Consultative Committee for Space Data Systems

RECOMMENDATION FOR SPACE DATA SYSTEM STANDARDS

SPACE COMMUNICATIONS PROTOCOL SPECIFICATION (SCPS)—

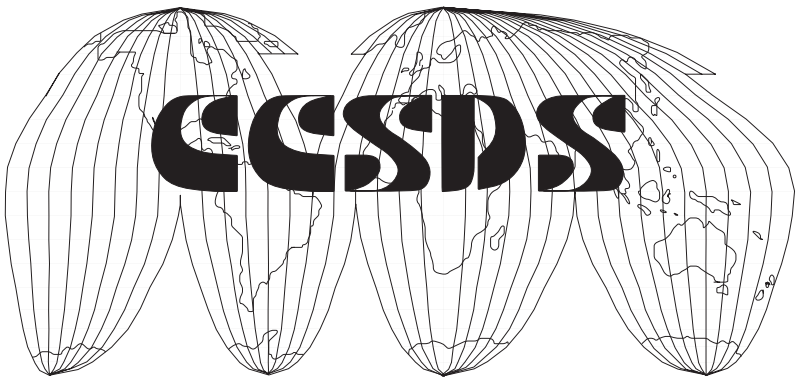
FILE PROTOCOL

(SCPS-FP)

CCSDS 717.0-B-1

BLUE BOOK

May 1999



AUTHORITY

Issue:	Blue Book, Issue 1
Date:	May 1999
Location:	Newport Beach, California, USA

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS Recommendations is detailed in reference [B1], and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This Recommendation is published and maintained by:

CCSDS Secretariat
Program Integration Division (Code MT)
National Aeronautics and Space Administration
Washington, DC 20546, USA

STATEMENT OF INTENT

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of member space Agencies. The Committee meets periodically to address data systems problems that are common to all participants, and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of Committee actions are termed **Recommendations** and are not considered binding on any Agency.

This **Recommendation** is issued by, and represents the consensus of, the CCSDS Plenary body. Agency endorsement of this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

- Whenever an Agency establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommendation**. Establishing such a **standard** does not preclude other provisions which an Agency may develop.

- Whenever an Agency establishes a CCSDS-related **standard**, the Agency will provide other CCSDS member Agencies with the following information:

  -- The **standard** itself.

  -- The anticipated date of initial operational capability.

  -- The anticipated duration of operational service.

- Specific service arrangements shall be made via memoranda of agreement. Neither this **Recommendation** nor any ensuing **standard** is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommendation** will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or, (3) be retired or canceled.

In those instances when a new version of a **Recommendation** is issued, existing CCSDS-related Agency standards and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each Agency to determine when such standards or implementations are to be modified. Each Agency is, however, strongly encouraged to direct planning for its new standards and implementations towards the later version of the Recommendation.

FOREWORD

The Space Communications Protocol Specification (SCPS) File Protocol (SCPS-FP) provides End-to-End Application-Layer Services for flight missions in the space environment. It is based on Internet File Transfer Protocol (FTP) and is generally inter-operable with it. It does, however, extend and modify FTP in order to meet the unique requirements of space flight operations. This document anticipates reader familiarity with FTP and, therefore, addresses only those capabilities that have been added to FTP.

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommendation is therefore subject to CCSDS document management and change control procedures as defined in reference [B1]. Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this document should be addressed to the CCSDS Secretariat at the address indicated on page i.

At time of publication, the active Member and Observer Agencies of the CCSDS were

Member Agencies

- British National Space Centre (BNSC)/United Kingdom.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- National Aeronautics and Space Administration (NASA)/USA.
- National Space Development Agency of Japan (NASDA)/Japan.
- Russian Space Agency (RSA)/Russian Federation.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- Centro Tecnico Aeroespacial (CTA)/Brazil.
- Chinese Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Communications Research Laboratory (CRL)/Japan.
- Danish Space Research Institute (DSRI)/Denmark.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Federal Service of Scientific, Technical & Cultural Affairs (FSST&CA)/Belgium.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Industry Canada/Communications Research Centre (CRC)/Canada.
- Institute of Space and Astronautical Science (ISAS)/Japan.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Korea Aerospace Research Institute (KARI)/Korea.
- Ministry of Communications (MOC)/Israel.
- National Oceanic & Atmospheric Administration (NOAA)/USA.
- National Space Program Office (NSPO)/Taipei.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 717.0-B-1	Space Communications Protocol Specification (SCPS)—File Protocol (SCPS-FP)	May 1999	Original issue

CONTENTS

CONTENTS (continued)

Section Page

Figure

Table

1 INTRODUCTION

1.1 PURPOSE

The purpose of this Recommendation is to define the Space Communications Protocol Specification (SCPS) File Protocol (SCPS-FP). This definition will allow independent implementations of the SCPS-FP in the space and ground segments of the SCPS Network to interoperate.

1.2 SCOPE

This Recommendation addresses communications between space platforms and ground systems, and between space platforms. It is designed to support efficient file-transfer and file-operation requirements of current and upcoming space missions through extensions and modifications to Internet File Transfer Protocol (FTP).

1.3 APPLICABILITY

This Recommendation is applicable to space missions and ground systems that claim conformance to the SCPS-FP. It is intended that this Recommendation become a uniform standard for all active Member and Observer Agencies of the CCSDS.

1.4 ORGANIZATION OF RECOMMENDATION

This SCPS-FP Recommendation is organized as follows:

- Section 1 provides an introduction to the standard, introduces new protocol terminology, and documents the normative references.
- Section 2 provides an overview of the SCPS File Protocol.
- Section 3 documents the protocol specifications for a SCPS-FP.
- Section 4 documents the Management Information Base (MIB) requirements for SCPS-FP.
- Section 5 documents the conformance requirements for SCPS-FP.
- Annex A defines acronyms and abbreviations used in the document.
- Annex B lists informative references.
- Annex C contains the Protocol Implementation Conformance Statement (PICS) proforma for SCPS-FP.
- Annex D contains the Service Specification.

This Recommendation modifies and adds to Internet FTP. It makes normative reference to the Internet standards that define FTP (references [1] and [2]). It makes informative reference to other documents (see annex B).

1.5 TERMINOLOGY

The terminology as outlined in Internet Standard 9 (reference [1]), section 2.2, is applicable to the SCPS-FP with the following additions:

Cyclic Redundancy Check (CRC): A polynomial-based code used for protecting against recording and communications errors (see reference [B2]).

control data: A collection of the information necessary to control the operation of the record read and record update services. This information includes data such as the files to be read/updated, checksum, option flags, and update data.

network byte order: The order in which the bytes of a word are transmitted. For a 32-bit word, W1.W2.W3.W4, where W1 is the most significant octet, W1 is transmitted first, W2 next, then W3, and finally W4. The same applies for a 16-bit number, W1.W2, where W1 corresponds to the high 8 bits. W1 is sent first, and then W2.

rollback: Of or pertaining to the act of rolling back (undoing) incomplete changes made to a file for which the transfer or record operation terminated with errors and thereby restoring the file to its pre-transfer or pre–record operation state.

user-FP: Same as user-FTP in Internet Standard 9 (reference [1]). Terminology changed in SCPS-FP specification to remove direct reference to FTP.

server-FP: Same as server-FTP in Internet Standard 9 (reference [1]). Terminology changed in SCPS-FP specification to remove direct reference to FTP.

NVT-ASCII: The ASCII collation sequence.

1.6 REFERENCES

The following documents contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Recommendation are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS Recommendations.

[1] J. Postel and J. Reynolds. *File Transfer Protocol*. STD 9, October 1985. [RFC 959]†

[2] R. Braden. *Hosts Requirements*. STD 3, October 1989. [RFC 1122, RFC 1123]

[3] *Packet Telemetry*. Recommendation for Space Data System Standards, CCSDS 102.0-B-4. Blue Book. Issue 4. Washington, D.C.: CCSDS, November 1995.

[4] *Space Communication Protocol Specification (SCPS)—Transport Protocol (SCPS-TP).* Recommendation for Space Data System Standards, CCSDS 714.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 1999.

† Internet Request for Comments (RFC) texts are available on line in various locations (e.g., http://ietf.org/rfc/). In this list, Internet Standards are identified by 'STD' followed by the number of the standard, and RFCs are identified by 'RFC' followed by the number of the RFC. RFCs comprised by Internet Standards are given in square brackets following the citation.

2 OVERVIEW

The extensions to FTP and its augmentations as contained in Internet Standards 9 and 3 (references [1] and [2]) are necessitated by technical requirements for transfers of and operations on files and analogous delimited data sets in the space communications environment. These technical requirements include

- transfers of command and data files to spacecraft;
- transfers of application software to spacecraft;
- transfers of science or mission data to ground without special level-zero processing;
- limited management of files onboard spacecraft (delete, rename, and directory services);
- automatic restart of transfers after an interruption;
- reading of portions of files resident on board spacecraft; and
- updates/changes to files on board spacecraft.

The SCPS-TP protocol is characterized by a command/reply dialog on a stream connection. The client Protocol Interpreter (PI) accepts input from the user, translates user commands to server commands, and issues the server commands on the control connection. The server PI parses those commands, looks up commands in a table, and transfers control to the routine associated with the command. This specification defines procedures for each PI to execute upon success or failure of commands. When a command has been parsed and processed, either successfully or unsuccessfully, the server issues a reply message on the control connection. Commands and replies are short ASCII strings terminated with the ASCII control sequence <CRLF>.

SCPS-FP is based on Internet FTP, which is described in references [1] and [2]. SCPS-FP maintains interoperability with Internet FTP on the control connection. This Recommendation is designed to be scaleable to fit onto the most resource-constrained service and yet expandable to full Internet compatibility according to mission requirements.

SCPS-FP is a profile of Internet FTP that adds capabilities to the Internet standard as follows:

- automatic restart of a failed transfer;
- read, write, add, change, and delete individual file records;
- interrupt (pause) a file transfer so that it can be restarted later from the interrupt point;
- suppress reply text to improve bandwidth efficiency;
- assure file and record integrity in the case of abnormal connection loss.

These capabilities are designed to yield efficient transfer of file-oriented data sets across the space link.

3 TECHNICAL SPECIFICATION

3.1 INTERNET FTP

SCPS-FP adopts the File Transfer Protocol (FTP) as specified in Internet Standards 9 and 3 (references [1] and [2]), with the modifications and extensions specified in section 3 of this document.

3.2 DATA TRANSFER FUNCTIONS

3.2.1 DATA REPRESENTATION AND STORAGE

3.2.1.1 Data Types

3.2.1.1.1 General

The non-print format controls are applicable to SCPS-FP. EBCDIC and Local data types may be supported as additional, non-SCPS capabilities.

3.2.1.1.2 ASCII Type

The ASCII data type is optional for SCPS-FP file transfer operations; it does not apply to SCPS-FP record operations. The ASCII type is not the SCPS-FP default.

3.2.1.1.3 Image Type

The IMAGE data type is the default for SCPS-FP and must be supported by all SCPS-FP implementations.

3.2.1.2 Data Connection Management

The SCPS-FP user-PI must initiate the use of non-default ports because of the need for SCPS-TP to hold the connection record for a timeout period to guarantee reliable communication.

3.2.1.3 Stream Mode

If the structure is a file structure, the EOF is indicated by the sending host’s closing of the data connection, in which case all bytes are data bytes.

3.2.1.4 CCSDS Packets

If the structure is a record structure, indicating the file consists of contiguous CCSDS Packets, the following provisions are required to enable use of FP restart and record read/update capabilities.

a) the FP shall issue the STRU command with the Record argument (ASCII character 'R');

b) the FP shall issue the TYPE command if necessary to switch to IMAGE data type before transferring files of CCSDS Packets;

c) the FP shall invoke the record boundary option of SCPS-TP (reference [4]) to delimit records during transfer;

d) in record operations, record identifiers shall be interpreted as referring to the Packet Sequence Count field in the record;

e) the protocol shall use the Packet Data Length field to delimit records in the file.

3.2.2 ERROR RECOVERY AND RESTART

3.2.2.1 Stream Mode Restart with IMAGE Data Type

3.2.2.1.1 When a system failure is detected, the receiving Data Transfer Process (DTP) shall determine the restart marker by querying the file system for the current size of the file.

3.2.2.1.2 Restart markers shall not be imbedded in the data in stream mode.

3.2.2.1.3 There are three cases in which SCPS-FP restart may be accomplished in stream mode:

a) user-to-server transfer (e.g., STOR):

   1) the user-FP shall send 'SIZE pathname' to the receiving server-FP;

   2) the server-FP shall send '213 pathname SIZE rrrr' to the user-FP, where rrrr is the size of the file (indicated by pathname) stored locally at the server;

   3) to restart the transfer, the user-FP shall reposition its local file system using restart marker rrrr and send the command 'REST rrrr' to the server-FP;

   4) the server-FP shall save restart marker rrrr for restart;

b) server-to-user transfer (e.g., RETR):

   1) the user-FP shall determine the size (rrrr) of the file as stored by the local file system;

2) to restart the transfer, the user-FP shall reposition its local file system using restart marker rrrr and send the command 'REST rrrr' to the server-FP;

3) the server-FP shall save restart marker rrrr for restart;

c) server-to-server ('third-party') transfer (e.g., PROXY):

1) the user-FP shall send 'SIZE pathname' to the receiving server-FP;

2) the receiving server-FP shall send '213 pathname SIZE rrrr' to the user-FP, where rrrr is the size of the file (indicated by pathname) stored locally at the receiving server;

3) to restart the transfer, the user-FP shall

– send the command 'REST rrrr' to the receiving server-FP;

– send 'REST ssss' to the sending server-FP, where ssss = rrrr;

4) each server shall save the received restart markers for restart.

