INTERNATIONAL STANDARD

ISO 11783-13

Third edition
2022-05

Tractors and machinery for agriculture and forestry — Serial control and communications data network —

Part 13: File server

Tracteurs et matériels agricoles et forestiers — Réseaux de commande et de communication de données en série —

Partie 13: Serveur de fichiers

Reference number
ISO 11783-13:2022(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 23, *Tractors and machinery for agriculture and forestry*, Subcommittee SC 19, *Agricultural electronics*.

This third edition cancels and replaces the second edition (ISO 11783-13:2011), which has been technically revised.

The main changes are as follows:

— removal of support for short, 8.3 format, path and file names;

— addition of support for Unicode characters in path and file names;

— addition of clarifications to improve the implementation and testability of the file server protocol.

A list of all parts in the ISO 11783 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

ISO 11783 specifies a communications system for agricultural equipment based on the ISO 11898-2 protocol. SAE J1939 documents[1], on which parts of ISO 11783 are based, were developed jointly for use in truck and bus applications and for construction and agriculture applications. Joint documents were completed to allow electronic units that meet the truck and bus SAE J1939 specifications to be used by agricultural and forestry equipment with minimal changes. General information on ISO 11783 can be found in this document.

The purpose of ISO 11783 is to provide an open, interconnected system for on-board electronic systems. It is intended to enable electronic control units (ECUs) to communicate with each other, providing a standardized system.

1) Society of Automotive Engineers, Warrendale, PA, USA.

Tractors and machinery for agriculture and forestry — Serial control and communications data network —

Part 13: File server

1 Scope

The message set specified in this document is designed to support the needs of tractors and implements in using the services of a file server (FS).

An FS is a distinct control function (CF) on the mobile implement control system that enables all CFs to store or retrieve data from a file-based storage device.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 11783-1, *Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 1: General standard for mobile data communication*

ISO 11783-3, *Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 3: Data link layer*

ISO 11783-5, *Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 5: Network management*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 11783-1 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
client
control function (CF) on the mobile implement bus that uses the services of the *file server* (3.5)

3.2
directory
file which stores administrative information about other *files* (3.3)

3.3
file
data object that stores data on a storage device

3.4
file attribute
bit-coded information that defines the type and features of a *file* (3.3)

3.5
file server
FS
control function (CF) on the mobile implement bus that provides storage for *files* (3.3) and uses a set of commands for the handling of, and access to, these files

3.6
filename
name conforming to requirements of a character set, which identifies a file or *directory* (3.2)

Note 1 to entry: See Annex A for the character set.

3.7
Handle
data object used for accessing *files* (3.3) and *directories* (3.2)

3.8
hidden attribute
file attribute (3.4) indicating that the file should not appear in a directory listing

Note 1 to entry: A *client* (3.1) sets this attribute by using the file server (FS) commands.

3.9
path
specification of a *filename* (3.6) that may also include the directory name

3.10
read-only attribute
file attribute (3.4) used to prevent writing to, or deletion of, a file

Note 1 to entry: A *client* (3.1) sets this attribute by using the file server (FS) commands.

3.11
volume
directory (3.2) that refers to a specific logical or physical storage unit or space

Note 1 to entry: The primary volume is the volume used as current volume when the file server (FS) is started.

4 Requirements

4.1 General message format

The general message format uses the parameter group number as the label for a group of parameters in accordance with Annex C. Each of the parameters within the group can be expressed as characters, as scaled data defined by the ranges given in 4.2, or as one or more bits. Characters shall be transmitted with the left-most character first. Numerical parameters consisting of two or more data bytes shall be transmitted least significant byte (LSB) first. Individual parameters shall be in accordance with Annex B. When variable-length messages have eight or less data bytes, these messages shall be transmitted in a single controller area network (CAN) frame. When variable length messages have nine or more data bytes, the transport protocol (TP) or the extended transport protocol (ETP), in accordance with ISO 11783-3, is required. When a message has less than eight data bytes, the unused bytes shall be filled with FF_{16} values.

4.2 File data format

4.2.1 Data

Data consists of a block of bytes (unsigned eight-bit values). All values in the range of 0_{10} ... 255_{10} are allowed. There is no special handling of individual characters (control characters, end-of-line markers, end-of-file markers or similar characters).

4.2.2 Bit groups

Groups of one to eight bits are packed into one byte as bit 7 ... bit 0. Groups of nine to 16 bits are packed into two bytes in the order of LSB as bit 7 ... bit 0, followed by most significant byte (MSB) as bit 15 ... bit 8. Unused bits in a bit group default to a value of 0 (zero).

4.2.3 Integer

Integer values are formatted as follows:

Unsigned 8 bit	1 byte	$0 \ldots 2^8-1$	$0_{10} \ldots 255_{10}$
Unsigned 16 bit	2 bytes, LSB first	$0 \ldots 2^{16}-1$	$0_{10} \ldots 65535_{10}$
Unsigned 32 bit	4 bytes, LSB first	$0 \ldots 2^{32}-1$	$0_{10} \ldots 4294967295_{10}$
Signed 32 bit	4 bytes, LSB first, two's compliment	$-2^{31} \ldots 2^{31}-1$	$-2147483648_{10} \ldots +2147483647_{10}$

4.2.4 Character string

A string contains characters represented by one or more bytes (unsigned eight-bit values). The length of a string is specified by a string length data item. The characters in a string used as a filename or a path name shall be as specified in Annex A. The support for Unicode characters is added in file server version 4, see B.5 Version Number. Prior to version 4, a character is by a single byte.

4.3 Data transmission control

4.3.1 General

Each communication transaction between a client and the FS is initiated by a request from the client and terminated by a response from the FS. In order to provide fail-safe communications, it is important that the client assigns the received response to a corresponding request and repeat an erroneous request without triggering the complete execution again.

4.3.2 Strategy

The client can issue a request and receive no response because of transient communication problems. The failure can happen during the request message, i.e. the FS does not receive the request, or during the response message, i.e. the client does not receive the response. The client cannot distinguish between these two cases and shall repeat the request to obtain the requested data.

If there is no transaction strategy, the problem of the FS not receiving the request is resolved by the client sending a second request and the FS responding with the requested data. However, if the client does not receive the correct response data message and sends a second request, the FS then sends the next data from the file; this is because a data request automatically advances to the next data in the file.

A transaction strategy is therefore required to prevent such errors. Each client on the network maintains its own transaction number (TAN) counter, which should start at 0 after a power cycle.

Each client generates a TAN for each request, that it sends to the FS. Each TAN shall be different from the previous. This can be done by incrementing the last TAN used, for the next request. The client is responsible for checking that a received response contains the same TAN that was used in the request during the communication session, thus ensuring that there are no lost commands. The FS shall remember the last command processed and response message sent for each client. The FS compares each new request with the previous request from the same client. If the TAN is not the same, the request is implemented, and the response is sent. If the TAN is the same as the previously received request, the request is not implemented, and the previous response is sent. Thus, if the client sends a second request, in the case where the FS never received the first request, the FS receives the TAN for the first time, implements the request and sends the correct data response. If the FS receives a request with the same TAN that it has already received, it does not implement the request, but the previous response is retransmitted.

4.3.3 Timeout

The execution time of all FS commands (the time between request and response) is maintained within reasonable limits. The client shall monitor the time while waiting for a response.

The timeouts specified in ISO 11783-3 for the transport protocol and the extended transport protocol shall be met for the execution of commands.

If a timeout expires, the request is assumed to have failed and the client shall repeat the request using the same TAN.

If a request response takes longer than 200 ms after the completion of the request, the FS shall send the status message to indicate busy state to the client. This provides a request timeout of 600 ms if the FS status message does not show a busy status.

4.4 Date and time support

Several FS commands require a file date and time. UTC[2] is used for this time. The file server's implementation of real time support can be either by maintaining its own real time information or by requesting the time and date information using the Time/Date parameter group specified in ISO 11783-7.

If a client requests the date and time for the root of a volume, or the volume list itself, the file server response shall include the error “Access denied”.

The file server shall respond with the most recent time where the folder/file was accessed if the operating system provides this information. Otherwise, the file server shall respond with the information that is available.

Due to possible date and time changes (by the operator or other means) there is no guarantee, that the date and time associated with files and folders indicates the actual chronological order, in which the files/folders were accessed.

4.5 Multi-client support

The file server shall support multiple clients. If more than one client has a connection simultaneously, the file server shall function with each client as if it is the only one on the network. There shall be no interference between the commands processed for different clients.

The file server shall accept connections from all clients on the network. If the file server has limited resources, it may limit the number of open file handles. Prior to file server version 4 the requirement to support all connections was not clearly defined.

Upon connection of a client, the file server initiates the current directory for that client as the root directory of the primary volume of the file server file system. If there are no volumes, then the current

2) Coordinated universal time, or universal time, formerly known as Greenwich mean time (GMT).

directory is set to the list of volumes “\\”. The client is required to use the appropriate Change Current Directory or Open File commands to access files that need to be unique for that client. In the case where multiple clients require access to common files, these clients are responsible for synchronizing their directory and file naming conventions to enable access to these common files. To prevent unintentional access to manufacturer proprietary files, a reserved directory name is specified. The naming convention of the manufacturer-specific directory is:

MCMC0000

where 0000 contains the four-digit manufacturer code (defined in accordance with ISO 11783-5 and listed in ISO 11783-1) in decimal representation, formatted with leading zeroes. A client shall not use this manufacturer-coded directory name using a manufacturer code other than the manufacturer code in its NAME field. When the client attempts to open a file in a manufacturer-specific directory where the manufacturer code in the NAME of that client is not that of the manufacturer-specific directory name, the FS shall prevent access and return an “access denied” error code.

The complete value range from 0000 to 9999 shall be handled as manufacturer code, even though the manufacturer code is defined as an 11-bit value in ISO 11783-5, which limits the possible value range to 0 to 2047.

When a file server supports multiple volumes, manufacturer-specific directories can be created on each volume. Creation of a manufacturer-specific directory is the responsibility of the client. The manufacturer-specific directory shall only be placed at the root of each volume. If a folder with the same naming convention (MCMC0000) is found in a subfolder, it shall be treated like any other folder, i.e. it is not manufacturer specific.

4.6 File Handles

An FS may support multiple file Handles. Many of the commands available for the FS create and/or use file Handles. However, there are some commands that only use folder or filenames. Internally, if the FS creates a file Handle to process these commands, the number of open files shall be incremented to reflect the internal status of the FS.

4.7 Volumes

Different types of media (Flash memory, removable media, ruggedized disk drives, etc.) can be assigned to different volumes.

An FS can support multiple volumes. It is also possible for an FS to list no volumes — for example, with uninitialized media or no device found.

The root of a removable media is the root of the volume provided by the file server to the clients. For non-removable volumes, like display memory, a folder on that volume can be the root of the volume provided to the clients, in order to protect critical areas of this memory.

The list of volumes, specified by “\\”, is the highest layer (or base) of a directory structure.

Executing a change current directory command with “..\” based on the root of a volume, would set the current directory to the volume list. Executing the command based on the volume list itself, will keep the current directory in the volume list.

If the current directory is the volume list, the client can use relative paths starting with the volume name.

EXAMPLE Relative path Vol1\ results in absolute path \\Vol1 if \\ is current directory.

It is recommended to use only media with a file system that supports long file names. The behaviour with other file systems is not defined and can result in unexpected behaviour when changing such volumes between file server systems.

The names of the volumes are determined by the FS; however, the manufacturer of a FS may allow a proprietary service tool to name volumes using the Initialize Volume Request message (see C.5.2.2).

NOTE This document does not specify how the service tool selects the media or volumes to initialize if they are not named and listed in the list of volumes "\\".

4.8 Primary volume

The primary volume shall be a removable media. If no primary volume is available when a client connects to the file server, Get Current Directory Response shall report an empty path and error code 4. Any file operations with a relative path shall return error code 4. When a removable media becomes available, and the current directory is still an empty string, the primary volume is being set to this removable media and current directory points automatically to the root of the removable media.

Servers shall use removable volumes as primary volume to be compatible with legacy clients which expect a removable media as the primary volume.

Clients should always read the volume list and select a volume suitable for their task.

See Figure 1.

