

Note: Only one of LFM-2, LFM-3, LFM-4, or LFM-5 needs to be passed.

LP SYSTEMS - GROUP 2 TESTS
TABLE 2

TEST	IMPLEMENTATION PROFILE
FTAM1-R-1	T1
FTAM1-R-2	T1
FTAM3-R-1	T1
FTAM3-R-2	T1
FTAM3-R-3	T1
FTAM3-R-4	T1
FTAM3-R-5	T1
*LFM-1	M1
*LFM-2	M1
*LFM-3	M1
*LFM-4	M1
*LFM-5	M1
*LFM-6	M1
*LFM-8	M1
*EFM-1	M1
*EFM-2	M1

* Duplicate of test in Table 1, skip (for initiator) if this test has been previously passed (as initiator). Skip (for responder) if this test has been passed as responder.

Note: Only one of LFM-2, LFM-3, LFM-4, or LFM-5 needs to be passed.

Note 2: Test case FTAM1-R-3 is recommended to demonstrate interoperability between a limited-purpose and full-purpose system. This test can only be run if the responder supports the FTAM-2 document type.

FP SYSTEMS - GROUP 1 TESTS
TABLE 3

TEST	IMPLEMENTATION PROFILE
AFTAM1-W-1	A1
AFTAM1-W-2	A1
FTAM2-W-1	T2
AFTAM2-W-1	A1
AFTAM2-RW-1	A1
AFTAM2-WW-1	A1
AFTAM3-W-1	A1
AFTAM3-W-2	A1
AFTAM3-W-3	A1
AFTAM3-W-4	A1
AFTAM3-WW-1	A1

FP SYSTEMS - GROUP 2 TESTS
TABLE 4

TEST	IMPLEMENTATION PROFILE
AFTAM1-R-1	A1
FTAM2-R-1	T2
AFTAM2-R-1	A1
AFTAM2-RR-1	A1
AFTAM2-RW-1	A1
AFTAM3-R-1	A1
AFTAM3-R-2	A1
AFTAM3-R-3	A1
AFTAM3-R-4	A1

Note: Each FP system must pass all of the 'A' tests in both tables 3 and 4 for each role implemented.

Note 2: AFTAM2-RW-1 is duplicated in tables 3 and 4 and only needs to be passed once.

Appendix B. Connection Parameters

Appendix B defines the connection parameters for each of the NIST Implementation Profiles.

NIST Implementation Profile T1

Group Summary This group tests the simple file transfer capability.

Profile Connection Parameters

Initiator The initiator establishes a connection with the responder, proposing according to its capability:

- the Transfer service class or the Transfer, Management, Transfer and Management service class;
- at least the kernel file attribute group;
- at least the kernel, read and/or write, and grouping functional units; and,
- at least the abstract syntax for FTAM-1 and FTAM-3 contents types, other contents types may be proposed.

Responder The responder accepts a connection with the initiator, negotiating down as necessary, but must support:

- the Transfer service class;
- the kernel file attribute group;
- the kernel, read and/or write, and grouping functional units; and,
- the abstract syntax for FTAM-1 and FTAM-3 contents types.

NIST Implementation Profile T2

Group Summary This group tests the positional file transfer capability.

Profile Connection Parameters

Initiator The initiator establishes a connection with the responder, proposing according to its capability:

- the Transfer service class or the Transfer, Management, Transfer and Management service class;
- at least the kernel file attribute group;
- at least the kernel, read and/or write, and grouping functional units; and,

- at least the abstract syntax for FTAM-1, FTAM-2, and FTAM-3 contents types, other contents types may be proposed.

Responder The responder accepts a connection with the initiator, negotiating down as necessary, but must support:

- the Transfer service class;
- the kernel file attribute group;
- the kernel, read and/or write, and grouping functional units; and,
- the abstract syntax for FTAM-1, FTAM-2, and FTAM-3 contents types.

The NBS-6, NBS-7 and NBS-9 contents types may be accepted if they are supported.