3.2.2.1.4 Immediately after sending the REST command, the user-FP shall send the file transfer command (such as RETR, STOR or LIST) that was being executed when the system failure occurred.

3.2.2.1.5 Before the actual data transfer resumes, the server-FP process for the file transfer command shall advance the file to the restart marker.

3.2.2.1.6 The automatic and manual restart of a CCSDS Packet file transfer (RETR or STOR of files having the CCSDS Packet structure) shall follow the processing documented in 3.2.2.2 and 3.2.2.4 with the following additional provisions:

a) restart must be on the CCSDS Packet boundary for transfers of files having the CCSDS Packet structure;

b) the restart marker shall be the Packet Sequence Count of the last complete Packet received.

NOTE – General provisions for operations involving the CCSDS Packet structure are given in 3.2.1.4.

3.2.2.2 Stream Mode Restart with ASCII Data Type

When the data type is ASCII, the restart point is always relative to the start of the transfer image, not the flat file size in the file system. All other processing is the same as with the IMAGE data type.

3.2.2.3 Automatic Restart

3.2.2.3.1 For SCPS-FP restart cases, the user Protocol Interpreter (user-PI), rather than the user-FP, shall be responsible for the decision making for

- reestablishing connections,
- re-logging in the user,
- sending the SIZE and/or REST commands,
- interpreting and responding to the replies for these commands, and
- resending the original file transfer command (e.g., RETR, STOR, LIST) to reinitiate the file transfer.

3.2.2.3.2 To use the autorestart capability, the user-FP and the server-FP must both be in autorestart mode:

a) the autorestart protocol command (ARST) shall be used to enable autorestart at the server-FP;

b) the no-autorestart protocol command (NARS) shall be used to disable autorestart at the server-FP.

NOTE - As documented in 3.3.2, the ARST command indicates to the server that the client is in autorestart mode and therefore file changes should not be rolled back when an error occurs (otherwise a partial file that has been transferred will be wiped out). Partial files are saved if the data connection aborts or times out. Conversely, the NARS command indicates to the server that the client is not in autorestart mode and that therefore it should roll back changes when the data connection aborts or times out.

3.2.2.3.3 When autorestart is enabled by the user, the user-PI shall restart a failed data transfer (without requiring the user to specify the restart point or command to restart) up to the maximum number of restarts specified in the MIB parameter fpMaxAutorestarts (see 4.6):

a) the maximum number of restarts shall be set to 3 as a default;

b) the user-FP may notify the user of the failed transfer and request confirmation to continue with the restart.

NOTE – User notification may be useful in situations where an automatic restart does not make sense (e.g., spacecraft is out of view).

3.2.2.3.4 Autorestart of a failed transfer in stream mode shall proceed as detailed in 3.2.2.1.

3.2.2.3.5 In cases where the control connection had to be reestablished in order to proceed with the autorestart, any transfer parameters that were set by the user prior to the start of the file transfer should be resent to the server to ensure that the state of the transfer parameters at the server is correct.

NOTE – Block and compressed modes are not required for SCPS-FP. These modes are not universally interoperable. Error recovery and restart for block and compressed modes for SCPS-FP are as described in reference [1], section 3.5, with the added feature that the user-PI can, if enabled, initiate the restart automatically.

3.2.2.4 Manual Interrupt and Manual Restart

NOTE – Manual interrupt provides the capability for the user-FP to stop a file transfer temporarily and then restart it at a later time.

3.2.2.4.1 There are three cases for SCPS-FP interrupt in all modes.

a) user-to-server transfer:

   1) the user-FP shall send ‘INTR’ to the server-FP;

   2) in response, the server-FP shall stop the file transfer and send ‘256 Transfer Interrupted at rrrr’ to the user-FP, where rrrr is the size of the file stored locally at the server;

   3) the user-FP shall save the restart marker rrrr for restart;

   4) the user-FP may reinitiate the transfer process by repositioning its local file system using restart marker rrrr and sending ‘REST rrrr’ to the server-FP;

b) server-to-user transfer:

   1) the user-FP shall send ‘INTR’ to the server-FP;

   2) the server-FP shall stop the file transfer and send ‘256 Transfer Interrupted at ssss’ to the user-FP;

   3) the user-FP shall ignore the restart marker specified in the INTR reply;

   4) the user-FP may reinitiate the transfer process by sending ‘REST rrrr’ to the server-FP, where rrrr is the amount of data received and stored successfully at the user site;

c) server-to-server (‘third-party’) transfer:

   1) the user-FP shall send ‘INTR’ to the sending server-FP;

2) the sending server-FP shall stop the file transmission and send ‘256 Transfer Interrupted at ssss’ to the user-FP;

3) the user-FP shall send ‘INTR’ to the receiving server-FP:

   – the user-FP should wait for the sending server-FP to respond with the 256 reply before sending ‘INTR’ to the receiving server-FP;

   – if the sending server-FP responds with a code other than 256 (e.g., 226 - file transfer was completed before interrupt received), then the file transfer should continue as it normally would in response to the received reply code;

4) in response to ‘INTR’, the receiving server-FP shall stop the file transfer process and send ‘256 Transfer Interrupted at rrrr’ to the user-FP;

5) the user-FP shall ignore the restart marker in the sending server-FP’s reply and use the restart marker in the receiving server-FP’s reply when issuing the REST command;

6) the user-FP may reinitiate the transfer process by sending ‘REST rrrr’ to the receiving server-FP and sending server-FP.

3.2.2.4.2 To restart the interrupted file transfer, the user-FP shall follow the restart procedures outlined in 3.2.2.1.

3.2.3 INTEGRITY

3.2.3.1 File Transfer Integrity

3.2.3.1.1 The user-FP and server-FP shall roll back any incomplete changes made to a file during a non-autorestart RETR or STOR operation that has terminated with errors, thereby restoring the affected file to its pre-transfer state.

3.2.3.1.2 At the start of a transfer, if the destination filename is in use, FP renames that file using a temporary name.

3.2.3.1.3 The rollback algorithm used shall be as detailed in table 3-1 when autorestart is disabled.

Table 3-1: Rollback Algorithm without Autorestart

Destination file existed at start of transfer	Reason for transfer abort	Action by FP
No	ABOR command	Delete partial file
No	INTR command	Keep partial file
No	time-out	Delete partial file
Yes	ABOR command	Restore original file, delete partial file
Yes	INTR command	Keep partial file, delete original file
Yes	time-out	Restore original file, delete partial file

3.2.3.1.4 In autorestart mode, when a transfer aborts because of a timeout, the FP actions shall be as detailed in table 3-2.

Table 3-2: Rollback Algorithm with Autorestart

Destination file existed at start of transfer	Reason for transfer abort	Action by FP
No	time-out	Keep partial file
Yes	time-out	Keep partial file, delete original file

3.2.3.2 Record Data Integrity

3.2.3.2.1 The user-FP and server-FP shall roll back any incomplete changes made to a file during a READ or UPDT operation that has terminated with errors, thereby restoring the affected file to its pre–record operation state.

3.2.3.2.2 To provide a secondary measure of integrity to ensure that a user does not update the wrong file inadvertently, the user-FP and server-FP shall employ the following CRC for the record update service to ensure the data integrity of the accessed files:

a) The encoding shall be defined by the generating polynomial

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

b) Mathematically, the CRC value corresponding to a given file shall be defined by the following procedure.

1) The n bits in the file are considered to be the coefficients of a mod-2 polynomial $M(x)$ of degree n-1. Bit 0 of the first octet corresponds to the $x^{(n-1)}$ term. Bit 7 of the first octet corresponds to the $x^{(n-8)}$ term. Bit 0 of the second octet corresponds to the $x^{(n-9)}$ term. Bit 7 of the second octet corresponds to the $x^{(n-16)}$ term, and so

on. Bit 7 of the last octet corresponds to the least significant bit in $M(x)$. The last octet may need to be padded with zero bits to achieve an integral number of octets.

2) $M(x)$ is divided by $G(x)$ using mod-2 division, producing a remainder $R(x)$ of degree ≤ 31. This mod-2 division is accomplished using the exclusive OR (XOR) operation for subtractions.

3) The coefficients of $G(x)$ and $R(x)$ are considered to be 32-bit sequences.

4) The bit sequence is complemented and the result is the CRC.

3.2.3.3 Files of CCSDS Packets

The rollback and CRC procedures for files of contiguous CCSDS Packets are as described in 3.2.3.1 and 3.2.3.2.

NOTE – General provisions for operations involving the CCSDS Packet structure are given in 3.2.1.4.

3.3 FILE TRANSFER FUNCTIONS

3.3.1 ACCESS CONTROL COMMANDS

Processing for the following access commands for SCPS-FP is as described in section 4.1.1 of Internet Standard 9 (reference [1]):

USER, PASS, CWD, ACCT, CDUP, QUIT.

3.3.2 TRANSFER PARAMETER COMMANDS

3.3.2.1 Processing for the following transfer parameter commands for SCPS-FP is as described in section 4.1.2 of Internet Standard 9 (reference [1]):

MODE, PORT, PASV, TYPE, STRU.

3.3.2.2 The additional SCPS-FP commands are:

a) ENABLE AUTORESTART (ARST):

– The ARST command shall cause the server-FP to set the MIB parameter fpAutorestartEnabled to ‘TRUE’ (enabled).

– While autorestart is enabled, partial files shall not be rolled back when an error occurs.

b) ENABLE BEST EFFORT TRANSPORT SERVICE OPTION (BETS):

NOTE – The BETS command is practicable only when SCPS-TP (reference [4]) provides the underlying transport service.

– The BETS command shall cause the server-FP to set the MIB parameter fpBETSEnabled to 'TRUE' (enabled) and save the BETS fill code to use during a file transfer, if supplied, in the MIB parameter fpBETSFillChar.

– While the BETS option is enabled,

  • the sending FP shall request the SCPS-TP to run in BETS mode for RETR and STOR operations;

  • the receiving FP/TP shall accept BETS-mode operation.

– The receiving FP shall inform the user of any gaps in the data and fill the gaps with the BETS fill code contained in the MIB parameter fpBETSFillChar.

– BETS shall be applied to RETR and STOR operations only.

c) DISABLE AUTORESTART (NARS):

– The NARS command shall cause the server-FP to set the MIB parameter fpAutorestartEnabled to 'FALSE' (disabled).

– While autorestart is disabled, partial files shall be rolled back when an error occurs.

d) DISABLE BEST EFFORT TRANSPORT SERVICE OPTION (NBES):

– The NBES command shall cause the server-FP to set the MIB parameter fpBETSEnabled to 'FALSE' (disabled).

– While the BETS option is disabled,

  • the sending FP shall not request BETS-mode operation;

  • the receiving FP/TP shall not accept BETS-mode operation.

3.3.3 SCPS-FP SERVICE COMMANDS

3.3.3.1 Processing for the following service commands for SCPS-FP is as described in section 4.1.3 of Internet Standard 9 (reference [1]) and as amended by sections 4.1.2.7 (LIST) and 4.1.2.10 (Telnet End-of-Line Code) of RFC 1123 in Internet Standard 3 (reference [2]):

RETR, STOR, RNFR, RNTO, ABOR, DELE, RMD, MKD, LIST, HELP, SITE, STAT, PWD, STOU, SYST, NOOP, NLST, APPE.

3.3.3.2 In addition, SCPS-FP supports the following commands:

a) COPY (COPY):

– The COPY command shall cause the server-FP to copy the file specified in the first pathname (argument 1) to the file specified in the second pathname (argument 2).

– The server-FP shall report the success or failure of the copy in a reply.

b) IDLE TIMEOUT (IDLE):

– The IDLE command shall cause the server-DTP to set the MIB parameter fpIdleTimeout to the value specified by the user.

– If the server is inactive for this idle timeout period, the server-FP shall terminate the process and close the control connection.

c) MANUAL INTERRUPT (INTR):

– The INTR command shall cause the server-DTP to interrupt a file transfer.

– The server-FP shall communicate the point of interrupt to the user-FP via reply message and code.

d) DO NOT SUPPRESS REPLY TEXT (NSUP):

– The NSUP command shall cause the server-FP to set the MIB parameter fpSuppressReplyText to ‘FALSE’ (disabled).