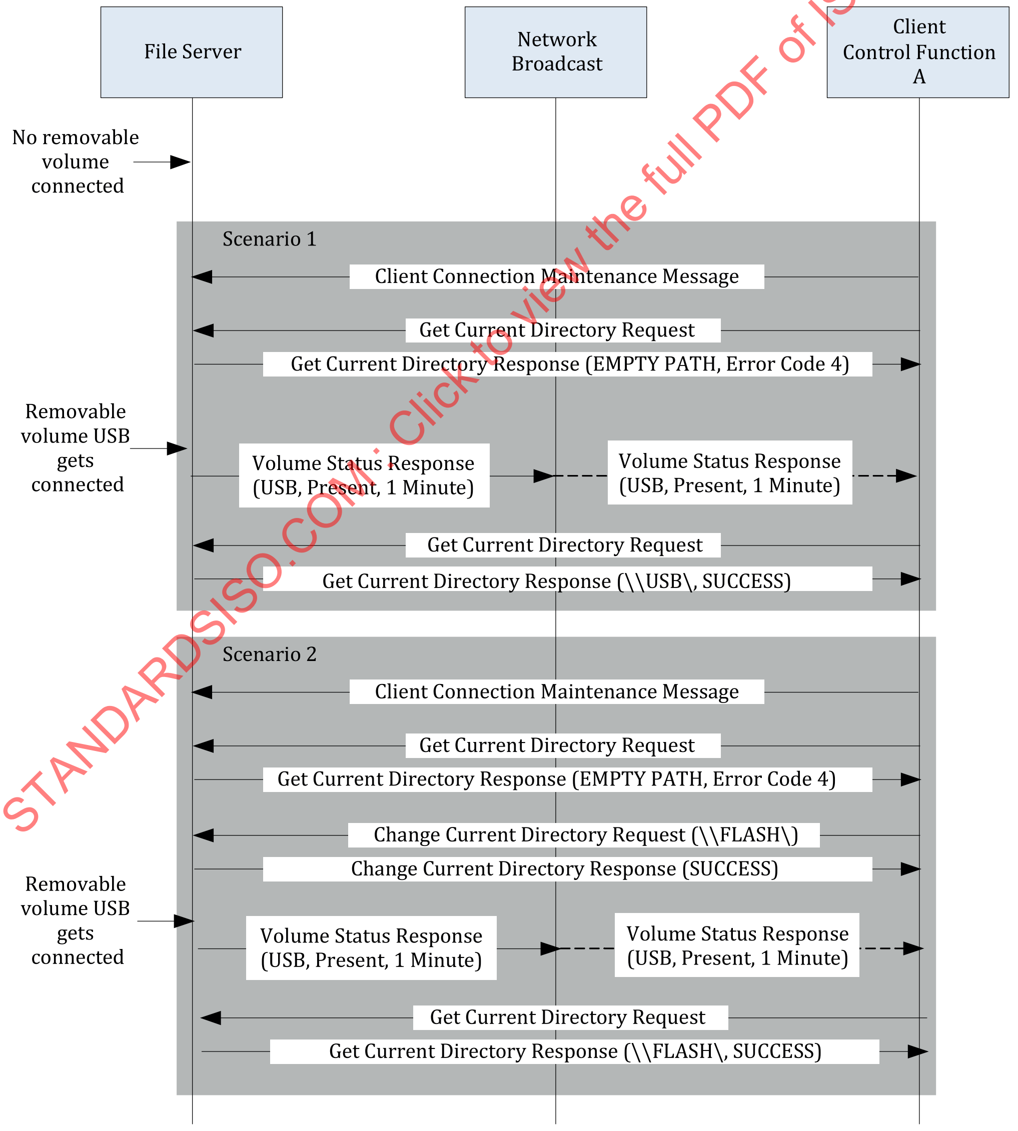
File Server
Network Broadcast
Client Control Function A
No removable volume connected
Scenario 1
Client Connection Maintenance Message
Get Current Directory Request
Get Current Directory Response (EMPTY PATH, Error Code 4)
Removable volume USB gets connected
Volume Status Response (USB, Present, 1 Minute)
Volume Status Response (USB, Present, 1 Minute)
Get Current Directory Request
Get Current Directory Response (\\USB\, SUCCESS)
Scenario 2
Client Connection Maintenance Message
Get Current Directory Request
Get Current Directory Response (EMPTY PATH, Error Code 4)
Change Current Directory Request (\\FLASH\)
Change Current Directory Response (SUCCESS)
Removable volume USB gets connected
Volume Status Response (USB, Present, 1 Minute)
Volume Status Response (USB, Present, 1 Minute)
Get Current Directory Request
Get Current Directory Response (\\FLASH\, SUCCESS)

Figure 1 — Selecting initial directory

4.9 Commands

To help identify communication issues, the FS shall respond to all destination specific commands.

The FS shall respond with a NACK (ISO 11783-3), when receiving:

— unspecified commands;

— empty request message (0 data byte).

The file server shall respond with Error Code 47 Malformed Request, if it receives a message, which is shorter than expected.

4.10 Connection/Disconnection of a client

Clients shall establish connection to the file server by sending the Client Connection Maintenance message (CCM) (C.1.3). The connection shall be established before the client sends any messages containing a TAN to the file server. The CCM shall be repeated at 2-second intervals, to maintain the connection.

NOTE For version 3 and prior, the requirement for CCM was not clearly specified, therefore those clients might not be sending CCM as specified above.

The file server shall accept a client as connected when the file server receives the CCM or any Client to Server message containing a TAN. After 6 seconds without receiving CCM or any Client to Server message containing a TAN, the file server shall consider the client as disconnected.

When a client is disconnected, the file server shall close all files, which are open by the disconnected client, and all Handles associated with that client become invalid. The client's Current Directory shall be reset to the default.

See Figure 2.

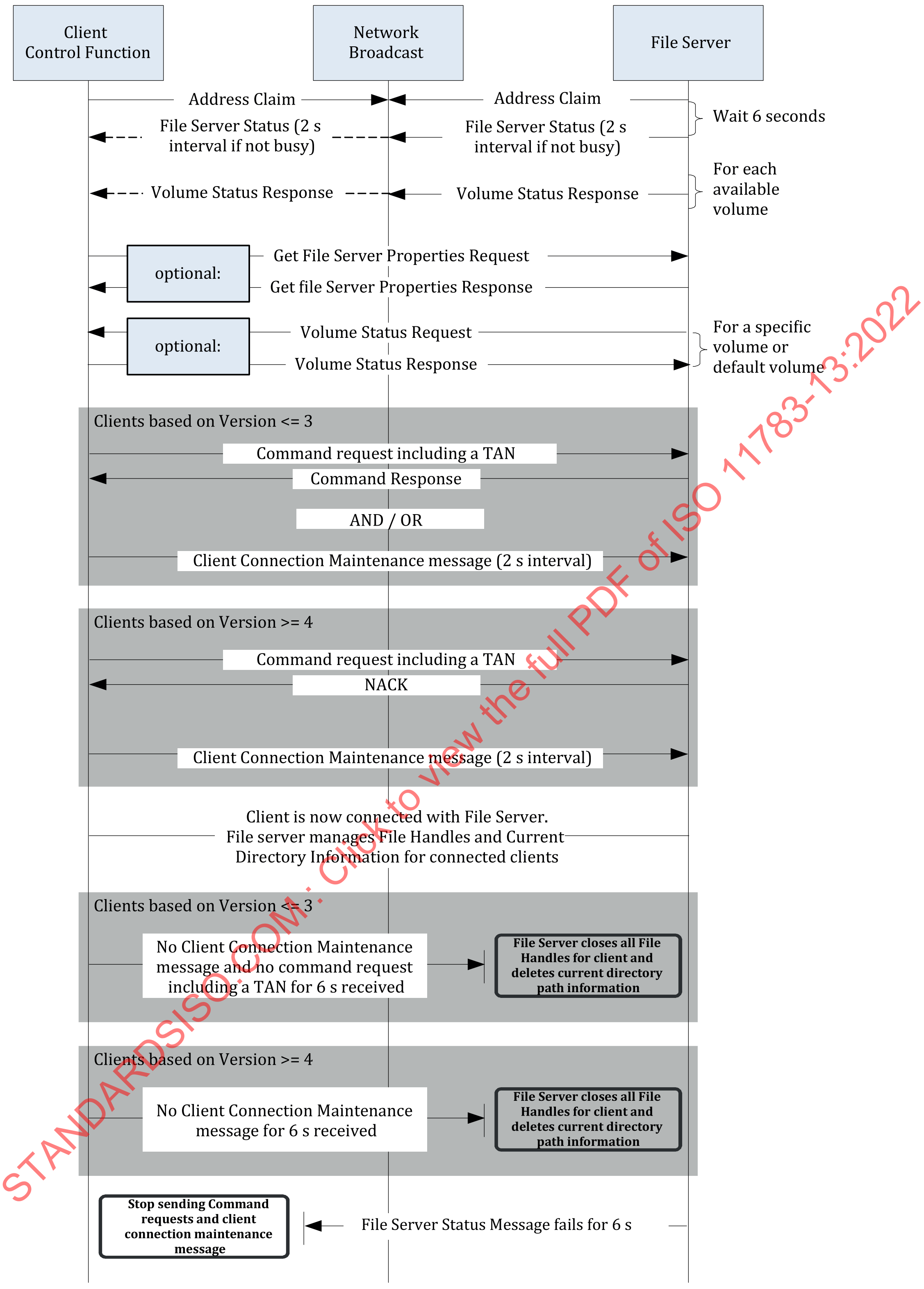
Client Control Function
Network Broadcast
File Server
Address Claim
Address Claim
Wait 6 seconds
File Server Status (2 s interval if not busy)
File Server Status (2 s interval if not busy)
Volume Status Response
Volume Status Response
For each available volume
optional:
Get File Server Properties Request
Get file Server Properties Response
optional:
Volume Status Request
Volume Status Response
For a specific volume or default volume
Clients based on Version <= 3
Command request including a TAN
Command Response
AND / OR
Client Connection Maintenance message (2 s interval)
Clients based on Version >= 4
Command request including a TAN
NACK
Client Connection Maintenance message (2 s interval)
Client is now connected with File Server. File server manages File Handles and Current Directory Information for connected clients
Clients based on Version <= 3
No Client Connection Maintenance message and no command request including a TAN for 6 s received
File Server closes all File Handles for client and deletes current directory path information
Clients based on Version >= 4
No Client Connection Maintenance message for 6 s received
File Server closes all File Handles for client and deletes current directory path information
Stop sending Command requests and client connection maintenance message
File Server Status Message fails for 6 s

Figure 2 — Client connection and disconnection

Annex A
(normative)

Character set

A.1 Valid characters

The file server uses filenames and path names. Every character used for filenames and path names is validated by the FS using the filename and path definitions given in A.2. Only printable characters are visible when presenting the filename or path name to a user.

The support for short, 8.3 format, file names, has been deprecated in file server version 4. An FS server implemented according to version 4 is required to support filename and path names per the definitions provided in A.2.

A.2 Filename and path definitions

A.2.1 Nomenclature

Definitions:

[ ]	any of the characters in the set, including none from the set (optional);
[A-B]	defines an inclusive range from the first through the last;
( )	group;
< >	character class;
\	escapes the following character, as in "\[", which indicates a single left bracket, not the containment of a set;
A | B	sequence "A" or "B";
A + B	sequence of A followed by B;
{m}	exactly m of the preceding set;
{m,n}	from m up to and including n of the preceding set;
\xXX	character code in hexadecimal notation where XX are two hexadecimal digits (\x20, for example, indicates character code 32, which is the space character).

A.2.2 Name definitions

A.2.2.1 Names

Names are from one to 255 bytes in length, using the character set given below. The commonly used file systems given in Annex D were used to determine name restrictions that allow these names to be used with minimal feature loss.

Note that the length of names is specified in bytes. The number of bytes required for a name depends on the Unicode characters and Unicode encoding method used in that name.

To avoid incompatibility between different operating systems, the client shall not create folder/files with names, which only differs in case, and names shall not end with a '.' or include ‘<’, ‘>’, ‘|’ (the latter three can cause issues on FAT32).