NIST Implementation Profile A1

Group Summary This group tests the simple file access capability.

Profile Connection Parameters

Initiator The initiator establishes a connection with the responder, proposing according to its capability:

- at least the Access service class;
- at least the kernel file attribute group;
- at least the kernel, read, write, and file access functional units; and,
- at least the abstract syntax for FTAM-1, FTAM-2, and FTAM-3 contents types, other contents types may be proposed.

Responder The responder accepts a connection with the initiator, negotiating down as necessary, but must support:

- the Access service class;
- the kernel file attribute group;
- the kernel, read, write, and file access functional units; and,
- the abstract syntax for FTAM-1, FTAM-2, and FTAM-3 contents types.

The NBS-6 and NBS-7 contents types may be accepted if they are supported.

NIST Implementation Profile M1

Group Summary This group tests the file management capability.

Profile Connection Parameters

Initiator The initiator establishes a connection with the responder, proposing according to its capability:

- at least the Management service class;
- the kernel file attribute group;
- optionally, the storage and/or security file attribute groups;
- the kernel, limited file management, enhanced file management, and grouping functional units; and,
- the abstract syntax for FTAM-3 contents type.

Note 1 In addition to the connection parameters given above, test case LFM-4 requires the use of the Transfer service class with Write functional units.

Note 2 LFM-12 test case requires the read functional unit.

Note 3 LFM-13 test case requires the write functional unit.

Responder The responder accepts a connection with the initiator, negotiating down as necessary, but must support:

- the Management service class;
- the kernel file attribute group;
- the kernel, limited file management, enhanced file management, and grouping functional units; and,
- the abstract syntax for FTAM-3 contents type.

References

[1] ISO 8571-1, *Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 1: General Introduction*, International Organization for Standardization (ISO), First Edition, 1 October 1988.

[2] ISO 8571-2, *Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 2: Virtual Filestore Definition*, International Organization for Standardization (ISO), First Edition, 1 October 1988.

[3] ISO 8571-3, *Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 3: File Service Definition*, International Organization for Standardization (ISO), First Edition, 1 October 1988.

[4] ISO 8571-4, *Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 4: File Protocol Specification*, International Organization for Standardization (ISO), First Edition, 1 October 1988.

[5] Dempsey, John P., editor, *FTAM Interoperability Requirements*, OSINET Technical Committee, Version 1, Edition 2, June 1990.

[6] Dempsey, John P., editor, *OSINET FTAM Interoperability Tests, Stable Test Cases*, OSINET Technical Committee, Version 1, Edition 2, June 1990.

[7] *Recommendation T.50 International Alphabet Number 5*, Consultive Committee International Telegraph and Telephone (CCITT), 1984.

[8] ISO 646, *7-BIT Coded Character Set for Information Processing Interchange*, International Organization for Standardization (ISO), 1983.

[9] ISO 8859-1, *8-Bit Single-byte Coded Graphic Character Sets - Part 1: Latin Alphabet Number 1*, International Organization for Standardization (ISO), 1987.

[10] ISO 2022, *7-bit and 8-bit Coded Character Sets - Code Extension Techniques*, International Organization for Standardization (ISO), 1986.

[11] ISO 8824, *Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)*, International Organization for Standardization (ISO), 19 May 1987.

NIST-114A
(REV. 3-89)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 4435

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
OCTOBER 1990

4. TITLE AND SUBTITLE

FTAM INTEROPERABILITY TESTS

5. AUTHOR(S)

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

10. SUPPLEMENTARY NOTES

[ ] DOCUMENT DESCRIBES A COMPUTER PROGRAM; SF-185, FIPS SOFTWARE SUMMARY, IS ATTACHED.

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

This is the File Transfer, Access and Management (FTAM) Interoperability Test Suite that was originally developed by the OSINET Technical Committee.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

Connections, FTAM, Interoperability, OSINET, Test Cases, Test Suites

13. AVAILABILITY

[X] UNLIMITED

[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).

[ ] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.

[X] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
13

15. PRICE
A02

ELECTRONIC FORM