– While suppression is disabled, all replies, including that in response to NSUP, shall have the text included in the reply.

e) RECORD READ (READ):

1) User operations for the READ command are as follows:

– the user-DTP shall create a control file as specified in 3.4.2.2;

– the user-DTP shall then transfer the control file to the server-DTP using the STOR command;

– after transferring the control file, the user-DTP shall issue the READ command, passing the remote control-file name as the argument;

– if the user has specified the forced read option in the user command, the user-FP shall transmit an enabled forced read option flag to the server-FP via the control data;

– if the user has not specified the forced read option in the user command, the user-FP shall transmit a disabled forced read option flag to the server-FP via the control data.

2) The READ command shall cause the server-DTP to transfer one or more records within a remote source file to the user-DTP or receiving server-DTP:

– if the remote source file that is specified in the control data exists at the server, the records identified by the record IDs in the control data shall be transferred and stored locally at the user-DTP or receiving server-DTP;

– the forced read option shall be executed if the user has specified this option in the record read user command;

– the status and contents of the file at the server site shall be unaffected by the record read service.

3) If the forced read option is enabled and either the user-FP or the server-FP encounters a recoverable error, each shall continue the read processing until the read transfer is deemed completed or an irrecoverable error occurs.

4) For file structure a record shall be retrieved based on its position in the file relative to the start of the file.

5) For CCSDS Packet structure a record shall be retrieved based on the value of the CCSDS Packet Sequence Count in the record rather than on its relative position in the file.

NOTE – General provisions for operations involving the CCSDS Packet structure are given in 3.2.1.4.

f) RECORD UPDATE (UPDT):

1) User operations for the UPDT command are as follows:

– prior to issuing the UPDT command to the server-FP, the user-FP shall

• compute a checksum against a local copy of the remote file;

• test the update request against the same file;

– the user-DTP shall create a control file as specified in 3.4.2.2, containing the remote file names, the local checksum, and the update signals;

– the user-DTP shall transfer the control file to the server-DTP using the STOR command;

– the user-DTP shall issue the UPDT command passing the remote control-file name as the argument.

2) The UPDT command shall cause the server-DTP to accept record update data transferred via the control data and apply it to a remote target file:

– the record update data, as specified in 3.4.2.2, shall consist of a series of signals that indicate which records to delete and modify in the remote file and where to add new records;

– if the remote file specified in the control data exists at the server,

• the server-FP first shall verify that the remote target file is the correct file to modify by comparing the remote file's checksum with that provided in the control data;

• on checksum mismatch, the server-FP shall notify the user-FP of the mismatch via standard reply means;

– if the remote file to be created already exists at the server, the server-FP shall issue a '550 File Action Not Taken, File Already Exists' reply to the user-FP;

– upon verifying that the target file exists and is the correct file, and that the file to be created does not already exist, the server-FP shall apply the update signals to a copy of the remote file and store the revised data at the server-FP in a new file.

3) For file structure a record shall be accessed based on its position in the file relative to the start of the file.

4) For CCSDS Packet structure a record shall be accessed based on the value of the CCSDS Packet Sequence Count in the record rather than on its relative position in the file.

NOTE – General provisions for operations involving the CCSDS Packet structure are given in 3.2.1.4.

g) SIZE (SIZE):

– The SIZE command shall cause the server-FP to send the size of the file specified in the pathname to the user-FP.

– The server-FP shall communicate the size of the file via the reply message and code.

h) SUPPRESS REPLY TEXT (SUPP):

The SUPP command shall cause the server-FP to set the MIB parameter fpSuppressReplyText to ‘TRUE’ (enabled).

i) RESTART (REST):

– The REST command shall cause the server-FP to begin execution the file transfer command immediately following the REST command at the restart marker rather than at the beginning of the file.

– The argument field for the REST command shall contain the restart marker at which the file transfer is to be restarted.

– The REST command shall be immediately followed by the appropriate SCPS-FP file transfer command, which shall cause the interrupted file transfer to resume.

NOTE – The command was changed from Internet Standard 9 (reference [1]) to remove the condition that the REST command skip over the file.

3.3.4 SCPS-FP REPLIES

3.3.4.1 Reply Processing

3.3.4.1.1 Reply processing for SCPS-FP differs from Internet FTP as follows:

– The default reply mode shall be ‘suppress reply text’.

– An FP reply with the reply text suppressed shall consist of a three digit number (transmitted as three alphanumeric characters) followed by a Space <SP> and terminated by the Telnet end-of-line code (e.g., ‘150<SP><CRLF>’).

– Reply text for replies to commands such as PASV, SIZE, and INTR must be present to permit processing to continue.

3.3.4.1.2 The user **should** be provided with the capability to enable or disable the reply suppression.

3.3.4.2 Reply Codes

3.3.4.2.1 Specific reply text for use in SCPS-FP is suggested for the codes listed below:

110	MARK ssss = rrrr	(See Note 1)
211	Reply text suppression enabled OR Reply text suppression disabled	(See Note 2)

211 BETS option enabled with fill code xxxx OR
BETS option disabled (See Note 3)

213 <pathname> SIZE <file size>

3.3.4.2.2 SCPS-FP also supports the following codes:

256 Transfer Interrupted at xxxx

258 Requested action forced to complete with errors

259 BETS transfer completed with gaps filled at location(s) xxxx

554 Requested action not taken: invalid REST parameter (See Note 4)

555 Requested action not taken: type or stru mismatch (See Note 5)

556 Requested action not taken
Record ID not found

557 Requested action not taken
Remote file does not match local file

NOTES

1 In this format, ssss is a text string that appeared in a Restart Marker in the data stream and encodes a position in the sender's file system; rrrr encodes the corresponding position in the receiver's file system.

2 Text is for replies to the SUPP command and NSUP command, respectively.

3 Text is for replies to the BETS command and NBES command, respectively.

4 A 554 reply may result from a SCPS-FP service command that follows a REST command. The reply indicates that the existing file at the server-FP cannot be repositioned as specified in the REST.

5 A 555 reply may result from an APPE command or from any SCPS-FP service command following a REST command. The reply indicates that there is some mismatch between the current transfer parameters (type and stru) and the attributes of the existing file.

3.4 DECLARATIVE SPECIFICATIONS

3.4.1 MINIMUM IMPLEMENTATION

NOTE – The minimum implementation is defined below under **Required**. Other implementations are listed under the **Optional** headings.

3.4.1.1 Required Commands

The following commands and options must be supported by every SCPS server-FP and user-FP:

Type:	IMAGE (binary)
Mode:	Stream
Structure:	File
Commands:	ABOR, RETR, STOR, IDLE, QUIT, INTR, REST, SIZE, READ, UPDT, SITE, DELE, SUPP, NSUP, ARST, NARS

3.4.1.2 Optional Commands Based on Underlying File System/Operating System Capabilities

The following commands and options **should** be supported by every SCPS server-FP and user-FP on systems whose underlying file system and/or operating system supports a directory structure:

Commands:	LIST, MKD, RMD, CWD, PWD

3.4.1.3 Optional Commands Based on Mission Needs

The following commands and options **should** be supported as dictated by the needs of the particular mission:

Type:	ASCII
Structure:	CCSDS PACKET
Commands:	PORT, PASV, STRU, TYPE, MODE, USER, PASS, ACCT, RNFR, RNTO, STAT, SYST, HELP, BETS, NBES, APPE, STOU, COPY, NOOP, CDUP, NLST

3.4.2 COMMANDS

3.4.2.1 SCPS-FP Commands

3.4.2.1.1 The syntax for the following SCPS-FP commands is as described in section 5.3.1 of Internet Standard 9 (reference [1]):

> USER, PASS, CWD, QUIT, PORT, PASV, TYPE, STRU, MODE, RETR, STOR, RNFR, RNTO, REST, ABOR, DELE, RMD, MKD, LIST, STAT, SITE, ACCT, CDUP, NLST, NOOP, PWD, STOU, SYST, APPE, HELP.

3.4.2.1.2 The syntax for the remaining SCPS-FP commands is as follows:

> ARST <CRLF>
> BETS <BETS-fill-code> <CRLF>
> COPY <SP> <pathname> <SP> <pathname> <CRLF>
> IDLE <SP> <decimal-integer> <CRLF>
> INTR <CRLF>
> NARS <CRLF>
> NBES <CRLF>
> NSUP <CRLF>
> READ <SP> <pathname> <CRLF>
> SIZE <SP> <pathname> <CRLF>
> SUPP <CRLF>
> UPDT <SP> <pathname> <CRLF>

3.4.2.2 SCPS-FP Command Arguments

3.4.2.2.1 Unless modified by this Recommendation, the command argument syntax is as described in section 5.3.2 of Internet Standard 9 (reference [1]).

3.4.2.2.2 The definition of <pathname> is expanded from the RFC as indicated below:

> <pathname> ::= <string>

3.4.2.2.3 The <pathname> represents the location of the file data. For a file system-based storage system, <pathname> would be a filename with possibly a directory specification. For a memory-based storage system (i.e., no file system), <pathname> would take the form of one of the following:

<address>:<file-length>	/for source file (e.g., 1:500 = at location 1, length of 500)
<address>	/for destination file

3.4.2.2.4 The <address> and <file-length> are still specified in ASCII form. This alternate representation of <pathname> is only applicable to the STOR, RETR, UPDT, and READ commands.

3.4.2.2.5 The syntax for the remaining SCPS-FP commands is as outlined below:

a) BETS Arguments:

<BETS-fill-code>	=	<decimal-integer> in the range of 0 to 255 representing an ASCII character
	=	default is 255 (0xFF)

b) Common Record Read and Record Update Arguments:

<octet>	::=	1 eight-bit byte
<ascii-record>	::=	<string><LF>
<binary-record>	::=	<octet>
<CCSDS-Pkt-record>	::=	CCSDS Packet as defined in reference [3]
<data>	::=	1{<binary-record>}<record-count> | 1{<CCSDS-Pkt-record>}<record-count>
<record>	::=	<binary-record> | <CCSDS-Pkt-record>
<record-count>	::=	Corresponds to the number of octets for file structures and to the number of CCSDS Packets for CCSDS Packet structures
	::=	unsigned 16-bit integer transmitted in network byte order
<record-id>	::=	Numeric ID of the 'record' to access; corresponds an octet number for file structure and to the CCSDS Packet Sequence Count field of a CCSDS Packet for CCSDS Packet structures
	::=	unsigned 32-bit integer transmitted in network byte order

REQUIREMENT – When using the IMAGE type <record-id> zero (0) shall indicate the first octet in the file.

c) Record Update Argument Definition:

<update-control-data>	::=	<path-remote-target><LF> <path-remote-new> <LF> <checksum><LF> <update-data>
<path-remote-new>	::=	name of file (with or without full pathname) in which to store the updated source file on remote system
	::=	<pathname>

<path-remote-new-contents> ::= 1{<binary-record>}many |
1{<CCSDS-Pkt-record>}many
<path-remote-target> ::= name of file (without full path identifier) to use as source file for updates on remote system
::= <pathname>
<checksum> ::= checksum computed for local file against which checksum of remote file must match
::= <decimal-32-bit-integer>
<path-remote-target-contents> ::= 1{<binary-record>}many |
1{<CCSDS-Pkt-record>}many
<update-data> ::= 1{<update-signal>}many

<update-signal> ::= <delete-signal><record-id><record-count> |
<change-signal-id><record-id><record-count><data>

REQUIREMENTS

1 <update-signals> shall be sorted in ascending order by <record-id>.
2 The record range associated with an update signal shall not overlap with the record range of any other update signal.

<delete-signal> ::= signal to delete record(s) whose start is specified by <record-id>
::= d

<change-signal-id> ::= <addafter-signal> |
<insertbefore-signal> |
<writeover-signal>
<addafter-signal> ::= signal to add (insert) records after the record specified by <record-id>
::= a
<insertbefore-signal> ::= signal to add (insert) records before the record specified by <record-id>
::= i
<writeover-signal> ::= signal to write over (replace) records starting at the record specified by <record-id>
::= w

d) Record Read Argument Definition:

<read-control-data> ::= <path-remote><LF>
<forced-read-option><LF>
<record-count><LF>
<sorted-record-ids><LF>
<EOF>

<path-remote>	::=	name of file (without full path identifier) to read on remote system
	::=	<pathname>
<path-remote-contents>	::=	1{<record>}many
<forced-read-option>	::=	ignored recoverable errors during the record read, potentially accepting corrupt or incomplete data
	::=	Y | N
<sorted-record-ids>	::=	<rcd-range> 1{<rcd-range>}many
<rcd-range>	::=	<record-id-start>-<record-id-end>
<record-id-start>	::=	<record-id>
<record-id-end>	::=	<record-id>
<EOF>	::=	as defined in 3.2.1.3

REQUIREMENTS

1 <record-id-start> shall be less than or equal to <record-id-end>.
2 When a single record is required, <record-id-start> shall equal <record-id-end>.
3 Spaces shall not appear anywhere in <rcd-range>.
4 <record-id> of zero (0) shall be allowed.
5 Record-ranges shall appear in a numerically ascending order.