LongNameChar ::= any single character defined by Unicode/ISO/IEC 10646, except 0x00 to 0x1F, 0x7F to 0x9F, ‘\’, ‘*’, ‘?’, ‘/’.

WildCardChar ::= [*?] {1}

ManufacturerSpecificDirectoryChar ::= [~] {1}

PathSeparatorChar ::= [\\ ] {1}

VolumeListIndicator ::= [\\ ] {2}

ParentFolderIndicator ::= [.] {2}

CurrentFolderIndicator ::= [.] {1}

MfgSpecificFolderIndicator ::= [<ManufacturerSpecificDirectoryChar>] {1}

LongWildCardNameChar::= [<LongNameChar> | <WildCardChar>] {1}

LongName ::= [<LongNameChar>] {1,254}

LongWildCardName ::= [<LongWildCardNameChar>] {1,254}

A.2.2.2 Filenames

Filenames use the names defined in A.2.2.1.

LongFileName ::= <LongName>

EXAMPLE Test, Test.txt, Test Filename.long.name (specifically a LongName).

A.2.2.3 Volumes

Volumes use the names defined in A.2.2.1.

VolumeName ::= <LongName>

EXAMPLE VOL_B, Flash Volume

A.2.3 Path definitions

A.2.3.1 General

A path definition is similar to a filename definition but has additional prefix definitions and delimiters between path segments.

When a directory listing from path “\\” (two backslashes) is requested, the FS shall return a list of volumes. All file servers shall support “\\” so that the clients can query the volumes (including removable media), even if the FS has only one volume.

The two predefined special directory names, “.” and “..”, refer to the current (“.”) and parent (“..”) directories. These predefined directory names shall not be reported in a directory listing but may be used in a path name to specify reference to a current or parent directory.

In case they are used in a path name, the file server shall normalize the path to remove “.” and “..” before the path is checked or used by any command.

EXAMPLE 1

Assumption: Current directory is \\Vol1\SomeDirectory\

Relative path: .\File.txt → Absolute path: \\Vol1\SomeDirectory\File.txt

Relative path: ..\File.txt → Absolute path: \\Vol1\File.txt

Absolute path:\\Vol1\SomeDirectory\..\.\File.txt → Absolute path: \\Vol1\File.txt

If the path is at the list of volumes ("\\") then ".." will be ignored, also if it happens before the end of the path. Consider this absolute path:

\\Vol1\SomeDirectory\..\..\..\..\..\File.txt

After the first two ".." the path is "\\", therefore the next three ".." are ignored, and the final path becomes:

\\File.txt

If the current directory is the volume list or the volume of the current directory is no longer available, and the client changes the current directory to "\", then the current directory shall be set to the root of the primary volume.

If there are no volumes, and the client changes the current directory to "\", then the file server shall return Error Code 10 "Media not present" , because the primary volume is not present.

The "~" character (*tilde*) may be used as a placeholder for the manufacturer-specific directory of a client. This character shall only be specified at the beginning of a path or after a volume name and shall be replaced by the FS with the manufacturer-specific directory name on the current volume. If there is no current volume, then the server primary volume shall be used.

The "~" may be used anywhere in a name, and shall only be interpreted as the manufacturer-specific directory in the following two cases:

— ~\

— \\VolumeName\~\

In the following EXAMPLES the '~' shall be interpreted as name:

— .\~\ (adresses the directory with the name '~' in the current directory)

— ~name (adresses the file with the name '~name' in the current directory)

— \\VolumeName\Folder\~

(adresses the file with the name '~' in the directory 'Folder')

— \~\ (adresses the directory with the name '~' in the root directory of the current volume)

— \\~\ (adresses the Volume with name '~')

This rule enables to handle folders and files with the name '~' created outside of the file server on a removable media.

LongFolderName ::= [<LongName> | <ParentFolderIndicator> | <CurrentFolderIndicator>] {1}

LongPathName ::= [

[<VolumeListIndictor>] |

[[<VolumeListIndictor>] + <VolumeName> + <PathSeparatorChar> + [<MfgSpecificFolderIndicator> + <PathSeparatorChar>] {0,1} + [<LongFolderName> + <PathSeparatorChar>] {0,n}] |

[[<PathSeparatorChar>] {0,1} + [<LongFolderName> + <PathSeparatorChar>] {0,n}] |

[[<MfgSpecificFolderIndicator> + <PathSeparatorChar>] {0,1} + [<LongFolderName> + <PathSeparatorChar>] {0,n}]

] {1}

Notice that a path always ends with a <PathSeparatorChar>. Annex E describes the reason for this.

EXAMPLE 1 Path relative to current directory:

.\

..\path\

..\Long path name\

Path\

Level1\Level2\

Path.dir\

Long path name\

EXAMPLE 2 Path relative to root of current volume:

\Path\

\Level1\Level2\

\Path.dir\

\Long path name\ Path including volume:

\\VOL_B\path\

\\VOL_B\Level1\Level2\

\\Flash Volume\Long path name\ (specifically a LongName)

EXAMPLE 3 Path using manufacturer-specific folder indicator:

~\

~\Path\

~\Level1\Level2\

\\VOL_B\~\path\

\\VOL_B\~\Level1\Level2\

A.2.3.2 Path and filename

This path name includes as much path information as needed to produce an unambiguous specification of the path to the file:

LongPathAndFileName ::= [

[[<VolumeListIndictor>] + <VolumeName> + <PathSeparatorChar> + [<MfgSpecificFolderIndicator> + <PathSeparatorChar>] {0,1} + [<LongFolderName> + <PathSeparatorChar>] {0,n} + [<LongFileName>]] |

[[<PathSeparatorChar>] {0,1} + [<LongFolderName> + <PathSeparatorChar>] {0,n} + [<LongFileName>]] |

[[<MfgSpecificFolderIndicator> + <PathSeparatorChar>] {0,1} + [<LongFolderName> + <PathSeparatorChar>] {0,n} + [<LongFileName>]]

] {1}

EXAMPLE 1 Path relative to current directory:

Test.txt

path\Test.txt

Long path name\Test Filename.long.name

EXAMPLE 2 Path relative to root of current volume:

\path\Test

~\path\Test

~\Level1\Level2\Test

EXAMPLE 3 Path including volume:

\\VOL_B\path\Test

\\VOL_B\~\path\Test

\\VOL_B\path\Test.txt

\\VOL_B\Level1\Level2\Test

\\Flash Volume\Long path name\Test Filename.long.name

A.2.3.3 Search definitions

The wildcards "*" and "?" may be used:

— "*" is a wildcard for 0 or more characters of a filename or folder name;

— "?" is a wildcard for a single character in a filename or folder name.

Wildcards shall only be used for directory listings, and only after the last path separator.

LongPathAndWildCardName ::= [

[[<VolumeListIndictor>] + <VolumeName> + <PathSeparatorChar> + [<MfgSpecificFolderIndicator> + <PathSeparatorChar>] {0,1} + [<LongFolderName> + <PathSeparatorChar>] {0,n} + <LongWildCardName>] |

[[<PathSeparatorChar>] {0,1} + [<LongFolderName> + <PathSeparatorChar>] {0,n} + <LongWildCardName>] |

[[<MfgSpecificFolderIndicator> + <PathSeparatorChar>] {0,1} + [<LongFolderName> + <PathSeparatorChar>] {0,n} + <LongWildCardName>]

] {1}

EXAMPLE 1 Path relative to current directory:

*

?ath

?a*

~*

```
~\?ath
```

EXAMPLE 2 Path relative to root of current volume:

```
\*
\?ath
~\path\*
```

EXAMPLE 3 Path including volume:

```
\\VOL_B\*
\\VOL_B\?ath
\\VOL_B\path\Test*
\\VOL_B\path\Test*.txt
\\VOL_B\Level1\Level2\Test.*
\\VOL_B\Level1\Level2\T?st.txt
\\Flash Volume\Long path name\Test ???? Name.long.name
\\Flash Volume\Long path name\Test * Name.*.name
\\Flash Volume\Long path name\T?st Filename.long.name
```

Annex B
(normative)

Parameter definitions

B.1 Command groups

File server commands are divided into groups; four bits specify the group of the commands.

Data length: 4 bit

Value	Meaning
0000	Connection management
0001	Directory handling
0010	File access
0011	File handling
0100	Volume handling
0101..1111	Reserved

B.2 Command functions

Each FS command group has a number of functions. The lower four bits of a command byte specify the function of the command.

Data length: 4 bit

Value	Meaning
0_{16} … F_{16}	Defined in each command message

B.3 File Server Status

The current status of the FS.

Data length: 1 byte

Bit	Value	Meaning
7 … 2	000000	Reserved, send as 000000
1	1	Busy writing
0	1	Busy reading

B.4 Number of Open Files

The number of files that are currently open at the FS.

Data length:	1 byte
Resolution:	1 bit
Data Range:	0_{10} ... 255_{10} (unsigned 8 bit)

B.5 Version Number

This indicates the edition of ISO 11783-13 with which the FS server or client conforms with.

The version number parameter reported by the client shall reflect the edition of the International Standard (i.e. ISO 11783-13) to which the client is designed. It shall not change at runtime due to adaptations to different file servers. For example, a Version 3 client would still report Version 3 behaviour in this parameter, even if the client falls back to Version 2 behaviour to communicate to a Version 2 file server. The FS may choose to report this or provide it for diagnostics, but shall not reject communication or the request based on the reported Version Number.

Data length:	1 byte

Value	Meaning
0_{10}	Draft edition of the International Standard
1_{10}	Final draft edition of the International Standard
2_{10}	First published edition of the International Standard
3_{10}	Second published edition of the International Standard
4_{10}	Third published edition of the International Standard
5_{10} ... 254_{10}	Reserved
255_{10}	Compliant with Version 2 and prior (client only)

B.6 Maximum Number of Simultaneously Open Files

The maximum number of files that can be opened simultaneously at the FS.

Data length:	1 byte
Resolution:	1 bit
Data Range:	2_{10} ... 255_{10} (unsigned 8 bit)

B.7 File Server Capabilities

Data length:	1 byte

Bit	Value	Meaning
7 ... 2	000000	Reserved, send as 000000.
1	1	File server supports removable volumes.
0	1	File server supports multiple volumes.

B.8 Transaction Number

A number (TAN) assigned to a request so that the corresponding response can be identified.

Data length:	1 byte
Resolution:	1 bit
Data Range:	0 to 255

B.9 Error Code

Error codes defined here, are valid for all commands, they are referenced in.

If a response contains an error code other than 0 (Success) or 45 (File pointer at end of file), all other data beside the Command and TAN in the response shall be disregarded, unless it is otherwise specified.

Data length: 1 byte

Value	Meaning	Possible reason
0_{10}	Success.	
1_{10}	Access denied.	Client tries to access a path including a manufacturer-specific folder, which doesn't match the client's manufacturer code. File server has no access to a given path. The given handle is not assigned to the requester. In general, if the file server has no access rights to a given path.
2_{10}	Invalid access.	In all cases where the path shall point to a directory, but the existing item is a file, or vice versa.
3_{10}	Too many files open.	Maximum number of handles that can be managed by the file server is reached. No new handles can be generated. Numbers reported by status message.
4_{10}	File, path or volume not found.	The specified path is not found on the given volume (maybe folder or file was deleted before), or the specified volume is unknown to the system.
5_{10}	Invalid Handle.	The command request includes a client handle, which is not known at all to the file server.
6_{10}	Invalid given source name.	The given source folder or file name doesn't fulfil the naming requirements defined in A.2. Or is longer than what the OS can handle. Or doesn't fulfil the file system limitations.