3.4.3 SEQUENCING OF COMMANDS AND REPLIES

3.4.3.1 The sequencing of commands and replies for the following SCPS-FP commands is as described in section 5.4 of Internet Standard 9 (reference [1]):

USER, PASS, CWD, QUIT, PORT, PASV, TYPE, STRU, MODE, RETR, RNFR, RNTO, REST, ABOR, DELE, RMD, MKD, LIST, STAT, SITE, ACCT, CDUP, NLST, NOOP, PWD, STOU, SYST, APPE, HELP.

3.4.3.2 SCPS-FP adds the reply code of 551 to the RMD command. If the specified directory does not exist the reply code should be 550. If the directory is not empty, the reply code should be 551.

3.4.3.3 The sequencing of replies for the STOR command is as amended below:

STOR
125, 150
(110)
226, 259, 250
425, 426, 451, 551, 552
532, 450, 452, 553
500, 501, 421, 530

3.4.3.4 The sequencing of commands and replies for the remaining SCPS-FP commands is as follows:

a) Transfer Parameter commands:

ARST
200
500, 502, 421
NARS
200
500, 502, 421
BETS
211
500, 421
NBES
211
500, 421

b) File action commands:

COPY
250
500, 502, 421
INTR
256
500, 501, 421, 530
NSUP
211
500, 421
READ
125, 150
(110)
226, 250, 258
425, 426, 451, 557
450, 550, 556
500, 501, 421, 530
SIZE
213
500, 501, 504, 530, 550
SUPP
211
500, 421
UPDT
250
451, 551, 552
532, 450, 452, 550, 557
500, 501, 421, 530
IDLE
200
500, 502, 421

3.5 STATE DIAGRAMS

3.5.1 The state diagrams for ABOR, ACCT, APPE, CDUP, CWD, DELE, HELP, LIST, MKD, MODE, NLST, NOOP, PASS, PASV, PORT, PWD, QUIT, REST, RETR, RMD, RNFR, RNTO, SITE, STAT, STOR, STOU, STRU, SYST, TYPE, and USER are given in section 6 of Internet Standard 9 (reference [1]). Figure 3-1 is the state diagram for the largest group of SCPS-FP commands:

ARST, BETS, COPY, IDLE, INTR, NARS, NBES, NSUP, SIZE, SUPP, UPDT.

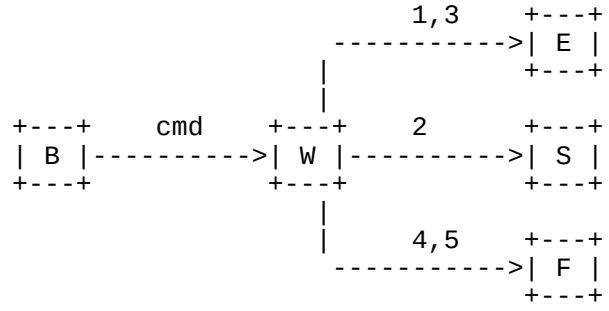


Figure 3-1: State Diagram 1

3.5.2 Figure 3-2 is the state diagram for the SCPS-FP READ command:

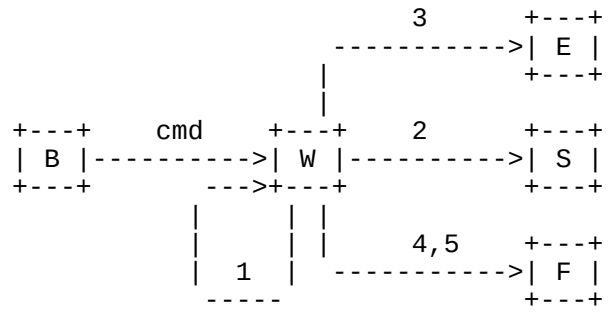


Figure 3-2: State Diagram 2

3.6 CONNECTION ESTABLISHMENT

Internet FTP uses TCP to establish control and data connections. In the same way, the SCPS-FP control connection is established via TCP or SCPS-TP (reference [4]) between the user process port U and the server process port L. This protocol is assigned the service port 21 (25 octal); that is, L=21.

3.7 INTERFACE TO LOWER PROTOCOL STACK LAYERS

Table 3-3 summarizes the types of transport services that the SCPS-FP will use. Detailed functional descriptions of these services are provided in Internet Standard 7 (reference [B4]) under the heading 'User/TCP Interface'.

Table 3-3: Transport Services Summary

Command	Service	Comment
Open	I: local port I: foreign socket I: active/passive I: [time-out] I: [precedence] I: [security/compartment] I: [options] O: local connection name	Open either an active or passive (LISTEN) connection.
Send	I: local connection name I: buffer address I: byte count I: PUSH flag I: URGENT flag I: [time-out]	Send data on the indicated connection.
Receive	I: local connection name I: buffer address I: byte count O: byte count O: urgent flag O: push flag	Receive data from the indicated connection.
Close	I: local connection name	Gracefully close the indicated connection.
Status	I: local connection name O: status data	Get the status of the indicated connection.
Abort	I: local connection name	Abort all pending Sends and Receives.
Legend:	I = input to the service O = output from the service [] = optional parameter	

3.8 MISCELLANEOUS SCPS-FP REQUIREMENTS

3.8.1 Concurrency of the data and control connections as described in RFC 1123, section 4.1.3.3 (see reference [2]), is applicable to SCPS-FP.

3.8.2 Pathname specification as described in RFC 1123, section 4.1.4.1 (see reference [2]), is applicable to SCPS-FP.

3.8.3 The requirement for maintaining synchronization with the server as described in RFC 1123, section 4.1.4.4 (see reference [2]), is applicable to SCPS-FP.

4 MANAGEMENT INFORMATION BASE (MIB) REQUIREMENTS

4.1 OVERVIEW

The MIB for systems implementing the SCPS-FP shall include, at a minimum, the following entries. The SCPS-FP Implementation should reset MIB parameters to their default values at client/server startup. The definitions for each MIB entry are documented below.

4.2 DATA TRANSFER TYPE

The Data Transfer Type parameter contains the default data transfer type to use for a file transfer when the user does not specify a data transfer type.

Name: fpTransferType
Syntax: Character {ASCII (A),
IMAGE (I) (default)
}
Access by FP: Read/Write

4.3 TRANSMISSION MODE

The Transmission Mode parameter contains the default data transmission mode to use for a file transfer when the user does not specify a transmission mode.

Name: fpTransmissionMode
Syntax: Character {STREAM (S)
}
Access by FP: Read/Write

4.4 DATA STRUCTURE

The Data Structure parameter contains the default data structure to use for a file transfer when the user does not specify the data structure.

Name: fpDataStructure
Syntax: Character {File (F),
Record (R)
}
Access by FP: Read/Write

4.5 AUTORESTART

The Autorestart Enabled parameter contains the default setting to use for the autorestart capability when the user does not specify whether or not autorestart is needed.

Name: fpAutorestartEnabled
Syntax: Boolean {enabled (1),
disabled (0)
}
Access by FP: Read/Write

4.6 MAXIMUM NUMBER OF AUTOMATIC RESTARTS

The Maximum Number of Automatic Restarts parameter contains the default maximum number of automatic restarts to use for a failed file transfer when autorestart is enabled and the user does not specify the maximum number of restarts.

Name: fpMaxAutorestarts
Syntax: Integer
Access by FP: Read/Write

4.7 USE PORT COMMAND ON EACH SEND

The Use PORT parameter contains the default setting for whether or not the PORT command is used on each file transfer when the user does not specify if the PORT command is needed.

Name: fpUsePORT
Syntax: Boolean {use PORT (1),
don’t Use PORT (0)
}
Access by FP: Read/Write

4.8 SUPPRESS SERVER REPLY TEXT

The Suppress Server Reply Text parameter contains the default setting to use for suppressing the reply text of many of the FP messages when the user does not specify whether or not the text should be suppressed.

Name: fpSuppressReplyText
Syntax: Boolean {suppress (1) (default),
don’t suppress (0)
}
Access by FP: Read/Write

4.9 SERVER IDLE TIMEOUT

The Server Idle Timeout parameter contains the default setting for the number of seconds the FP server should be idle before timing out.

Name: fpIdleTimeout
Syntax: Integer (seconds)
Access by FP: Read/Write

NOTE – Recommended default per Internet Standard 3 (see reference [2]) is 300 seconds (5 minutes).

4.10 BEST EFFORT TRANSPORT SERVICE

The Best Effort Transport Service parameter contains the default setting for the Best Effort Transport Service option.

Name: fpBETSEnabled
Syntax: Boolean {enabled (1),
disabled (0) (default)
}
Access by FP: Read/Write

4.11 BEST EFFORT TRANSPORT SERVICE FILL CHARACTER

The Best Effort Transport Service Fill Character parameter contains the default decimal number representing the ASCII character to use for the Best Effort Transport Service fill code.

Name: fpBETSFillChar
Syntax: Integer {0 to 255, inclusive
}
Access by FP: Read/Write

5 CONFORMANCE REQUIREMENTS

5.1 CONFORMANCE REQUIREMENTS OVERVIEW

5.1.1 The SCPS-FP is derived from Internet FTP and uses the FTP model as described in Internet Standard 9 (reference [1]), section 2.3, as its basis. The base standards, Internet Standards 9 and 3 (references [1] and [2]), provide for much of the functionality required for the SCPS-FP. However, the following service extensions are required:

- automatic restart of file transfer;
- record update;
- record read;
- manual interrupt;
- suppression of reply text.

5.1.2 Defined below are the requirements for four types of implementations of the SCPS-FP:

a) a conforming minimum SCPS-FP implementation that is Internet FTP interoperable;

b) a conforming full SCPS-FP implementation;

c) an implementation that is 100% Internet compatible;

d) an implementation that includes optional non-SCPS features.

5.1.3 The subsections below document the protocol requirements. Refer to annex D of this document for the service specification, which documents the recommended interfaces for the user commands as well as the required interactions between the user command processing and the file protocol itself.

5.2 SCPS-FP CONFORMING MINIMUM IMPLEMENTATION

A conforming minimum implementation of SCPS-FP shall support the following capabilities in accordance with the indicated base standards:†

a) IMAGE data type as specified in RFC 959, section 3.1.1.3;

b) file structure as specified in RFC 959, sections 3.1.2 and 3.1.2.1;

c) data connection establishment as specified in RFC 959, section 3.2;

d) data connection management as specified in RFC 959, section 3.3, as amended by RFC 1123, section 4.1.2.5;

† The remainder of this section references specific sections of Internet RFCs 959 and 1123, which are contained in Internet Standards 9 (reference [1]) and 3 (reference [2]), respectively.

e) stream transmission mode as specified in 3.2.1.3 of this document;

f) file integrity as specified in 3.2.3.1 of this document;

g) access control command QUIT as specified in RFC 959, section 4.1.1;

h) service commands DELE, RETR, STOR, SITE, REST, and ABOR as specified in RFC 959, section 4.1.3, as amended by RFC 1123, section 4.1.2.8 (SITE), and 3.3.3 of this document (REST); and service commands SUPP, NSUP, INTR, READ, SIZE, and UPDT as specified in 3.3.3 of this document;

i) Telnet End-of-Line Code as specified in RFC 959, section 4.1.3 (page 34), as amended by RFC 1123, section 4.1.2.10;

j) protocol replies for the SCPS-FP minimum implementation as specified in RFC 959, sections 4.2, 4.2.1, and 4.2.2, as amended by RFC 1123, section 4.1.2.11;

k) connections as specified in RFC 959, section 5.2, as amended by RFC 1123, section 4.1.2.12;

l) command interpretation as specified in RFC 959, section 5.3;

m) protocol commands for the SCPS-FP minimum implementation as specified in RFC 959, section 5.3.1;

n) protocol command arguments for the SCPS-FP minimum implementation as specified in RFC 959, section 5.3.2, as amended by 3.4.2.2 of this document;

o) sequencing of commands and replies for the SCPS-FP minimum implementation as specified in RFC 959, section 5.4;

p) state diagrams for the SCPS-FP minimum implementation as specified in 3.5 of this document;

q) control connection establishment as specified in 3.6 of this document;

r) interface to lower protocol stack layers as specified in 3.7 of this document;

s) protocol features as specified in 3.7 of this document;

t) idle time-out as specified by the IDLE command in 3.3.3 of this document;

u) concurrency of data and control as specified in RFC 1123, section 4.1.3.3;

v) pathname specification as specified in RFC 1123, section 4.1.4.1;

w) maintain synchronization as specified in RFC 1123, section 4.1.4.4;

x) record data integrity as specified in 3.2.3.2 of this document;

y) transfer parameters commands ARST and NARS as specified in 3.3.2 of this document.