7_{10}	Invalid given destination name.	The given destination folder or file name doesn't fulfil the naming requirements defined in A.2. Or is longer than what the OS can handle. Or doesn't fulfil the file system limitations.
8_{10}	Volume out of free space	
9_{10}	Failure during a write operation	An error in the file system occurs.
10_{10}	Media is not present.	The media referenced by the given volume name, is no longer present. For example, the removable media was removed, but the volume name is still provided within in the volume list.
11_{10}	Failure during a read operation.	An error in the file system occurs.
12_{10}	Function not supported.	Is reported by the file server for all functions, which are not provided by the file server, and which are marked optional in this document.
13_{10}	Volume is possibly not initialized.	Volume is physically available but cannot be used by the file server (e.g. no supported File System, or not mounted, or no access).
14_{10} ... 41_{10}	Reserved.	
42_{10}	Invalid request length	The file pointer hits the start of the file. On invalid space request of the volume.
43_{10}	Out of memory.	file server is out of resources and cannot complete request, e.g. no memory to store request or response data.
44_{10}	Any other error	
45_{10}	File pointer at end of file.	
46_{10}	TAN Error	Same TAN, but different request compared to the previous one (change in content or size).
47_{10}	Malformed Request.	Message is shorter than expected. If the message is too short to provide a TAN (less than 2 data bytes), the TAN shall be set to 0xFF in the response message.
48_{10} ... 255_{10}	Reserved.	

B.10 Handle

The data object assigned by the FS and used by the FS and client to reference a file or directory for requested actions.

Data length: 1 byte

Value	Meaning
0_{10} … 254_{10}	Value of the Handle assigned by the FS for further access to a file
255_{10}	Error when assigning a Handle for a file

B.11 Space

Indication of, e.g., total or free volume space.

Data length:	4 bytes
Resolution:	512 byte
Data Range:	0_{10} … 4294967295_{10} (unsigned 32 bit)

B.12 Path Name Length

Number of characters in a path name. This can be a volume, path, filename, wildcard, or combination thereof.

Data length:	2 bytes
Resolution:	1 bit
Data Range:	0_{10} … 65535_{10} (unsigned 16 bit)

B.13 Path Name

The allowed characters in a path name are specified in A.2.3.1.

B.14 Flags

The data object used by a client to specify the mode of file or directory access requested.

Data length: 11 byte

Bit	Value	Meaning
7 … 6	00	Reserved, send as 00
5		Version 4 and later:
This bit shall be ignored by the server if bit 1 and 0 are not set for ‘Open directory’		
	0	Do not report hidden files and folders in directory listing.
	1	Report hidden files and folders in directory listing.
		Version 3 and prior:
		Reserved, send as 0.
4	0	Open file for shared read access
	1	Open file with exclusive access (fails if already open)
3	0	Open file for random access (file pointer set to the start of the file)

	1	Open file for appending data to the end of the file (file pointer set to the end of the file). The file pointer can be moved by the seek commend. Reading (if enabled) will be done from the current file pointer location. Prior to any write command the file pointer is automatically moved to the end of the file. This means that new data will always be added to the end of the file, regardless of prior seek or read commands.
2	0	Open an existing file (fails if non-existent file)
	1	Create a new file and/or directories if not yet existing
1,0	00	Open file for reading only
	01	Open file for writing only
	10	Open file for reading and writing
	11	Open directory

When bits 1 and 0 are set for “Open directory”, bits 3 and 4 are ignored.

B.15 Attributes

The data object used by the FS to describe the file to the client.

Data length: 1 byte

Bit	Value	Meaning
7	0	Volume is case-insensitive
	1	Volume is case-sensitive (Version 3 and later FS support this attribute)
6	0	Volume is removable
	1	Volume is not removable
5	0	Volume does not support long filenames
	1	Volume supports long filenames
4	0	Does not specify a directory
	1	Specifies a directory
3	0	Does not specify a volume
	1	Specifies a volume
2	0	Volume does not support hidden attribute or the implementation does not support the hidden attribute for the given volume
	1	Volume supports hidden attribute and the implementation supports the hidden attribute for the given volume
1	0	“Hidden” attribute is not set

	1	"Hidden" attribute is set (not applicable unless volume supports hidden attribute)
0	0	"Read-only" attribute is not set
	1	"Read-only" attribute is set

A volume is only reported as volume and not as a directory in a Directory Entry (see B.21). In all other cases (Open File, Get File Attributes, etc.) it is reported as a directory only, because the root directory is addressed, not the volume itself.

EXAMPLES

Volume supports hidden attribute and the implementation supports the hidden attribute for the given Volume:

— USB stick with FAT32 on a Linux system.
-> hidden is most likely supported, since many Linux destributions do support hidden files on FAT32 file systems
-> the server would report "hidden attribute is supported"

— USB stick with NTFS on a Linux System
-> hidden is most likely NOT supported, since many Linux destributions do NOT support hidden files on NTFS file systems
-> the server would report "hidden attribute is NOT supported"

— USB stick with NTFS or FAT32 on a Windows based system.
-> hidden is most likely supported, since many Windows destributions do support hidden files on FAT32 and NTFS file systems
-> the server would report "hidden attribute is supported"

B.16 Set Attributes Command

The command from the client to the FS for setting or clearing file attributes.

Data length: 1 byte

Bit	Value	Meaning
7,6	11	Reserved, set to 11
5,4	11	Reserved, set to 11
3,2	00	Clear "hidden" attribute
	01	Set "hidden" attribute (not applicable unless volume supports hidden attribute)
	10	Reserved
	11	Don't care, leave "hidden" attribute in current state
1,0	00	Clear "read-only" attribute
	01	Set "read-only" attribute
	10	Reserved
	11	Don't care, leave "read-only" attribute in current state

B.17 Position Mode

The position mode specifies the location from which the offset value is used to determine the file pointer position.

Data length: 1 byte

Value	Meaning
0_{10}	From the beginning of the file
1_{10}	From the current pointer position
2_{10}	From the end of the file
3_{10} ... 255_{10}	Reserved

B.18 Offset

The offset is used with the position mode to determine the file pointer position.

Data length:	4 bytes
Resolution:	1 bit
Data Range:	-2147483648_{10} ... $+2147483647_{10}$ (signed 32 bit)

B.19 Position

The value of the file pointer position.

Data length:	4 bytes
Resolution:	1 bit
Data Range:	0_{10} ... 4294967295_{10} (unsigned 32 bit)

B.20 Count

The number of bytes of data requested to be read from, or written to, a file, or the number of directory entries read from a directory.

Data length:	2 bytes
Resolution:	1 bit
Data Range:	0_{10} ... 65535_{10} (unsigned 16 bit)

B.21 Directory Entry

Data length: Variable

Byte 1	Filename Length	see B.22
Byte 2–n	Filename	see B.23
Byte n + 1	Attributes	see B.15

Bytes $n + 2$, $n + 3$	File Date	see B.24
Bytes $n + 4$, $n + 5$	File Time	see B.25
Bytes $n + 6$ … $n + 9$	Size	see B.26

B.22 Filename Length

Number of characters in a filename.

For an FS without support for long filenames, the filename length is a maximum of 1 … 12 characters.

For an FS with support for long filenames, the filename is a maximum of 1 … 254 characters. (FS version 2 and prior specified a maximum of 31 characters.)

Data length:	1 byte
Resolution:	1 bit
Data Range:	1_{10} … 254_{10} (unsigned 8 bit)

B.23 Filename

The allowed characters in a filename are specified in A.2.2.2.

B.24 File Date

The file date is represented by a bit group with the following 16 bit encoding.

Data length:	2 bytes

Bits 15 … 9	0 … 127	Year − 1980 (difference between the year and 1980)
Bits 8 … 5	1 … 12	Month (1 = January, …, 12 = December)
Bits 4 … 0	1 … 31	Day

If a file date is not available in an implementation, all bits are set to zero ($0_{10} = 0000_{16}$) resulting in “1980-00-00”.

B.25 File Time

The file time is represented by a bit group with the following 16 bit encoding.

Data length:	2 bytes

Bits 15 … 11	0 … 23	Hours
Bits 10 … 5	0 … 59	Minutes
Bits 4 … 0	0 … 29	Seconds (in steps of 2 s)

If a file time is not available in an implementation, all bits are set to zero ($0_{10} = 0000_{16}$) resulting in “00-00-00”.

B.26 Size

The number of bytes in a file or the number of files and folders in a directory.

Data length:	4 bytes
Resolution:	1 bit
Data range:	0_{10} ... 4294967295_{10} (unsigned 32 bit)

B.27 File Handling Mode

Data length: 1 byte

Bit	Value	Meaning
7–3	00000	Reserved, send as 00000
2	0	“Recursive” mode is not set
	1	“Recursive” mode is set
1	0	“Force” mode is not set
	1	“Force” mode is set
0	0	“Copy” mode is not set
	1	“Copy” mode is set

B.28 Report Hidden Files

NOTE This applies to version 3 and prior. It is not used in version 4 and later.

Data length: 1 byte

Value	Meaning
0_{10}	Do not report hidden files in a directory listing
1_{10}	Report hidden files in a directory listing
2_{10} ... 254_{10}	Reserved
255_{10}	Parameter is not available, FS shall not report hidden files in a directory listing

B.29 Volume Flags

The data object used by the client to specify the mode of volume access requested.

Data length: 1 byte

Bit	Value	Meaning
7 ... 2	000000	Reserved, send as 000000
1	0	Create volume using all available space
	1	Create volume using space specified
0	0	Create a new volume if not yet existing (fails if existing volume)
	1	Overwrite the existing volume

B.30 Volume Mode

The data object used by a client to specify the mode of the volume access requested. A value of 00000000 requests the current status and does not request a change of the volume status. (This parameter applies to Version 3 and later file servers.)

Data length: 1 byte

Bit	Value	Meaning
7 ... 2	000000	Reserved, send as 000000
1	1	Request volume to prepare for removal
0	1	Report volume in use by client
	0	Report volume not in use by client

B.31 Volume Status

The current status of the volume. This parameter applies to Version 3 and later FS.

Data length: 1 byte

Bit	Value	Meaning
7 ... 3	00000	Reserved, send as 00000
2 ... 0	7 ... 4	Reserved
	3	Removed
	2	Preparing for removal
	1	In use
	0	Present

B.32 Maximum Time before Volume Removal

Maximum time a volume can be held off from being removed. (This parameter applies to Version 3 and later FS.)

Data length:	1 byte
Resolution:	1 min/bit, 0 offset
Data range:	0 to 250
Type:	Measured

B.33 Volume Name

The allowed characters in a volume name are specified in A.2.2.3.

B.34 Volume, Path, File and Wildcard Name

The allowed characters in a volume, path, file and wildcard name are specified in A.2.2.3, A.2.3.1, A.2.3.2 and A.2.3.3, respectively.

B.35 Volume, Path and Filename

The allowed characters in a volume, path and file name are specified in A.2.2.3, A.2.3.1 and A.2.3.2, respectively.

Annex C
(normative)

File server message definitions

C.1 File server messages and data transfer

C.1.1 Overview

Two PGN (parameter group numbers) are reserved for the FS message protocol:

a) File Server to Client

Transmission repetition rate:	As required
Data length:	Variable
Data Page field:	0
PDU Format field:	171
PDU specific field:	Destination address
Default priority:	7
Parameter group number:	43776 ($AB00_{16}$)

b) Client to File Server

Transmission repetition rate:	As required
Data length:	Variable
Data Page field:	0
PDU Format field:	170
PDU specific field:	Destination address
Default priority:	7
Parameter group number:	43520 ($AA00_{16}$)

Before a client CF begins to maintain a connection with the file server, it can obtain information about the file server's capabilities using the above-listed PGN. Those PGN are also used to transfer the data to or from the FS utilizing the transport protocol or the extended transport protocol specified in ISO 11783-3. Destination-specific messages shall be used and connection management shall be implemented.

The client CF shall wait for a response before sending another command. A fixed timeout for reception of the response cannot be given due to the fact that various commands can have very different response times. Instead, FS command processing can be monitored by the client through the File Server Status message.

The error code 12_{10}, "Function not supported", enables the FS to indicate to a client that a specific function is not supported. All file servers shall implement the complete set of functions not marked as optional; nevertheless, a file server based on an older revision of this document might support fewer functions, in the case of the addition of new functions. If a client sends a command, which is not defined within this document, the file server shall respond with a NACK (ISO 11783-3).

Even if a response message from the file server includes an error code, the file server shall fill the message with valid data.

C.1.2 File Server Status

The File Server Status message shall be sent with a Transmission repetition rate of 2 000 ms when the status is not busy, 200 ms when the status is busy reading or writing.

In addition, the message shall be sent on change of byte 2 (status), up to five messages per second, if the current value of byte 2 differs from the previous message.

The busy flag shall be set by the file server if a response cannot be sent within 200 ms of receiving the request.

A timeout of 6 s on this message indicates a possible shutdown of the file server, thus enabling the client to free allocated file server communication resources.

Transmission repetition rate:		2 000 ms when the status is not busy, 200 ms when the status is busy reading or writing and, on change of byte 2, up to five messages per second		
Data length:		8 bytes		
Priority:		5		
Parameter group number:		FS to client, destination-specific, use global address: FF_{16}		
Byte 1	FS function = 0_{10}			
	Bits 7–4 0000	Command	Connection Management	see B.1
	Bits 3–0 0000	Function	File Server Status	see B.2
Byte 2		File Server Status		see B.3
Byte 3		Number of Open Files		see B.4
Bytes 4–8		Reserved, transmit as FF_{16}		

C.1.3 Client Connection Maintenance

The Client Connection Maintenance Message is sent by a client to maintain a connection with the FS (see 4.10 Connection/Disconnection of a client).

Transmission repetition rate:		2 000 ms		
Data length:		8 bytes		
Parameter group number:		Client to FS, destination-specific		
Byte 1	FS function = 0_{10}			
	Bits 7–4 0000	Command	Connection Management	see B.1

	Bits 3–0 0000	Function	Client Connection Maintenance	see B.2
Byte 2		Version Number		see B.5
Bytes 3–8		Reserved, transmit as FF_{16}		

C.1.4 Get File Server Properties

The Get File Server Properties message is sent by the client to request the FS properties.

The client may send this message before sending a Client Connection Maintenance message. This allows the client to check the version compatibility before establishing a connection with the file server.

Transmission repetition rate:		On request		
Data length:		8 bytes		
Parameter group number:		Client to FS, destination-specific		
Byte 1	FS function = 1_{10}			
	Bits 7–4 0000	Command	Connection Management	see B.1
	Bits 3–0 0001	Function	Get File Server Properties	see B.2
Bytes 2–8		Reserved, transmit as FF_{16}		

C.1.5 Get File Server Properties Response

The Get File Server Properties Response message is sent by the FS to a client in response to the Get File Server Properties message.

Transmission repetition rate:		In response to Get File Server Properties message		
Data length:		8 bytes		
Parameter group number:		FS to client, destination-specific		
Byte 1	FS function = 1_{10}			
	Bits 7–4 0000	Command	Connection Management	see B.1
	Bits 3–0 0001	Function	Get File Server Properties	see B.2
Byte 2		Version Number		see B.5
Byte 3		Maximum Number of Simultaneously Open Files		see B.6
Byte 4		File Server Capabilities		see B.7
Bytes 5–8		Reserved, transmit as FF_{16}		

C.1.6 Volume Status Request

This message applies for Version 3 and later FS.

The Volume Status Request message is sent by the client to command the file server volume status or request the current volume status. Path Name Length of zero shall be used to request the volume of the current directory of the client.

The client may send this message before sending a Client Connection Maintenance message. This allows the client to check the availability of a volume before establishing a connection with the file server.

If there is an error, the file server shall respond with one of the valid values of volume status or 0xFF.

Transmission repetition rate:		On request		
Data length:		Variable		
Parameter group number:		Client to FS, destination-specific		
Byte 1		FS function = 2_{10}		
	Bits 7–4 0000	Command	Connection Management	see B.1
	Bits 3–0 0010	Function	Volume Status	see B.2
Byte 2		Volume Mode		see B.30
Byte 3, 4		Path Name Length		see B.12
Bytes 5–*n*		Volume Name		see B.34

C.1.7 Volume Status Response

This message applies to version 3 and later FS.

The Volume Status Response message is sent by the file server to a client in response to the Volume Status Request or on volume status change. On volume status changes, the FS sends this message to the global address (GA). If a client request causes the status to change, then the FS shall send the response to the GA, so that all clients are informed. Since the global response is on status change, this implies the error code will be Success; therefore, non-success shall only be sent to the requester. If the client "Request volume to prepare for removal" and the volume requested is not removable, or the Path Name Length of request is zero and the current directory is not set, then the FS shall return "Invalid access". The file server can use a Volume Name of "\\" to indicate a status change of all volumes on the FS.

Once the FS detects a removable volume is present, it shall report that the volume is "Present".

If a "Request volume to prepare for removal" is sent by a client, the FS shall report that the volume is "Preparing for removal" and all clients shall close their files and/or directories. In order to keep the volume at "Preparing for removal", the client needs to continually send a "Maintain" request.

The volume status transition from "Preparing for removal" to "Removed" is sent to all clients once they have all closed their files and/or directories. The FS maintains the volume for 2 s after the last client request of "Maintain" (by client reporting that the volume is in use within the Volume Status Request) for a "Maximum Time Before Volume Removal". The latter is determined by the FS and could be the time presented to the user when ejecting the media or the time allowed at power down that the FS can be held on. If all clients report the volume is no longer in use, the FS can remove the volume immediately.

If a client requests the status of the volume and there is a file or directory open on the volume or there has been a "Maintain" request on a "Preparing for removal", then the FS shall report the volume as "In use".

The maximum repetition rate for the Volume Status response message shall be 1 message per 200 ms.

If there is an error, the file server shall respond with one of the valid values of volume status or 0xFF.

Transmission repetition rate:	On request and on change of Volume Status
Data length:	Variable

Parameter group number:		FS to client, destination-specific or use global address: FF_{16}		
Byte 1		FS function = 2_{10}		
	Bits 7–4 0000	Command	Connection Management	see B.1
	Bits 3–0 0010	Function	Volume Status	see B.2
Byte 2		Volume Status		see B.31
Byte 3		Maximum Time Before Volume Removal		see B.32
Byte 4		Error Code		see B.9
		0_{10}	Success	
		1_{10}	Access denied	
		2_{10}	Invalid access	
		4_{10}	File, path or volume not found	
		6_{10}	Invalid given source name	
		43_{10}	Out of memory	
		44_{10}	Any other error	
Byte 5, 6		Path Name Length		see B.12
Byte 7–*n*		Volume Name		see B.34

EXAMPLE 1

This example shows how the clients and file server shall react, if a client requests to remove a volume, and another client still has files open.

Based on C.1.7. there is a maximum time defined for the client to finish the work, where the minimum time is 2 s (based on: file server maintains the volume for 2 seconds after the last client request of „Maintain“) and with each Maintain request of the client there is another 2 seconds, until the maximum time before volume removal is reached.

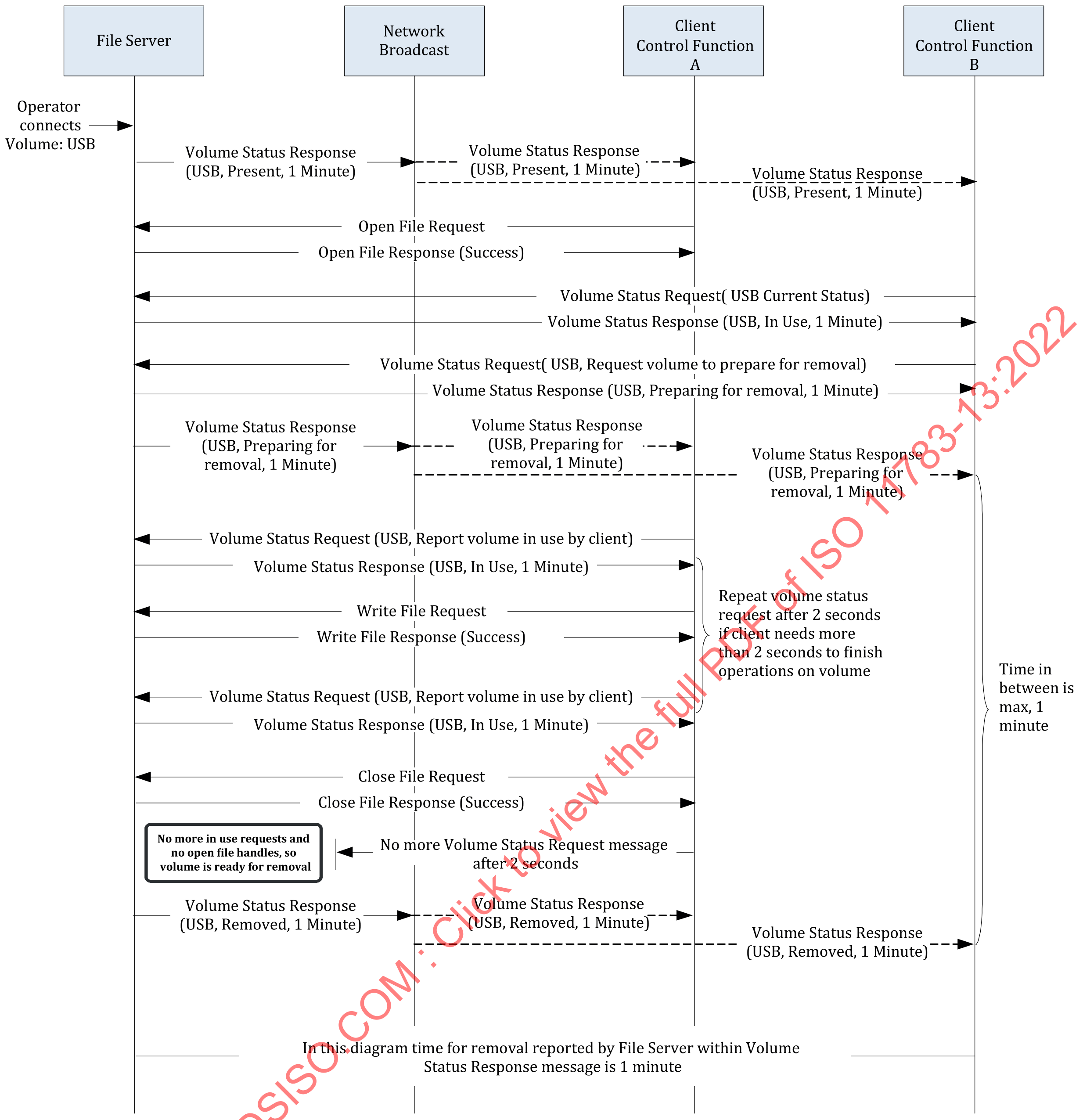
File Server
Network Broadcast
Client Control Function A
Client Control Function B
Operator connects Volume: USB
Volume Status Response (USB, Present, 1 Minute)
Volume Status Response (USB, Present, 1 Minute)
Volume Status Response (USB, Present, 1 Minute)
Open File Request
Open File Response (Success)
Volume Status Request(USB Current Status)
Volume Status Response (USB, In Use, 1 Minute)
Volume Status Request(USB, Request volume to prepare for removal)
Volume Status Response (USB, Preparing for removal, 1 Minute)
Volume Status Response (USB, Preparing for removal, 1 Minute)
Volume Status Response (USB, Preparing for removal, 1 Minute)
Volume Status Response (USB, Preparing for removal, 1 Minute)
Volume Status Request (USB, Report volume in use by client)
Volume Status Response (USB, In Use, 1 Minute)
Write File Request
Write File Response (Success)
Repeat volume status request after 2 seconds if client needs more than 2 seconds to finish operations on volume
Volume Status Request (USB, Report volume in use by client)
Volume Status Response (USB, In Use, 1 Minute)
Time in between is max, 1 minute
Close File Request
Close File Response (Success)
No more in use requests and no open file handles, so volume is ready for removal
No more Volume Status Request message after 2 seconds
Volume Status Response (USB, Removed, 1 Minute)
Volume Status Response (USB, Removed, 1 Minute)
Volume Status Response (USB, Removed, 1 Minute)
In this diagram time for removal reported by File Server within Volume Status Response message is 1 minute

Figure C.1 — Volume Status Behaviour EXAMPLE 1

EXAMPLE 2

Operator removes the USB stick from the system. There are two options:

1. Operator directly removed the USB stick

2. Operator uses UI to request that he wants to remove USB stick

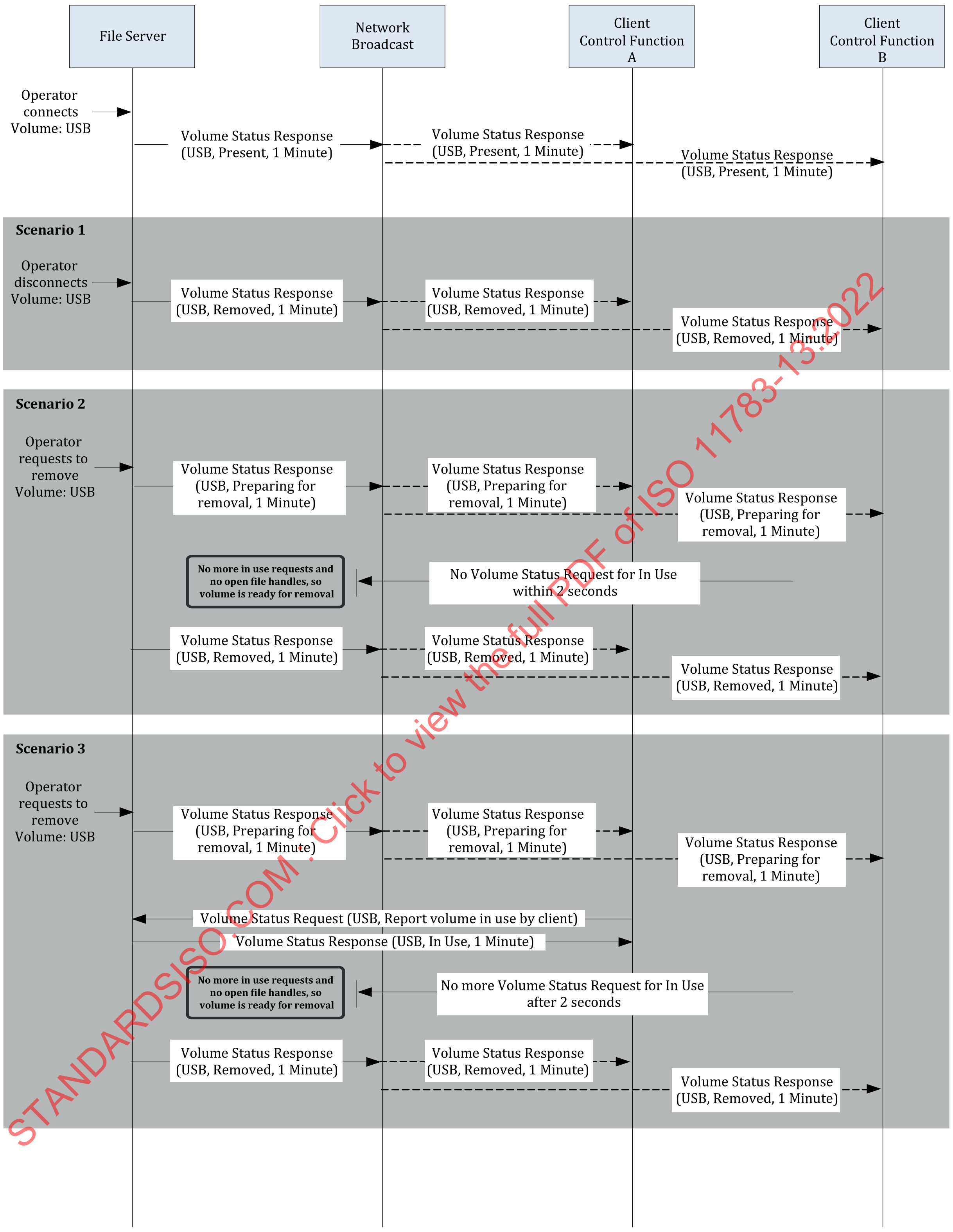
File Server
Network Broadcast
Client Control Function A
Client Control Function B
Operator connects Volume: USB
Volume Status Response (USB, Present, 1 Minute)
Volume Status Response (USB, Present, 1 Minute)
Volume Status Response (USB, Present, 1 Minute)
Scenario 1
Operator disconnects Volume: USB
Volume Status Response (USB, Removed, 1 Minute)
Volume Status Response (USB, Removed, 1 Minute)
Volume Status Response (USB, Removed, 1 Minute)
Scenario 2
Operator requests to remove Volume: USB
Volume Status Response (USB, Preparing for removal, 1 Minute)
Volume Status Response (USB, Preparing for removal, 1 Minute)
Volume Status Response (USB, Preparing for removal, 1 Minute)
No more in use requests and no open file handles, so volume is ready for removal
No Volume Status Request for In Use within 2 seconds
Volume Status Response (USB, Removed, 1 Minute)
Volume Status Response (USB, Removed, 1 Minute)
Volume Status Response (USB, Removed, 1 Minute)
Scenario 3
Operator requests to remove Volume: USB
Volume Status Response (USB, Preparing for removal, 1 Minute)
Volume Status Response (USB, Preparing for removal, 1 Minute)
Volume Status Response (USB, Preparing for removal, 1 Minute)
Volume Status Request (USB, Report volume in use by client)
Volume Status Response (USB, In Use, 1 Minute)
No more in use requests and no open file handles, so volume is ready for removal
No more Volume Status Request for In Use after 2 seconds
Volume Status Response (USB, Removed, 1 Minute)
Volume Status Response (USB, Removed, 1 Minute)
Volume Status Response (USB, Removed, 1 Minute)

Figure C.2 — Volume Status Behaviour EXAMPLE 2

EXAMPLE 3

This example describes a file server (display) shut down. The Clients have a chance within the Maximum time for removal to store their data.

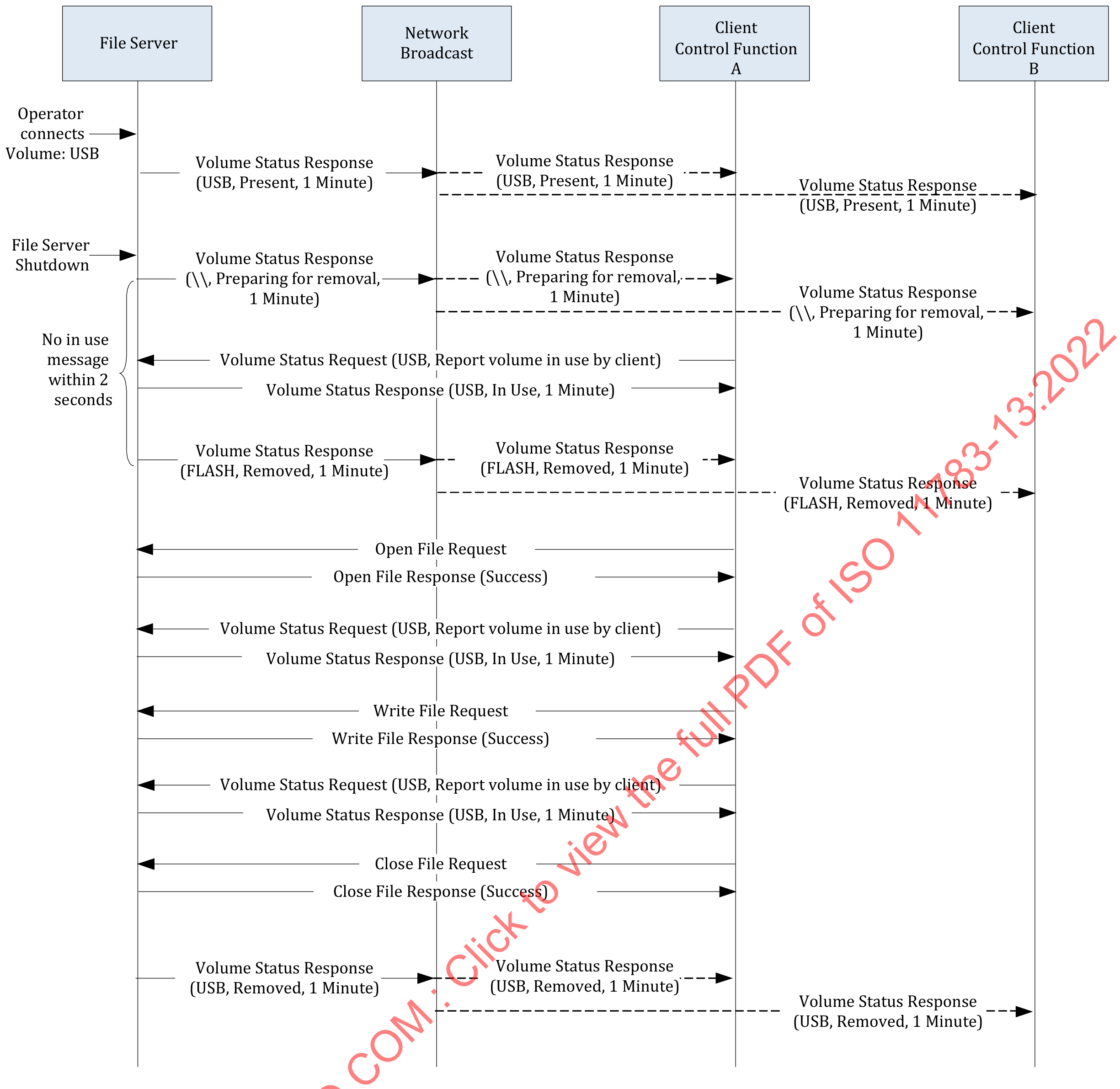
File Server
Network Broadcast
Client Control Function A
Client Control Function B
Operator connects Volume: USB
Volume Status Response (USB, Present, 1 Minute)
Volume Status Response (USB, Present, 1 Minute)
Volume Status Response (USB, Present, 1 Minute)
File Server Shutdown
Volume Status Response (\\, Preparing for removal, 1 Minute)
Volume Status Response (\\, Preparing for removal, 1 Minute)
Volume Status Response (\\, Preparing for removal, 1 Minute)
No in use message within 2 seconds
Volume Status Request (USB, Report volume in use by client)
Volume Status Response (USB, In Use, 1 Minute)
Volume Status Response (FLASH, Removed, 1 Minute)
Volume Status Response (FLASH, Removed, 1 Minute)
Volume Status Response (FLASH, Removed, 1 Minute)
Open File Request
Open File Response (Success)
Volume Status Request (USB, Report volume in use by client)
Volume Status Response (USB, In Use, 1 Minute)
Write File Request
Write File Response (Success)
Volume Status Request (USB, Report volume in use by client)
Volume Status Response (USB, In Use, 1 Minute)
Close File Request
Close File Response (Success)
Volume Status Response (USB, Removed, 1 Minute)
Volume Status Response (USB, Removed, 1 Minute)
Volume Status Response (USB, Removed, 1 Minute)

Figure C.3 — Volume Status Behaviour EXAMPLE 3

EXAMPLE 4

This example shows how a client can react to store files in case somebody wants to remove the volume.

It includes also the fact, that the client needs some time to collect data before he can write them for example.