5.3 SCPS CONFORMING FULL IMPLEMENTATION

A full implementation of SCPS-FP should support the capabilities below (in addition to the minimum requirements) in accordance to the indicated base standards:

NOTE – Intermediate implementations are permitted. In general the inclusion of any of the following capabilities in an implementation will be mission-dependent. The base implementation for any intermediate implementation must be the minimum implementation.

a) ASCII data type as specified in RFC 959, section 3.1.1.1, as amended by 3.2.1.1.2 of this document;

b) non-print format controls as specified in RFC 959, section 3.1.1.5.1;

c) error recovery and restart as specified in 3.2.2 of this document and its subsections;

d) access control commands USER, PASS, and CWD, as specified in RFC 959, section 4.1.1;

e) transfer parameters commands TYPE, STRU, MODE, PORT and PASV as specified in RFC 959, section 4.1.2 and Appendix I (for PASV), as amended by RFC 1123, section 4.1.2.6 (for PASV); also BETS and NBES as specified in 3.3.2 of this document;

f) service commands STAT, RNFR, RNTO, RMD, MKD, and LIST as specified in RFC 959, section 4.1.3, as amended by RFC 1123, section 4.1.2.7 (LIST);

g) protocol replies for the additional SCPS capabilities as specified in RFC 959, sections 4.2, 4.2.1, and 4.2.2, as amended by RFC 1123, section 4.1.2.11, and as amended by 3.3.4 and 3.3.4.2 of this document;

h) protocol commands for the additional SCPS capabilities as specified in RFC 959, section 5.3.1, as amended by 3.4.2.1 of this document;

i) protocol command arguments for the additional SCPS capabilities as specified in RFC 959, section 5.3.2, as amended by 3.4.2.2 of this document;

j) sequencing of commands and replies for the additional SCPS capabilities as specified in RFC 959, section 5.4, as amended by 3.4.3 of this document;

k) state diagrams for the additional SCPS capabilities as specified in 3.5 of this document;

l) display of replies to user as specified in RFC 1123, section 4.1.4.3;

m) MIB usage and support as specified in section 4 of this document.

5.4 FTP INTEROPERABLE IMPLEMENTATION

An implementation of the SCPS-FP having interoperability requirements with the FTP should support the following *additional* capabilities in accordance to the indicated base standards:

a) LOCAL 8 data type as specified in RFC 959, section 3.1.1.4, as amended by RFC 1123, section 4.1.2.1;

b) record structure as specified in RFC 959, sections 3.1.2 and 3.1.2.2, as amended by RFC 1123, section 4.1.2.4;

c) restart mechanisms for block and compressed modes as specified in RFC 959, section 3.5, as amended by RFC 1123, section 4.1.3.4;

d) compressed transmission mode as specified in RFC 959, sections 3.4.1 and 3.4.2;

e) access control command CDUP as specified in RFC 959, section 4.1.1;

f) service commands APPE, PWD, NLST, SYST, and NOOP as specified in RFC 959, section 4.1.3, as amended by RFC 1123, section 4.1.2.7 (for NLST);

g) minimum implementation as specified in RFC 1123, section 4.1.2.13;

h) non-standard command verbs as specified in RFC 1123, section 4.1.3.1;

i) ‘QUOTE’ command as specified in RFC 1123, section 4.1.4.2;

j) protocol commands for the FTP minimum implementation as specified in RFC 959, section 5.3.1;

k) protocol command arguments for the FTP minimum implementation as specified in RFC 959, section 5.3.2;

l) sequencing of commands and replies for the FTP minimum implementation as specified in RFC 959, section 5.4;

m) state diagrams for the FTP minimum implementation as specified in 3.5 of this document.

5.5 OTHER OPTIONAL IMPLEMENTATIONS

Other implementations of SCPS-FP may support the following *additional* non-SCPS capabilities in accordance to the indicated base standards:

a) service commands ALLO and STOU as specified in RFC 959, section 4.1.3, as amended by RFC 1123, section 4.1.2.9 (for STOU);

b) EBCDIC data type as specified in RFC 959, section 3.1.1.2;

c) LOCAL m data type as specified in RFC 959, section 3.1.1.4, as amended by RFC 1123, section 4.1.2.1;

d) Telnet format control and carriage control (ASA) as specified in RFC 959, sections 3.1.1.5.2 (as amended by RFC 1123, section 4.1.2.2) and 3.1.1.5.3, respectively;

e) block transmission mode as specified in RFC 959, section 3.4.2;

f) access control commands SMNT and REIN as specified in RFC 959, section 4.1.1;

g) protocol commands for the FTP optional commands as specified in RFC 959, section 5.3.1;

h) protocol command arguments for the FTP optional commands as specified in RFC 959, section 5.3.2;

i) sequencing of commands and replies for the FTP optional commands as specified in RFC 959, section 5.4;

j) state diagrams for the FTP optional commands as specified in 3.5 of this document.

ANNEX A

ACRONYMS AND ABBREVIATIONS

(This annex is **not** part of the Recommendation.)

Term	Meaning
EOF	End Of File
FTP	File Transfer Protocol
NP	Network Protocol
PDU	Protocol Data Unit
PICS	Protocol Implementation Conformance Statement
RFC	Request for Comments
SCPS	Space Communications Protocol Specification
SCPS-NP	SCPS Network Protocol
SCPS-SP	SCPS Security Protocol
SCPS-TP	SCPS Transport Protocol
TCP	Transmission Control Protocol

ANNEX B

INFORMATIVE REFERENCES

(This annex **is not** part of the Recommendation.)

[B1] *Procedures Manual for the Consultative Committee for Space Data Systems*. CCSDS A00.0-Y-7. Yellow Book. Issue 7. Washington, D.C.: CCSDS, November 1996.

[B2] Harold S. Stone, ed. *Introduction to Computer Architecture*. The SRA Computer Science Series. 2nd ed. Chicago: Science Research Associates, 1980.

[B3] *Space Communications Protocol Specification (SCPS)—Rationale, Requirements, and Application Notes*. Report Concerning Space Data System Standards, CCSDS 710.0-G-1. Green Book. Issue 1. Washington, D.C.: CCSDS, May 1999.

[B4] J. Postel. *Transmission Control Protocol*. STD 7, September 1981. [RFC 793]†

† Internet Request for Comments (RFC) texts are available on line in various locations (e.g., http://ietf.org/rfc/). In this list, Internet Standards are identified by 'STD' followed by the number of the standard, and RFCs are identified by 'RFC' followed by the number of the RFC. RFCs comprised by Internet Standards are given in square brackets following the citation.

ANNEX C

SCPS-FP PROTOCOL IMPLEMENTATION CONFORMANCE STATEMENT PROFORMA

(This annex **is** part of the Recommendation.)

C1 INTRODUCTION

This annex provides the Protocol Implementation Conformance Statement (PICS) Requirements List (PRL) for implementations of SCPS-FP. The PICS for an implementation is generated by completing the PRL in accordance with the instructions below. An implementation shall satisfy the mandatory conformance requirements of the base standards referenced in the PRL.

An implementation's completed PRL is called the PICS. The PICS states which capabilities and options of the protocol have been implemented. The following can use the PICS:

- the protocol implementer, as a checklist to reduce the risk of failure to conform to the standard through oversight;
- the supplier and acquirer or potential acquirer of the implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma;
- the user or potential user of the implementation, as a basis for initially checking the possibility of interworking with another implementation (note that, while interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICSes);
- a protocol tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation.

C1.1 NOTATION

The following are used in the PRL to indicate the status of features:

Status Symbols

M	mandatory.
M.<n>	support of every item of the group labeled by the same numeral <n> required, but only one is active at a time.
O	optional.

O.<n> optional, but support of at least one of the group of options labeled by the same numeral <n> is required.

C conditional.

- non-applicable field/function (i.e., logically impossible in the scope of the PRL).

I out of scope of PRL (left as an implementation choice).

X excluded or prohibited.

Two character combinations may be used for dynamic conformance requirements. In this case, the first character refers to the static (implementation) status, and the second refers to the dynamic (use) status; thus 'MO' means 'mandatory to be implemented, optional to be used'.

Notations for Conditional Status

The following predicate notations are used:

<predicate>:: This notation introduces a group of items, all of which are conditional on <predicate>.

<predicate>: This notation introduces a single item which is conditional on <predicate>.

In each case, the predicate may identify a protocol feature, or a Boolean combination of predicates. ('^' is the symbol for logical negation, '|' is the symbol for logical OR, and '&' is the symbol for logical AND.)

<index>: This notation indicates that the status following it applies only when the PICS states that the features identified by the index are supported. In the simplest case, <index> is the identifying tag of a single PRL item. The symbol <index> also may be a Boolean expression composed of several indices.

<index>:: This notation indicates that the associated clause should be completed.

Notations Used in the Protocol Feature Column

<r> Symbol used to denote the receiving system.

<t> Symbol used to denote the transmitting system.

Support Column Symbols

The support of every item as claimed by the implementer is stated by entering the appropriate answer (Y, N, or N/A) in the support column:

Y Yes, supported by the implementation.

N No, not supported by the implementation.

N/A Not applicable.

C1.2 REFERENCED BASE STANDARDS

The base standards referenced in the PRL are:

- SCPS-FP (this document);
- RFC 959 (reference [1]);
- RFC 1123 (reference [2]).

In the tables below, the notation in the Reference column combines one of the short-form document identifiers above (e.g., SCPS-FP) with applicable subsection numbers in the referenced document. RFC numbers are used to facilitate reference to subsections within the Internet Standards.

C1.3 GENERAL INFORMATION

C1.3.1 IDENTIFICATION OF PICS

Ref	Question	
1	Date of Statement (DD/MM/YYYY)	
2	PICS serial number	
3	System Conformance statement cross-reference	

C1.3.2 IDENTIFICATION OF IMPLEMENTATION UNDER TEST (IUT)

Ref	Question	Response
1	Implementation name	
2	Implementation version	
3	Machine name	
4	Machine version	
5	Operating System name	
6	Operating System version	
7	Special Configuration	
8	Other Information	

C1.3.3 IDENTIFICATION

Supplier	
Contact Point for Queries	
Implementation name(s) and Versions	
Other Information Necessary for full identification - e.g., name(s) and version(s) for machines and/or operating systems; System Name(s)	

C1.3.4 PROTOCOL SUMMARY

Protocol Version	
Addenda Implemented	
Amendments Implemented	
Have any exceptions been required? (Note: A YES answer means that the implementation does not conform to the protocol. Non-supported mandatory capabilities are to be identified in the PICS, with an explanation of why the implementation is non-conforming.	Yes_____ No_____
Date of Statement	

C1.4 INSTRUCTIONS FOR COMPLETING THE PRL

An implementer shows the extent of compliance to the protocol by completing the PRL; that is, compliance to all mandatory requirements and the options that are not supported are shown. The resulting completed PRL is called a PICS. In the Support column, each response shall be selected either from the indicated set of responses, or it shall comprise one or more parameter values as requested. If a conditional requirement is inapplicable, N/A shall be used. If a mandatory requirement is not satisfied, exception information must be supplied by entering a reference Xi, where i is a unique identifier, to an accompanying rationale for the noncompliance. When the requirement is expressed as a two-character combination (as defined above), the response shall address each element of the requirement; e.g., for the requirement 'MO', the possible compliant responses are 'YY' or 'YN'.

C1.5 DATA TRANSFER FUNCTIONS

Item	Protocol Feature	Reference	Status	Support
dt1	Files are transferred via the data connection only	RFC 959: 3	M	
dt2	Commands and replies are transferred via control connection only		M	

C2 DATA REPRESENTATION AND STORAGE

C2.1 DATA TYPES

Item	Protocol Feature	Reference	Status	Support
dtas	ASCII Type	RFC 959: 3.1.1.1	O	
dtas1	<t> converts internal data form to standard eight-bit ASCII for transmission		dtas:M	
dtas2	<r> converts data from standard eight-bit ASCII to internal form		dtas:M	
dtas3	<CRLF> sequence used when necessary to denote end of a line of text		dtas:M	
dtim	IMAGE Type	RFC 959: 3.1.1.3	M	

C2.2 DATA STRUCTURES

C2.2.1 File structure

Item	Protocol Feature	Reference	Status	Support
dsfs	File Structure	RFC 959: 3.1.2.1	M	
dsfs1	File structure is default data structure		M	
dsfs2	Files of structure File are treated as a continuous sequence octets when transferred		M	

C2.2.2 CCSDS Packet Structure

Item	Protocol Feature	Reference	Status	Support
cps	CCSDS Packet Structure	SCPS-FP: 3.2.3.3	O	
cps1	STRU R turns on CCSDS Packet processing		cps:M	
cps2	<r> ends partial file on Packet boundary if transfer of a file of CCSDS Packets aborts for any reason (abnormal connection loss, receipt of the INTR command or the ABOR command)		cps:M	
cps3	In record operations, record identifiers are interpreted as referring to the Packet Sequence Count field in the record		cps:M	
cps4	Packet Data Length field delimits records in file		cps:M	
cps5	Protocol invokes the record boundary option of SCPS-TP (Reference [4]) to delimit records during transfer		cps:M	
cps6	Client issues TYPE command if necessary to switch to IMAGE data type before transferring files of CCSDS Packets		cps:M	

C2.3 DATA CONNECTION ESTABLISHMENT

Item	Protocol Feature	Reference	Status	Support
dce1	Data Ports: user-FP			
dce1a	Default ports are supported	RFC 959: 3.2, 3.3	O	
dce1b	User-process default data port is 20 decimal	RFC 959: 3.2	dce1a:M	
dce2	Data Ports: server-FP			
dce2a	Default ports supported	RFC 959: 3.2, 3.3	O	
dce2b	Server-process default data port is adjacent to control connection (L-1)	RFC 959: 3.2	dce2a:M	
dce2c	Transfer byte size is 8 bits		M	
dce3	Data connection establishment			
dce3a	passive DTP 'listens' on the data port prior to sending transfer request command	RFC 959: 3.2	M	
dce3b	Server initiates data connection to the port		M	
dce3c	Data transfer occurs between DTPs in the direction that is determined by the FP request command		M	
dce3d	Server-PI sends a confirming reply		M	
dce4	Data Connection Termination			
dce4a	Server closes data connection under following conditions:			
dce4a1	Server completes sending data in a transfer mode that requires a close to indicate EOF	RFC 959: 3.2	M	
dce4a2	Server receives an ABORT command from user		M	
dce4a3	Port specification is changed by a command from user		M	
dce4a4	Control connection is closed		M	
dce4a5	An irrecoverable error condition occurs		M	
dce4a6	Server receives an INTR command from user	SCPS-FP: 3.2.2.4	M	
dce4b	Server indicates closing by 226, 250, 256, 258, 259 reply only	RFC 959: 3.2 SCPS-FP: 3.2.2.4	M	

C2.4 DATA CONNECTION MANAGEMENT

Item	Protocol Feature	Reference	Status	Support
dcm1	Only user-PI initiates use of non-default port	RFC 959: 3.2	M	
dcm2a	User-PI can specify non-default user-side data port with PORT command	RFC 959: 3.2	M	
dcm2b	User-PI can request server to identify a non-default server-side data port with PASV command		O	
dcm3	User-PI sends PORT command prior to each transfer when in STREAM mode	RFC 1123: 4.1.2.5	M	

C2.5 TRANSMISSION MODE

Item	Protocol Feature	Reference	Status	Support
trmo	Complete data transfers terminated with an End of File (EOF) indication, either explicitly or implicitly	RFC 959: 3.4	M	

C2.6 ERROR RECOVERY AND RESTART

C2.6.1 Restart Mechanism in Stream Mode

Item	Protocol Feature	Reference	Status	Support
err1a	For ASCII data type, restart point is relative to the start of the transfer image	SCPS-FP: 3.2.2.1	M	
err1b	For Image data types, restart point is relative to the start of the file		M	
err2	For CCSDS Packet structure, restart point identifies a particular packet by its Packet Sequence count rather than the Packet's position relative to the start of the file		M	
err3a	Restart markers are imbedded in the data stream	SCPS-FP: 3.2.2.1	X	
err3b	Receiving DTP determines restart marker by querying file system for current size of the file		M	
err3c1	User-FP sends SIZE command to the server	SCPS-FP: 3.2.2.1	M	
err3c2	User-FP extracts restart marker from reply to SIZE		M	
err3c3	User-FP repositions file pointer to the restart marker and sends REST to the server-FP		M	
err3d1	User-FP determines size of the file received so far	SCPS-FP: 3.2.2.1	M	
err3d2	User-FP repositions file pointer to restart marker and sends REST to the server-FP		M	
err3e1	User-FP sends SIZE command to the <r> server	SCPS-FP: 3.2.2.1	M	
err3e2	User-FP extracts restart marker from the reply to SIZE		M	
err3e3	User-FP sends REST to <r> and <t> server-FPs		M	
err3f	User-FP waits for REST replies before sending file transfer command that was being executed when the system failure occurred		M	
err3g	Server-FPs advance their respective files to the restart point subject to data type before the data transfer resumes		M	

C2.6.2 Automatic Restart

Item	Protocol Feature	Reference	Status	Support
aurs1	User-PI, not user-FP, responsible for autorestart decision making	SCPS-FP: 3.2.2.2	M	
aurs2	Partial files are saved if data connection times out or aborts when autorestart is enabled		M	
aurs3	Autorestart engages when data transfer ends abnormally		M	
aurs4	all changes rolled back if the data connection times out or aborts when autorestart is disabled		M	
aurs5a	User-DTP restarts failed file transfer up to the maximum number of restarts	SCPS-FP: 3.2.2.2	M	
aurs5b	User-DTP requires user to specify the restart point or restart command		X	
aurs5c	User-DTP notifies user of failed transfer		O	
aurs5d	User-DTP requests confirmation from the user to restart		O	
aurs5e	User-DTP sends SIZE command to the server-FP when <r> DTP is a server		M	
aurs5f	User-DTP uses the data in SIZE reply to fill in the argument to the REST command when the receiving DTP is a server		M	
aurs5g	User-DTP sends REST command to server		M	
aurs5h	User-DTP restarts file transfer after successfully processing the REST command		M	
aurs5i	User-DTP resends all user-set transfer parameters prior to file transfer to the server, if the control connection had to be reestablished to proceed with the autorestart		M	
aurs6	Default maximum number of restarts equal to 3		M	
aurs7	<r> and <t> DTPs skip over the successfully transferred data before continuing with restarted file transfer		M	

C2.6.3 Manual Interrupt and Manual Restart

Item	Protocol Feature	Reference	Status	Support
mimr1	User-FP sends 'INTR<CRLF>' to the server-FP	SCPS-FP: 3.2.2.4	M	
mimr2	On a user-to-server transfer the user-FP saves the data in INTR reply as the interruption point		M	
mimr3	On a server-to-user transfer the user-FP ignores the interruption point provided in the INTR reply and uses its more recent interruption point for restart		M	
mimr4	Server-to-Server Transfer			
mimr4a	User-FP sends INTR command to <r> server only after receiving a 256 reply from <t> server	SCPS-FP: 3.2.2.4	M	
mimr4b	User-FP uses interrupt point provided in <r> server-FP reply and ignores that provided in <t> server-FP reply		M	

C2.7 INTEGRITY

C2.7.1 File Integrity

Item	Protocol Feature	Reference	Status	Support
fi	Incomplete changes for a failed non-autorestarted RETR or STOR are rolled back	SCPS-FP: 3.2.3.1	M	

C2.7.2 Record Data Integrity

Item	Protocol Feature	Reference	Status	Support
rdi1	Incomplete changes for a failed READ or UPDT rolled back	SCPS-FP: 3.2.3.2	M	
rdi2	SCPS-FP checksum is used for UPDT		M	
rdi3	SCPS-FP-specified CRC polynomial used to compute checksum		M	
rdi4	SCPS-FP-specified CRC algorithm used to compute the checksum		M	

C2.8 FILE TRANSFER FUNCTIONS

C2.8.1 SCPS-FP Commands

Item	Protocol Feature	Reference	Status	Support
fciu	User-PI sends FP commands and interprets replies	RFC 959: 2.3	M	
fcis	Server-PI interprets FP commands and sends replies		M	

C2.8.2 Access Control Commands

Item	Protocol Feature	Reference	Status	Support
usco	USER Command	RFC 959: 4.1.1	O	
usco1	Server allows new USER command to be entered at any point to change access control		usco:M	
usco2	If a new USER command is issued, all transfer parameters remain unchanged		usco:M	
usco3	If a new USER command is issued, any file transfer in progress is completed under the old access control parameters		usco:M	
paco	PASS Command		O	
paco1	immediately preceded by user command		paco:M	
paco2	User-DTP hides sensitive password information		paco:M	
cdco	CWD Command		O	
cdco1	Server permits working with a different directory		cdco:M	
cdco2	Server preserves the user's login/accounting information		cdco:M	
cdco3	Server preserves transfer parameter settings		cdco:M	
quco	QUIT Command		M	
quco1	Server takes the effective action of an abort (ABOR) and a logout (QUIT) if an unexpected close occurs on the control connection		M	

C2.8.3 Transfer Parameter Commands

Item	Protocol Feature	Reference	Status	Support
tp1	Default value is the last specified value or standard default value	RFC 959: 4.1.2	M	
tp2	Server remembers the applicable default values		M	
potp	PORT command		M	
potp1	User-FP sends PORT command for stream mode for each transfer	RFC 1123: 4.1.2.5	M	
potp2	32-bit Internet host address and a 16-bit SCPS-TP port address are concatenated to form the PORT argument	RFC 959: 4.1.2	M	
potp3	Address information is broken into eight-bit fields		M	
potp4	Value of each field is transmitted as a decimal number (in character string representation)		M	
potp5	Fields are separated by commas		M	
patp	PASV Command		O	
patp1	Server-FP has implemented PASV	RFC 1123: 4.1.2.6	O	
patp2	User-FP sends PASV per transfer		patp:M	
tytp	TYPE Command	RFC 959: 4.1.2	M	
sttp	STRU Command		M	
sttp1	Server resets STRU setting		M	
motp	MODE Command		M	
motp1	Server resets the MODE setting		M	
artp	ARST Command	SCPS-FP: 3.3.2	M	
artp1	Server turn on its autorestart flag		M	
natp	NARS Command		M	
natp1	Server turn off its autorestart flag		M	
betp	BETS Command		O	
betp1	Server turns on its BETS flag		betp:M	
betp2	Server saves BETS fill code to use during the file transfer		O	
nbtp	NBES Command		betp:M	
nbtp1	Server turn off its BETS flag		nbtp:M	

C2.8.4 SCPS-FP Service Commands

Item	Protocol Feature	Reference	Status	Support
sc	All data is transferred over the data connection except for certain informative replies	RFC 959: 4.1.3	M	
resc	RETR Command		M	
resc1	Server transfers a copy of the file, specified in the pathname, to the server- or user-DTP at the other end of the data connection		M	
resc2	Status and contents at the server site are unaffected		M	
stsc	STOR Command	RFC 959: 4.1.3	M	
stsc1	Server accepts the data transferred via the data connection		M	
stsc2	If the specified file already exists at the server site, it is replaced by the newly transferred data		M	
stsc3	If the specified file does not exist at the server site, a new file is created for the transferred data		M	
stsc4	If BETS is enabled	SCPS-FP: 3.3.2	O	
stsc5	Server-DTP requests BETS option from the transport service for STOR and RETR operations only		stsc4:M	
stsc6	Server-DTP fills gaps in the data with the BETS fill code		stsc4:M	
stsc7	Server-DTP notifies user if any gaps in the data are found		stsc4:M	
rfcs	RNFR Command	RFC 959: 4.1.3	O	
rtcs	RNTO Command		O	
absc	ABOR Command		M	
absc1	If the previous command has been completed including data transfer, ABOR is processed		X	
absc2	If FP command already completed when ABOR received			
absc2a	Server closes data connection, if open		M	
absc2b	Server sends 226 reply indicating abort successful		M	
absc3	FP command not completed when ABOR received			
absc3a	Server aborts FP service in progress		M	
absc3b	Server closes data connection		M	
absc3c	Server returns 426 indicating service request was terminated abnormally		M	
absc3d	Server sends 226 reply indicating abort successful		M	
desc	DELE Command		M	
desc1	Server deletes file specified in the pathname at the server site		M	
desc2	User-FP provides extra level of protection (‘do you really want to delete’)		O	
rdsc	RMD Command		O	
rdsc1	Server removes directory specified in pathname from server site		rdsc:M	
mdsc	MKD Command		O	
mdsc1	Server creates the directory specified in pathname at server site		mdsc:M	
lssc	LIST Command		O	
lssc1	Implied Type AN is used	RFC 1123: 4.1.2.7	O	
lssc2a	Server sends list to the passive DTP	RFC 959: 4.1.3	lssc:M	
lssc2b	Server transfers list of files in the specified directory if the pathname specifies a directory or other group of files		lssc:M	
lssc2c	Server sends current information on file if pathname specifies a file		lssc:M	

lssc2d	Server accesses user's current working directory or default directory if null argument is used with LIST		lssc:M	
hesc	HELP Command		O	
hesc1	Server sends help information over the control connection		hesc:M	
hesc2	If an argument is provided (e.g., a command), more detailed information is provided as the response		O	
sisc	SITE Command		M	
stsc	STAT Command		O	
stsc1	Server sends the status response over the control connection in the form of a reply		stsc:M	
stsc2a	Telnet IP and Synch signals are included		stsc:M	
stsc2b	Server responds with the status of the operation in progress		stsc:M	
stsc3	STAT can be sent between file transfers		O	
stsc3a	If a partial pathname is given, server responds with list of file names or attributes associated with the pathname specification		O	
stsc3b	If no argument is given, server returns general status information about the server FP process		stsc:M	
stsc3c	General status information includes current values of all transfer parameters and the status of connections		stsc:M	
tosc	IDLE Time-out Command	SCPS-FP: 3.3.3	M	
tosc1	Server resets default idle time-out for the server for the current session		M	
tosc2	If the server is inactive (no command or data in process) for the default idle time-out period, server-FP terminates the process and closes the control connection	RFC 1123: 4.1.3.2	M	
tosc3	Default idle time-out is at least 5 minutes		M	
insc	INTR Command	SCPS-FP: 3.3.3	M	
insc1	Server interrupts file transfer		M	
insc2	Server communicates point of interruption in the reply message		M	
nssc	NSUP Command		M	
nssc1	Server disables suppression of reply text		M	
nssc2	Server includes reply text in the NSUP reply and all subsequent replies up until the suppression of reply text is re-enabled		M	
reasc	READ Command		M	
reasc1	Control data is transferred via the data connection		M	
reasc2	Only the records identified by the record IDs in the control data are transferred to the user-DTP		M	
reasc3	If forced read option is selected		O	
reasc3a	User-FP transmits an enabled forced read option flag to server-FP via control data		reasc3:M	
reasc3b	User-FP and server-FP continue processing the read when a recoverable error occurs		reasc3:M	
reasc4	If forced read option is not selected		O	
reasc4a	User-FP transmits a disabled forced read option to the server-FP via the control data		reasc4:M	
reasc5	Status/contents of file at the server site are unaffected		M	
reasc6	For ASCII and Image data types, records are retrieved based on their relative position from the start of the file		M	
reasc7	For the CCSDS Packet structure, records accessed based on the CCSDS Packet Sequence Count		M	
upsc	UPDT Command		M	
upsc1	Control data is transferred via the data connection		M	