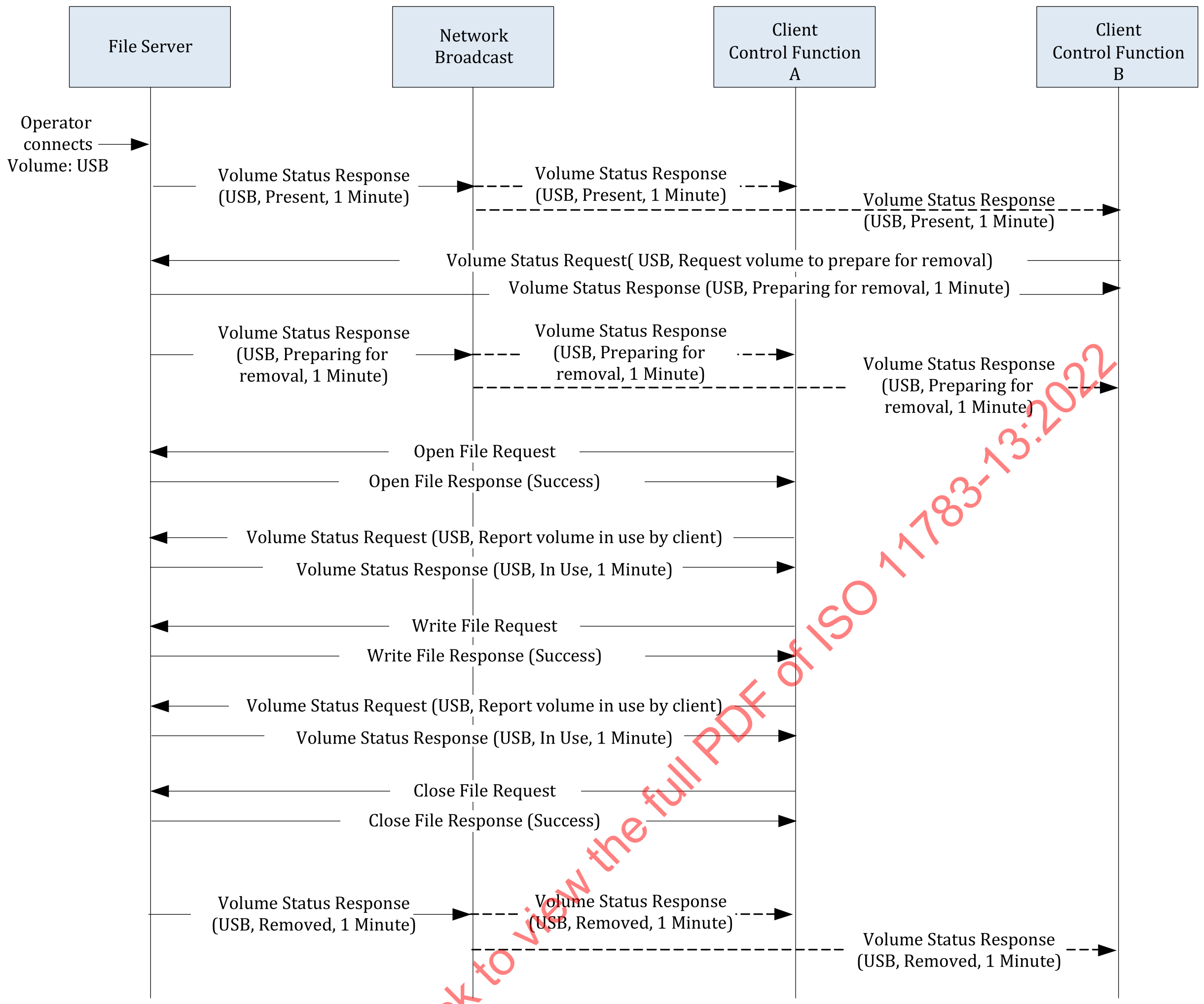
File Server
Network Broadcast
Client Control Function A
Client Control Function B
Operator connects Volume: USB
Volume Status Response (USB, Present, 1 Minute)
Volume Status Response (USB, Present, 1 Minute)
Volume Status Response (USB, Present, 1 Minute)
Volume Status Request(USB, Request volume to prepare for removal)
Volume Status Response (USB, Preparing for removal, 1 Minute)
Volume Status Response (USB, Preparing for removal, 1 Minute)
Volume Status Response (USB, Preparing for removal, 1 Minute)
Volume Status Response (USB, Preparing for removal, 1 Minute)
Open File Request
Open File Response (Success)
Volume Status Request (USB, Report volume in use by client)
Volume Status Response (USB, In Use, 1 Minute)
Write File Request
Write File Response (Success)
Volume Status Request (USB, Report volume in use by client)
Volume Status Response (USB, In Use, 1 Minute)
Close File Request
Close File Response (Success)
Volume Status Response (USB, Removed, 1 Minute)
Volume Status Response (USB, Removed, 1 Minute)
Volume Status Response (USB, Removed, 1 Minute)

Figure C.4 — Volume Status Behaviour EXAMPLE 4

C.2 Directory Handling

C.2.1 Overview

Directory Handling consists of commands to get or set the current directory, which is used for requests where the path argument specifies no directory.

C.2.2 Get Current Directory

C.2.2.1 General

Get Current Directory returns the current directory as a pathname. After successful completion of the Get Current Directory Request, the full path in the form “\\VOL\DIR\SUBDIR” is reported.

C.2.2.2 Get Current Directory Request

Transmission repetition rate:	On request
Data length:	8 bytes
Parameter group number:	Client to FS, destination-specific

Byte 1	FS function = 16_{10}			
	Bits 7–4 0001	Command	Directory Handling	see B.1
	Bits 3–0 0000	Function	Get Current Directory	see B.2
Byte 2		TAN		see B.8
Bytes 3–8		Reserved, transmit as FF_{16}		

C.2.2.3 Get Current Directory Response

Transmission repetition rate:	In response to Get Current Directory Request message
Data length:	Variable
Parameter group number:	FS to client, destination-specific

Byte 1	FS function = 16_{10}			
	Bits 7–4 0001	Command	Directory Handling	see B.1
	Bits 3–0 0000	Function	Get Current Directory	see B.2
Byte 2		TAN		see B.8
Byte 3		Error Code		see B.9
		0_{10}	Success	
		4_{10}	File, Path or volume not found	
		10_{10}	Media is not present	
		13_{10}	Volume is possibly not initialized	
		43_{10}	Out of memory	
		44_{10}	Any other error	
Bytes 4–7		Total Space (in units of 512 bytes)		see B.11
		Only if detectable. If not detectable 0 shall be reported		
Bytes 8–11		Free Space (in units of 512 bytes)		see B.11
		Only if detectable. If not detectable 0 shall be reported		
Bytes 12,13		Path Name Length		see B.12
Bytes 14–*n*		Path Name		see B.13
		Contains the current directory path, regardless of the error code.		

C.2.3 Change Current Directory

C.2.3.1 General

Change Current Directory selects the current directory.

C.2.3.2 Change Current Directory Request

Transmission repetition rate:		On request		
Data length:		Variable		
Parameter group number:		Client to FS, destination-specific		
Byte 1	FS function = 17_{10}			
	Bits 7–4 0001	Command	Directory Handling	see B.1
	Bits 3–0 0001	Function	Change Current Directory	see B.2
Byte 2		TAN		see B.8
Bytes 3,4		Path Name Length		see B.12
Bytes 5–*n*		Path Name		see B.13

C.2.3.3 Change Current Directory Response

Transmission repetition rate:		In response to Change Current Directory Request message		
Data length:		8 bytes		
Parameter group number:		FS to client, destination-specific		
Byte 1	FS function = 17_{10}			
	Bits 7–4 0001	Command	Directory Handling	see B.1
	Bits 3–0 0001	Function	Change Current Directory	see B.2
Byte 2		TAN		see B.8
Byte 3		Error Code		see B.9
		0_{10}	Success	
		1_{10}	Access denied	
		2_{10}	Invalid access	
			The request path is a file (not a directory).	
		4_{10}	File, path or volume not found	
		7_{10}	Invalid destination name given	
		10_{10}	Media is not present	
		13_{10}	Volume is possibly not initialized	
		43_{10}	Out of memory	
		44_{10}	Any other error	
Bytes 4–8		Reserved, transmit as FF_{16}		

C.3 File Access

C.3.1 Overview

File Access consists of messages for opening and closing files, for navigating through files and for reading and writing data to files.

C.3.2 Open File

C.3.2.1 General

Creating a file/folder using the Create File command is accomplished by using the Open File command with the Flag set to create a new file.

If a file is not existing and is created with the flag "open for reading only" and "create if not existing", an empty file will be created which cannot be written to from the returned handle.

When creating files or folders, all parent directories in the path shall be created by the file server, if not existing.

The file server shall not allow multiple simultaneous write access to a single file. If a file is already open for write access, then, when replying to all further write access requests (for writing only/for reading and writing), the response message shall contain the error code "Access Denied".

If an existing file is opened with the flags being set to "open file for writing-only" and "open file for random access", the content of the existing file will be deleted by the file server

Paths with wildcards are only allowed to open a directory and to filter the content that can be read. The wildcard needs to be included at the end of the path. In all other cases wildcards are not allowed for this command, and the response shall include error code 6, invalid given source name.

EXAMPLE 1

\\Vol\DirectoryToRead\*.txt → Executing the read command on such an opened handle will only read back all items that end with ".txt"

EXAMPLE 2

\\Vol\DirectoryToRead\ite?ms → will read only items like "ite1ms", "iteims", ...

C.3.2.2 Open File Request

Transmission repetition rate:		On request		
Data length:		Variable		
Parameter group number:		Client to FS, destination-specific		
Byte 1	FS function = 32_{10}			
	Bits 7–4 0010	Command	File Access	see B.1
	Bits 3–0 0000	Function	Open File	see B.2
Byte 2		TAN		see B.8
Byte 3		Flags		see B.14
Bytes 4,5		Path Name Length		see B.12
Bytes 6–*n*		Name		see below

The valid format of the Name parameter depends on the Flags parameter.

If the Flags parameter bit 0 and 1 indicates ‘Open file for reading only’, ‘Open file for writing only’ or ‘Open file for reading and writing’ then the Name parameter shall be a Filename (see A.2.3.2).

If the Flags parameter bit 0 and 1 indicates ‘Open directory’ and bit 2 indicates ‘Open an existing file’ then the Name parameter shall be either a Path (see A.2.3.1) or a Wildcard name (see A.2.3.3).

If the Flags parameter bit 0 and 1 indicates ‘Open directory’ and bit 2 indicates ‘Create new file and/or directories if not yet existing’ then the Name parameter shall be a Path (see A.2.3.1).

When an Open File request with flags set for “Open file for reading only” and “Open for random access” is received the file server shall:

— set the file pointer to the begin of the file;

— read data at the current file pointer position.

When an Open File request with flags set for “Open file for reading only” and “Open file for appending data to the end of the file” is received the file server shall:

— set the file pointer to the end of the file;

— read data at the current file pointer position.

When an Open File request with flags set for “Open file for writing only” and “Open for random access” is received the file server shall:

— set the file pointer to the begin of the file;

— write data at the current file pointer position;

— resize the file size to zero so that the file content, before the file was opened, is deleted.

When an Open File request with flags set for “Open file for writing only” and “Open file for appending data to the end of the file” is received the file server shall:

— set the file pointer to the end of the file;

— write data always at the end of the file by moving first file pointer to the end of the file and writing data at this position.

When an Open File request with flags set for “Open file for reading and writing” and “Open for random access” is received the file server shall:

— set the file pointer to the begin of the file;

— read and write data at the current file pointer position.

When an Open File request with flags set for “Open file for reading and writing” and “Open file for appending data to the end of the file” is received the file server shall:

— set the file pointer to the end of the file;

— read data at the current file pointer position;

— write data always at the end of the file by moving first file pointer to the end of the file and writing data at this position.

EXAMPLE 1

A client sends an Open File Request with flags indicating “Open directory” and “Open an existing file”, The Name parameter is set to “\\volume\folder1\”.

Subsequent Read File Requests will return Directory Entries for the files and folders in “\\volume\folder1\”.

EXAMPLE 2

A client sends an Open File Request with flags indicating “Open directory” and “Open an existing file”, The Name parameter is set to “\\volume\folder1”.

Subsequent Read File Requests will return Directory Entries for the files and folders in “\\volume\”, using “folder1” as a search pattern. In other words, it will return 1 Directory Entry with information about “folder1”.

C.3.2.3 Open File Response

Transmission repetition rate:		In response to Open File Request message		
Data length:		8 bytes		
Parameter group number:		FS to client, destination-specific		
Byte 1	FS function = 32_{10}			
	Bits 7–4 0010	Command	File Access	see B.1
	Bits 3–0 0000	Function	Open File	see B.2
Byte 2		TAN		see B.8
Byte 3		Error Code		see B.9
		0_{10}	Success	
		1_{10}	Access denied	
			Client tries to create a new volume in the volume list (Path only contains a volume name)	
			Try to open a read only file for writing only OR reading and writing	
			Open file for writing only OR reading and writing that is already open for write access	
			Path is already in use with exclusive access flag set	
			Path is already in use and cannot be open with exclusive access flag set.	
			Volume is mounted as read only and flag “create if not exist” is set or open for writing only OR reading and writing is set.	
		2_{10}	Invalid access	
			The given path doesn’t fit to the target type (open file and provide a directory path, or open directory and provide a file path)	
			Open a file with a wildcard in the path name	
		3_{10}	Too many files open	
		4_{10}	File, path or volume not found	

		Client tries to open a directory or file which doesn’t exist without create if not exist flag set.	
		Path refers to a volume that doesn’t exist, independent of the create if not exist flag set	
	6_{10}	Invalid given source name.	
		Client tries to create a file or directory with a wildcard as path name.	
	8_{10}	Volume out of free space	
		Volume is full	
		Volume is read only, and existing files shall be open for writing only or reading and writing	
	10_{10}	Media is not present	
	13_{10}	Volume is possibly not initialized	
	43_{10}	Out of memory	
	44_{10}	Any other error	
Byte 4	Handle		see B.10
Byte 5	Attributes		see B.15
Bytes 6–8	Reserved, transmit as FF_{16}		

C.3.3 Seek File

C.3.3.1 General

Seek File sets the file pointer for the next access within a file. Depending on the position mode, the file pointer can be set relative to the beginning of the file, the current file pointer or the end of the file. If the function succeeds, the new position is returned. The new file position is based on the following position mode:

0 New position = beginning of file + offset (can only be positive or 0 value);

1 New position = current position + offset (can be positive or negative value);

2 New position = end of file + offset (can only be negative or 0 value).

When the Handle references a file, the offset and position are defined in bytes. When the Handle references a directory, the offset and position are defined in Directory Entries.