upsc2	User-FP computes checksum against a local copy of the remote file		M	
upsc3	User-FP tests update against local copy of the remote file		M	
upsc4	Server-FP verifies remote target file is correct file to modify by comparing remote target file's checksum with that in control data		M	
upsc5	If the checksums mismatch, server-FP notifies user-FP of mismatch via standard reply means		M	
upsc6	If new remote file already exists at the server, server-FP issues '550 File Action Not Taken, File Already Exists'		M	
upsc7	Server-FP applies update signals to a copy of the remote file and saves revised data in specified new file		M	
upsc8	For the Image data types, records are accessed based on their relative position from start of file		M	
upsc9	For CCSDS Packet structure, records are accessed based on CCSDS Packet Sequence Count		M	
sizsc	SIZE Command		M	
sizsc1	Server communicates size of the referenced file subject to data type via the reply text		M	
susc	SUPP Command		M	
susc1	Server enables the suppression of reply text		M	
susc2	Server excludes reply text in subsequent replies (excluding PASV, SIZE, INTR) until the suppression of reply text is disabled		M	
rssc	REST Command		M	
rssc1	Server saves restart marker for later use		M	
rssc2	Server advances the file pointer to restart marker		M	

C2.8.5 FP Replies

Item	Protocol Feature	Reference	Status	Support
rplm	Multiple replies are distinguishable	RFC 959: 4.2	M	
rplr	Failure at any point during execution of a sequenced group of commands necessitates repetition of entire sequence from beginning		M	
rplu1	User-FP uses only the high digit of reply	RFC 1123: 4.1.2.11	M	
rplu2	User-FP handles multi-line reply lines		M	
rplu3	User-FP detects closing of control connection by interpreting 421 reply		X	
rplu4	User-FP allows only SYNCH and IP Telnet commands	RFC 1123: 4.1.2.12	M	
rplu5	User-FP negotiates Telnet options		X	
rpls1	Server-FP generates at least one reply for each command	RFC 959: 4.2	M	
rpls2	Server-FP adheres to syntax: 3-digit code, space, text, Telnet end of line code		M	
rpls3	Server-FP permits multi-line reply and adheres to the syntax for it		M	
rpls4	Server-FP strictly follows encoding specifications for reply codes		M	
rpls5	Server-FP allows reply text suppression	SCPS-FP: 3.3.3	M	
rpls6	Server-FP send only correct reply format	RFC 1123: 4.1.2.11	M	
rpls7	Server-FP uses defined reply code high digit		M	
rpls8	Server-FP uses new reply code following section 4.2, RFC 959		O	
rpls9	Server-FP handles Telnet options	RFC 1123: 4.1.2.12	M	

C2.8.6 Reply Codes

Item	Protocol Feature	Reference	Status	Support
rplc1	Server-FP sets the reply codes in RFC 959 according to the FP reply and sequencing	RFC 959: 4.2.1	M	
rplc2	Server-FP uses the suggested SCPS FP reply text on indicated commands	SCPS-FP: 3.3.4.2	O	
rplc3	Server-FP supports 256 reply code and reply text		M	
rplc4	Server-FP supports 258 reply code	SCPS-FP: 3.3.4	M	
rplc5	Server-FP supports 259 reply code		M	
rplc6	Server-FP supports 554 reply code		M	
rplc7	Server-FP supports 555 reply code		M	
rplc8	Server-FP supports 556 reply code		M	
rplc9	Server-FP supports 557 reply code		M	

C2.9 DECLARATIVE SPECIFICATIONS

C2.9.1 Minimum Implementation

Item	Protocol Feature	Reference	Status	Support
mimpl	Minimum Implementation	SCPS-FP: 3.4.1	M	
mimpl1	Image Type	RFC 959: 4.1.2	M	
mimpl2	Stream Mode		M	
mimpl3	File Structure		M	
mimpl4	Commands:			
mimpl4d	IDLE	SCPS-FP: 3.3.3	M	
mimpl4g	INTR		M	
mimpl4h	REST		M	
mimpl4i	SIZE		M	
mimpl4j	READ		M	
mimpl4k	UPDT		M	
mimpl4m	SUPP		M	
mimpl4n	NSUP		M	
mimpl4o	ARST		M	
mimpl4p	NARS		M	
mimpl4r	TYPE	RFC 959: 4.1.2	M	
mimpl4s	STRU		M	
mimpl4a	ABOR	RFC 959: 4.1.3	M	
mimpl4b	RETR		M	
mimpl4c	STOR		M	
mimpl4e	QUIT		M	
mimpl4l	DELE		M	
mimpl4q	SITE		M	
osimp	Optional commands based on underlying file system or operating system capabilities:			
osimp1a	LIST	RFC 959: 4.1.3	O	
osimp1b	MKD		O	
osimp1c	RMD		O	
osimp1d	CWD		O	
omimp	Optional based on mission needs:			
omimp1	ASCII Type	RFC 959: 4.1.2	O	
omimp2	Commands:			
omimp2k	BETS	SCPS-FP: 3.3.3	O	
omimp2l	NBES		O	
omimp2f	USER	RFC 959: 4.1.1	O	
omimp2a	PORT	RFC 959: 4.1.2	O	
omimp2b	PASV		O	
omimp2e	MODE		O	
omimp2g	PASS		O	
omimp2h	RNFR	RFC 959: 4.1.3	O	
omimp2i	RNTO		O	
omimp2j	STAT		O	

C2.9.2 Command Syntax

Item	Protocol Feature	Reference	Status	Support
cosy1	Command code length is 4 chars or less	RFC 959: 5.3	M	
cosy2	Processing of FP commands is case sensitive		X	
cosy3	Processing of symbols in arguments of FP commands is case sensitive		X	
cosy4	Each argument field is variable length character string ending with a space or the character sequence <CRLF>		M	
cosy5	Server takes no action until end of line code is received		M	
cosy6	All characters in the argument field are ASCII including ASCII represented decimal integers		M	

C2.9.3 SCPS-FP Command Syntax

Item	Protocol Feature	Reference	Status	Support
fcsy	SCPS-FP command syntax:			
fcsy01	USER	RFC 959: 5.3.1	M	
fcsy02	PASS		M	
fcsy03	CWD		M	
fcsy04	QUIT		M	
fcsy05	PORT		M	
fcsy06	PASV		M	
fcsy07	TYPE		M	
fcsy08	STRU		M	
fcsy09	MODE		M	
fcsy10	RETR		M	
fcsy11	STOR		M	
fcsy12	RNFR		M	
fcsy13	RNTO		M	
fcsy14	REST		M	
fcsy15	ABOR		M	
fcsy16	DELE		M	
fcsy17	RMD		M	
fcsy18	MKD		M	
fcsy19	LIST		M	
fcsy20	STAT		M	
fcsy21	SITE		M	
fcsy22	HELP		M	

Item	Protocol Feature	Reference	Status	Support
fcsy23	ARST	SCPS-FP: 3.4.2	M	
fcsy24	NARS		M	
fcsy25	INTR		M	
fcsy26	NSUP		M	
fcsy27	READ		M	
fcsy28	SIZE		M	
fcsy29	SUPP		M	
fcsy30	UPDT		M	
fcsy31	IDLE		M	
fcsy1	BETS		M	
fcsy2	NBES		M	

C2.9.4 SCPS-FP Command Arguments

Item	Protocol Feature	Reference	Status	Support
fcar	SCPS-FP command arguments			
fcar01	<username>	RFC 959: 5.3.2	M	
fcar02	<password>		M	
fcar03	<string>		M	
fcar04	<char>		M	
fcar05	<marker>		M	
fcar06	<host-port>		M	
fcar07	<host-number>		M	
fcar08	<port-number>		M	
fcar09	<number>		M	
fcar10	<form-code>		M	
fcar11	<type-code>		M	
fcar12	<structure-code>		M	
fcar13	<mode-code>		M	
fcar14	<decimal-integer>		M	
fcar15	<pathname> for file system-based storage	SCPS-FP: 3.4.2.2	M	
fcar16	<pathname> for memory-based storage		M	
fcar17	<update-control-data>		M	
fcar18	<read-control-data>		M	
fcar19	<BETS-fill-code>		M	

C2.9.5 Sequencing of Commands and Replies

Item	Protocol Feature	Reference	Status	Support
crsqu	User-FP waits for initial primary success or failure response before sending further commands	RFC 959: 5.4	M	
crsqs1	Server-FP sends spontaneous replies		X	
crsqs2	Server-FP alters meaning and action implied by reply code numbers and by the specific reply sequence		X	
crsq3	Server-FP alters reply text		O	
crsqc	Sequencing for commands:			
crsqc01	USER		M	
crsqc02	PASS		M	
crsqc03	CWD		M	
crsqc04	QUIT		M	
crsqc05	PORT		M	
crsqc06	PASV		M	
crsqc07	TYPE		M	
crsqc08	STRU		M	
crsqc09	MODE		M	
crsqc10	RETR		M	
crsqc11	STOR		M	
crsqc12	RNFR		M	
crsqc13	RNTO		M	
crsqc14	REST		M	
crsqc15	ABOR		M	
crsqc16	DELE		M	
crsqc17	RMD		M	
crsqc18	MKD		M	
crsqc19	LIST		M	
crsqc20	STAT		M	
crsqc21	SITE		M	
crsqc22	HELP		M	
crsqc23	ARST	SCPS-FP: 3.4.3	M	
crsqc24	NARS		M	
crsqc25	INTR		M	
crsqc26	NSUP		M	
crsqc27	READ		M	
crsqc28	SIZE		M	
crsqc29	SUPP		M	
crsqc30	UPDT		M	
crsqc31	IDLE		M	
crsqc32	BETS		M	
crsqc33	NBES		M	

C2.10 STATE DIAGRAMS

Item	Protocol Feature	Reference	Status	Support
stli	User-FP uses the Login state machine for	RFC 959: 6		
stli1	USER		M	
stli2	PASS		M	
stnd	User-FP uses the No Data Transfer state machine for			
stnd01	CWD		M	
stnd02	QUIT		M	
stnd03	PORT		M	
stnd04	PASV		M	
stnd05	TYPE		M	
stnd06	STRU		M	
stnd07	MODE		M	
stnd08	ABOR		M	
stnd09	DELE		M	
stnd10	RMD		M	
stnd11	MKD		M	
stnd12	STAT		M	
stnd13	SITE		M	
stnd14	HELP		M	
stnd15	ARST	SCPS-FP: 3.5	M	
stnd16	NARS		M	
stnd17	INTR		M	
stnd18	NSUP		M	
stnd19	SIZE		M	
stnd20	SUPP		M	
stnd21	IDLE		M	
stnd22	BETS		M	
stnd23	NBES		M	
stren	User-FP uses Rename state machine for			
stren1	RNFR	RFC 959: 6	M	
stren2	RNTO		M	
stdt	User-FP uses the Data Transfer state machine for		M	
stdt1	RETR	RFC 959: 6	M	
stdt2	STOR		M	
stdt3	LIST		M	
stdt4	READ	SCPS-FP: 3.5	M	
stdt5	UPDT		M	
stres	User-FP uses the Restart state machine for REST		M	

C2.10.1 Management Information Base (MIB) Requirements

Item	Protocol Feature	Reference	Status	Support
mib	Default FP configuration parameters are maintained for	SCPS-FP: 4	M	
mib1	Data transfer type	SCPS-FP: 4.2	M	
mib2	Transmission mode	SCPS-FP: 4.3	M	
mib3	Data structure	SCPS-FP: 4.4	M	
mib4	Autorestart	SCPS-FP: 4.5	M	
mib5	Maximum number of automatic restarts	SCPS-FP: 4.6	M	
mib6	PORT command	SCPS-FP: 4.7	M	
mib7	Reply Text Suppression	SCPS-FP: 4.8	M	
mib8	Server Idle Time-out	SCPS-FP: 4.9	M	
mib9	BETS option	SCPS-FP: 4.10	M	
mib10	BETS fill code	SCPS-FP: 4.11	M	

C2.10.2 SCPS User Interface

Item	Protocol Feature	Reference	Status	Support
ui1	Arbitrary pathnames can be processed	RFC 1123 4.1.4.1	M	
ui2	Error messages displayed to user as soon as possible	RFC 1123 4.1.4.3	M	
ui3	Synchronization is maintained with the server		M	

ANNEX D

SCPS-FP SERVICE SPECIFICATION

(This annex **is** part of the Recommendation.)