When the file pointer is at the end of file position and a request to move the file pointer past the end of file is sent, the response shall contain a “File pointer at end of file” error code.

If the Seek command tries to set the file pointer beyond the end of file (and the file pointer is not already at end of file) or before the start of the file, the response shall contain an “Invalid Request Length” error code.

If the response contains an error code, the pointer position is not changed and stays at the same position as before the request execution.

The File pointer position shall be changed only if the error code “Success” is contained in the response message.

In version 4 and later, bit 5 in the Flags attribute in the Open File command (see C.3.3.2) specifies whether files with the Hidden attribute shall be counted. In version 3 and prior this was not specified.

C.3.3.2 Seek File Request

Transmission repetition rate:		On request		
Data length:		8 bytes		
Parameter group number:		Client to FS, destination-specific		
Byte 1	FS function = 33_{10}			
	Bits 7–4 0010	Command	File Access	see B.1
	Bits 3–0 0001	Function	Seek File	see B.2
Byte 2		TAN		see B.8
Byte 3		Handle		see B.10
Byte 4		Position Mode		see B.17
Bytes 5–8		Offset		see B.18

C.3.3.3 Seek File Response

Transmission repetition rate:		In response to Seek File Request message		
Data length:		8 bytes		
Parameter group number:		FS to client, destination-specific		
Byte 1	FS function = 33_{10}			
	Bits 7–4 0010	Command	File Access	see B.1
	Bits 3–0 0001	Function	Seek File	see B.2
Byte 2		TAN		see B.8
Byte 3		Error Code		see B.9

0_{10}	Success.
1_{10}	Access denied. Handle is not assigned to requester.
5_{10}	Invalid Handle.
11_{10}	Failure during a read operation. File server is not able to move the file pointer.
42_{10}	Invalid request length. Given offset would result in a file pointer position outside the file, while the current file pointer position is not EOF.

		43_{10}	Out of memory.	
		44_{10}	Any other error.	
		45_{10}	File pointer at end of file.	
			File pointer is at end of file and a new request was received to go further.	
Byte 4		Reserved, transmit as FF_{16}		
Bytes 5–8		Position		see B.19

C.3.4 Read File

C.3.4.1 General

Read File reads data from the file referenced by the Handle. If the Handle refers to a file, Count specifies the number of data bytes to be read. The requested data (excluding the other parameters) is sent in the response (up to 1 780 bytes when TP is used, up to 65 530 bytes when ETP is used). The number of data bytes read can be less than requested if the end of the file is reached. If the Handle refers to a directory, Count specifies the number of directory entries to be read, while Report Hidden Files specifies whether files with attribute set to "hidden" are part of the directory entries listing.

When the file pointer is at the end of file position and a request to read past end of file is sent, the response shall contain a "File pointer at end of file" error code.

C.3.4.2 Read File Request

Transmission repetition rate:		On request		
Data length:		8 bytes		
Parameter group number:		Client to FS, destination-specific		
Byte 1	FS function = 34_{10}			
	Bits 7–4 0010	Command	File Access	see B.1
	Bits 3–0 0010	Function	Read File	see B.2
Byte 2		TAN		see B.8
Byte 3		Handle		see B.10
Bytes 4,5		Count		see B.20
Byte 6		**Version 4 and later:**		
		Reserved, transmit as FF_{16}		
		Version 3 and prior:		
		Report Hidden Files		see B.28
Bytes 7,8		Reserved, transmit as FF_{16}		

Version 4 and later file servers are allowed to ignore byte 6, regardless of version of the client.

C.3.4.3 Read File Response (handle-referenced file)

The Read File Response message contains the data read from a file referred to by the Handle specified in the Read File Request message.

Transmission repetition rate:		In response to Read File Request message		
Data length:		Variable		
Parameter group number:		FS to client, destination-specific		
Byte 1	FS function = 34_{10}			
	Bits 7–4 0010	Command	File Access	see B.1
	Bits 3–0 0010	Function	Read File	see B.2
Byte 2		TAN		see B.8
Byte 3		Error Code		see B.9
		0_{10}	Success	
		1_{10}	Access denied	
			Handle is not assigned to requester.	
			File was opened for writing only.	
		5_{10}	Invalid Handle	
		11_{10}	Failure during a read operation	
		43_{10}	Out of memory	
		44_{10}	Any other error	
		45_{10}	File pointer at end of file	
			File pointer is at end of file when the request is executed.	
Bytes 4,5		Count		see B.20
Bytes 6–*n*		Data		

C.3.4.4 Read Directory Response (handle-referenced directory)

The Read Directory Response message contains the directory entries read from a directory, referred to by the Handle specified in the Read File Request message.

Transmission repetition rate:		In response to Read File Request message		
Data length:		Variable		
Parameter group number:		FS to client, destination-specific		
Byte 1	FS function = 34_{10}			
	Bits 7–4 0010	Command	File Access	see B.1

	Bits 3–0 0010	Function	Read File	see B.2
Byte 2		TAN		see B.8
Byte 3		Error Code		see B.9
		0_{10}	Success	
		1_{10}	Access denied	
			Handle is not assigned to requester.	
		5_{10}	Invalid Handle	
		11_{10}	Failure during a read operation	
		43_{10}	Out of memory	
		44_{10}	Any other error	
		45_{10}	File pointer at end of file	
			File pointer is at end of file when the request is executed.	
Bytes 4,5		Count		see B.20
Bytes 6–*n*		Directory Entries		see B.21

C.3.5 Write File

C.3.5.1 General

Write File writes data to an open file that is addressed by a Handle. The data (excluding the other parameters) to be written is sent to the FS in the request (up to 1 780 bytes when TP is used and up to 65 530 bytes when ETP is used). The Write File command shall not be used with a Handle that references a directory.

C.3.5.2 Write File Request

Transmission repetition rate:	On request
Data length:	Variable
Parameter group number:	Client to FS, destination-specific

Byte 1	FS function = 35_{10}			
	Bits 7–4 0010	Command	File Access	see B.1
	Bits 3–0 0011	Function	Write File	see B.2
Byte 2		TAN		see B.8
Byte 3		Handle		see B.10
Bytes 4,5		Count		see B.20
Bytes 6–*n*		Data		

C.3.5.3 Write File Response

Transmission repetition rate:		In response to Write File Request message		
Data length:		8 bytes		
Parameter group number:		FS to client, destination-specific		
Byte 1	FS function = 35_{10}			
	Bits 7–4 0010	Command	File Access	see B.1
	Bits 3–0 0011	Function	Write File	see B.2
Byte 2		TAN		see B.8
Byte 3		Error Code		see B.9
		0_{10}	Success	
		1_{10}	Access denied	
			Handle is not assigned to requester.	
			File was opened for reading only	
			Handle references a directory, volume or volume list	
		5_{10}	Invalid Handle	
		8_{10}	Volume out of free space	
		9_{10}	Failure during a write operation	
		43_{10}	Out of memory	
		44_{10}	Any other error	
Bytes 4,5		Count		see B.20
Bytes 6–8		Reserved, transmit as FF_{16}		

C.3.6 Close File

Close File closes the file specified by the Handle. All internal buffers belonging to this file are written and the directory entry is updated. The Handle is invalid after the Close File Response message.

C.3.6.1 Close File Request

Transmission repetition rate:		On request		
Data length:		8 bytes		
Parameter group number:		Client to FS, destination-specific		
Byte 1	FS function = 36_{10}			
	Bits 7–4 0010	Command	File Access	see B.1
	Bits 3–0 0100	Function	Close File	see B.2

Byte 2		TAN		see B.8
Byte 3		Handle		see B.10
Bytes 4–8		Reserved, transmit as FF_{16}		

C.3.6.2 Close File Response

Transmission repetition rate:		In response to Close File Request message		
Data length:		8 bytes		
Parameter group number:		FS to client, destination-specific		
Byte 1	FS function = 36_{10}			
	Bits 7–4 0010	Command	File Access	see B.1
	Bits 3–0 0100	Function	Close File	see B.2
Byte 2		TAN		see B.8
Byte 3		Error Code		see B.9
		0_{10}	Success	
		1_{10}	Access denied	
			Handle is not assigned to requester.	
		5_{10}	Invalid Handle	
			Handle doesn't exist.	
		8_{10}	Volume out of free space	
			File server was not able to store buffered data before closing the handle.	
		9_{10}	Failure during a write operation	
		43_{10}	Out of memory	
		44_{10}	Any other error	
Bytes 4–8		Reserved, transmit as FF_{16}		

C.4 File handling

C.4.1 Overview

File Handling consists of messages used to move, copy and delete files, get or set attributes and get the date of a file.

C.4.2 Move File

C.4.2.1 General

Move File moves or copies a file from its current location to a new location. The type of action is specified by the File Handling Mode and destination specification:

a) if the destination filename differs from the file's present name, the file is renamed;

b) if the destination path differs from the source path, the file is moved;

c) if the destination path contains directories that do not exist, those directories are created;

d) if the “copy” mode is set, the file is copied.

If the directory or file exists in the destination path, an Error Code “Access denied” shall be returned unless the *force* mode is set. If a handle exists for such files, the file server cannot overwrite them even if the *force* mode is set. The *recursive* mode is required to move or copy a directory that contains further directories or files. In this case, the files and subfolders are merged with the existing target folder. Existing files are overwritten if the force mode is set. If, in this case, the recursive mode is not set, an Error Code “Access denied” shall be returned. If the destination of a recursive move or copy is in the source path, then an Error Code “Access denied” shall be returned. When specifying a directory, add a “\” to the end to indicate in the Source or Destination Path Name. If the destination of a recursive move or copy is a subfolder of the source path, then an Error Code “Access denied” shall be returned.

The file server cannot move files or directories where a handle exists for. In all cases where an existing handle is the cause to not complete the command, the response message shall contain the error code “Access Denied”.

In case of the volume root \\volumename\ as source of a move command the file server shall not modify the filesystem and the response message shall contain the error code “Access Denied”, because the root folder itself cannot be moved.

To execute the move command on the content of the volume root, \\volumename* shall be used.

In case of a recursive move command, it is up to the file server implementation if the command will be stopped at the first issue, or if the system runs the command as long as files or folders can be moved.

EXAMPLES

FileA → FileB	Rename FileA into FileB
FileA → FolderB\	Move FileA into FolderB -→ FolderB\FileA
FolderA\ → FileB	Illegal, a folder cannot be a file after the move operation
FileA → FolderA\FileA	Move FileA into FolderA →- FolderA\FileA
FileA →FolderA\FileB	Move FileA into FolderA and rename FileA into FileB → FolderA\FileB
FolderA\ →FolderB\	Rename FolderA into FolderB
FolderA\ →FolderB\FolderA\	Move FolderA into FolderB
FolderA\ →FolderB\FolderC\	Move FolderA into FolderB and rename FolderA into FolderC
FolderA\FolderB\ →FolderC\FolderD\	Move FolderB from FolderA into FolderC and rename FolderB into FolderD
FolderA\b → FolderC\b	Move file b from FolderA into FolderC
FolderV\v* → FolderX\	Move everything starting with v from FolderV into FolderX