D1 OVERVIEW OF THE FILE SERVICE

D1.1 The SCPS File services are defined in terms of primitives that are independent of specific implementation approaches. The SCPS File services require a Transport layer that provides error-free, in-order delivery of File-PDUs. The prefixes used to distinguish between the SCPS File Service provider and the Transport Service provider are as follows:

- F- File Service provider;
- T- Transport Service provider.

D1.2 The SCPS File Service provides for the end-to-end reliable transfer of file-oriented data sets. It includes the following service classes:

- access to the responding File Service provider;
- management of the remote File System;
- file transfer;
- record access.

D2 SERVICES PROVIDED BY THE FILE LAYER

D2.1 PRIMITIVES

All primitives are confirmed, and all are requested by the initiating File Service provider.

Primitive	Class	Parameters
F-CLOSE	Access	-
F-OPEN	Access	Host_Name
F-PASS_WORD	Access	Pass_Word
F-USER	Access	User_Name
F-AUTORESTART	Configuration	-
F-BETS	Configuration	-
F-IDLE	Configuration	Number_Of_Seconds
F-NOAUTORESTART	Configuration	-
F-NOBETS	Configuration	-
F-SENDPORT	Configuration	-
F-STATUS	Configuration	-
F-STRUCT	Configuration	Structure_ID
F-SUPPRESS	Configuration	-

Primitive	Class	Parameters
F-TYPE	Configuration	Data_Type
F-UNSENDPORT	Configuration	-
F-UNSUPPRESS	Configuration	-
F-CHANGE_DIRECTORY	File Management	Directory_Name
F-COPY	File Management	File_Name1 File_Name2
F-DELETE	File Management	File_Name
F-GET_SIZE	File Management	File_Name
F-LIST	File Management	File_or_Directory_Specification
F-MAKE_DIRECTORY	File Management	Directory_Name
F-REMOVE_DIRECTORY	File Management	Directory_Name
F-RENAME	File Management	File_Name1 File_Name2
F-ABORT	File Transfer	-
F-GET	File Transfer	File_Name1 File_Name2
F-INTERRUPT	File Transfer	-
F-PORT	File Transfer	-Address Port
F-PUT	File Transfer	File_Name1 File_Name2
F-RECORD_READ	File Transfer	File_Name1 File_Name2 Record_Set_Specification Force_Read
F-RECORD_UPDATE	File Transfer	File_Name1 File_Name2 File_Name3 File_Name4 Record_Update_Script
F-RESTART	File Transfer	Restart_Point

D2.2 RESPONSES

D2.2.1 The responding File Service provider issues the following primitives to provide notification of various events:

- F-POSITIVE_PRELIMINARY.response();
- F-POSITIVE_COMPLETION.response();
- F-POSITIVE_INTERMEDIATE.response();
- F-NEGATIVE_TRANSIENT.response();
- F-NEGATIVE_PERMANENT.response().

D2.2.2 The responding File Service provider passes a code to provide details about the event.

D2.2.3 A positive preliminary response means that the responding File Service provider has begun processing a request. The user should expect another response before making another request.

D2.2.4 A positive completion response means that the request has been successfully serviced.

D2.2.5 A positive intermediate response means that processing of the request by the responding File Service is in progress, but more information is needed from the initiating File Service provider to complete the request.

D2.2.6 A negative transient response means that the responding File Service provider could not service the request, but the error condition is temporary.

D2.2.7 A negative completion response means that the responding File Service provider could not service the request because of a permanent error condition. The user must expect this possible response after issuing any request.

D2.2.8 In general, the File Service user must wait for the File Service provider to issue a completion response primitive before issuing a subsequent request. The exceptions to this are only when the initiating File Service user wishes to interrupt or abort a file transfer. A completion response is any one of the following:

- F-POSITIVE_COMPLETION.response();
- F-NEGATIVE_TRANSIENT.response();
- F-NEGATIVE_PERMANENT.response().

D2.3 STATE DIAGRAMS

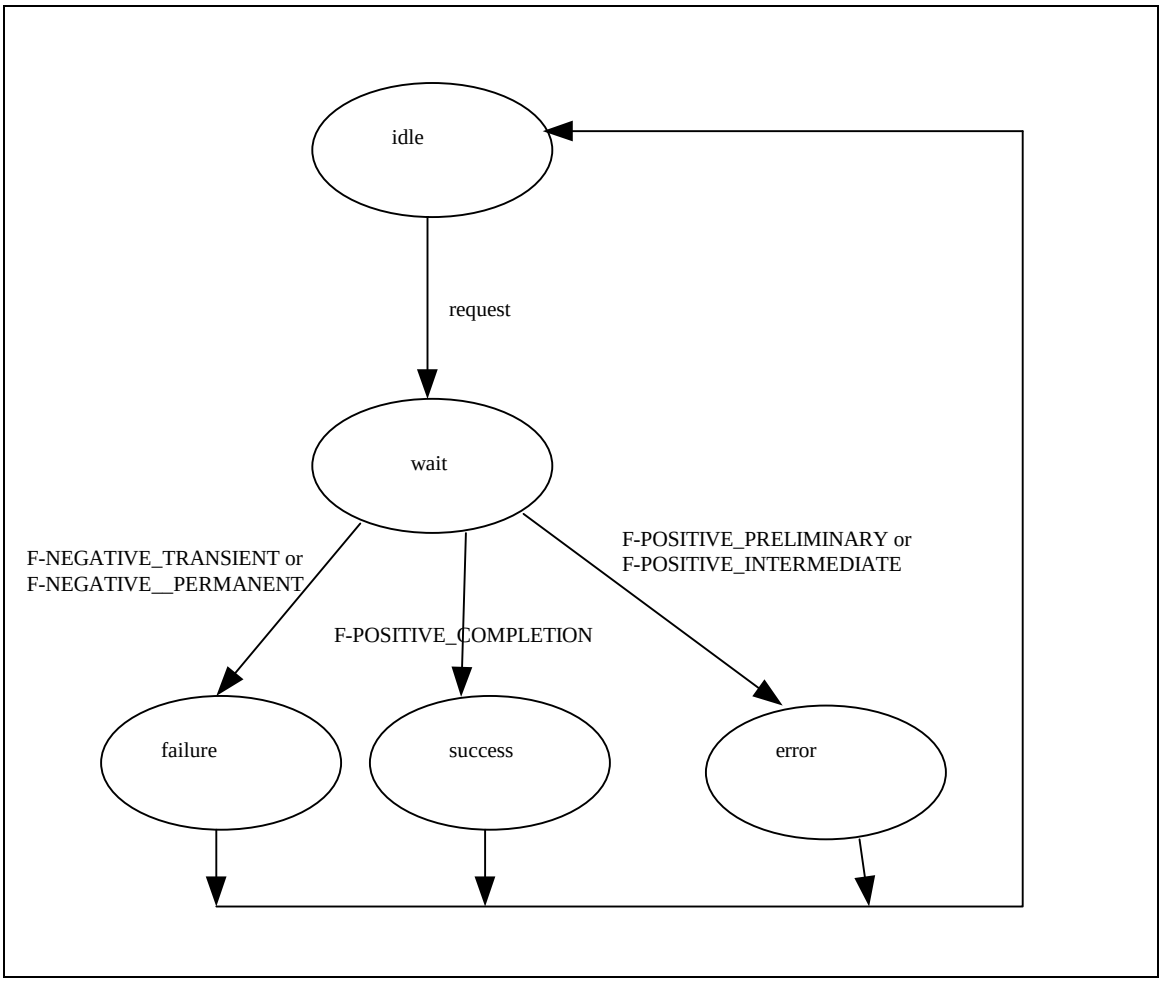


Figure D-1: State Diagram 1

D2.3.1 Figure D-1 models the following requests:

F-CLOSE	F-STRUCT	F-GET_SIZE
F-AUTORESTART	F-SUPPRESS	F-MAKE_DIRECTORY
F-BETS	F-TYPE	F-REMOVE_DIRECTORY
F-IDLE	F-UNSUPPRESS	F-RENAME
F-NOAUTORESTART	F-CHANGE_DIRECTORY	F-ABORT
F-NOBETS	F-COPY	F-INTERRUPT
F-STATUS	F-DELETE	F-RESTART

D2.3.2 The F-POSITIVE_PRELIMINARY and F-POSITIVE_INTERMEDIATE responses are invalid.

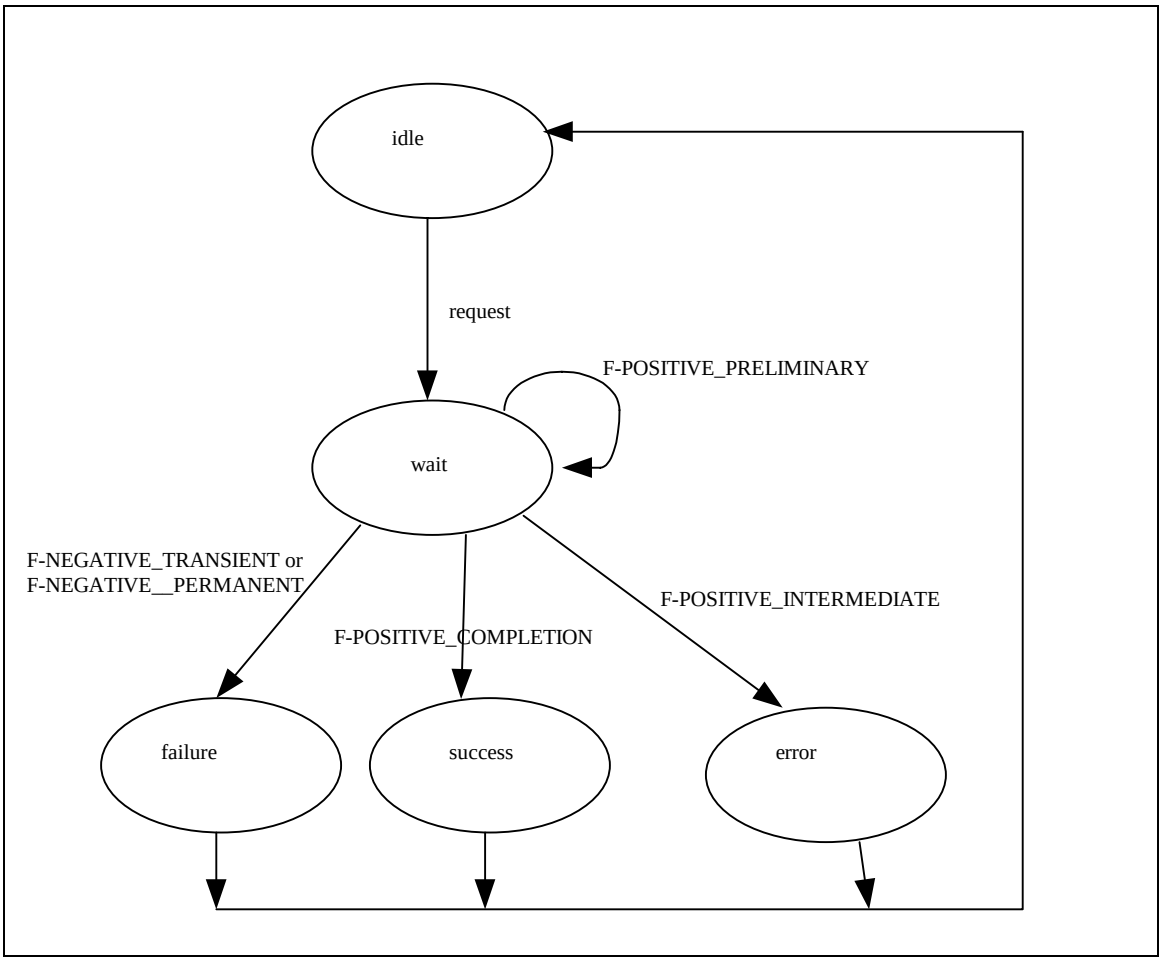


Figure D-2: State Diagram 2

D2.3.3 Figure D-2 models the following primitives:

- F-LIST;
- F-GET;
- F-PUT;
- F-RECORD_READ;
- F-RECORD UPDATE.

D2.3.4 While in the wait state, the responding File Service must be able to process an F-ABORT or F-INTERRUPT request from the initiating File Service. It processes these requests as shown in State Diagram 1. Upon completion of the F-ABORT or F-INTERRUPT, the responding File Service issues the response to the interrupt, followed by the response to the original request (F-LIST, F-GET, or F-PUT).

D2.3.5 F-POSITIVE_INTERMEDIATE is an invalid response to these primitives.

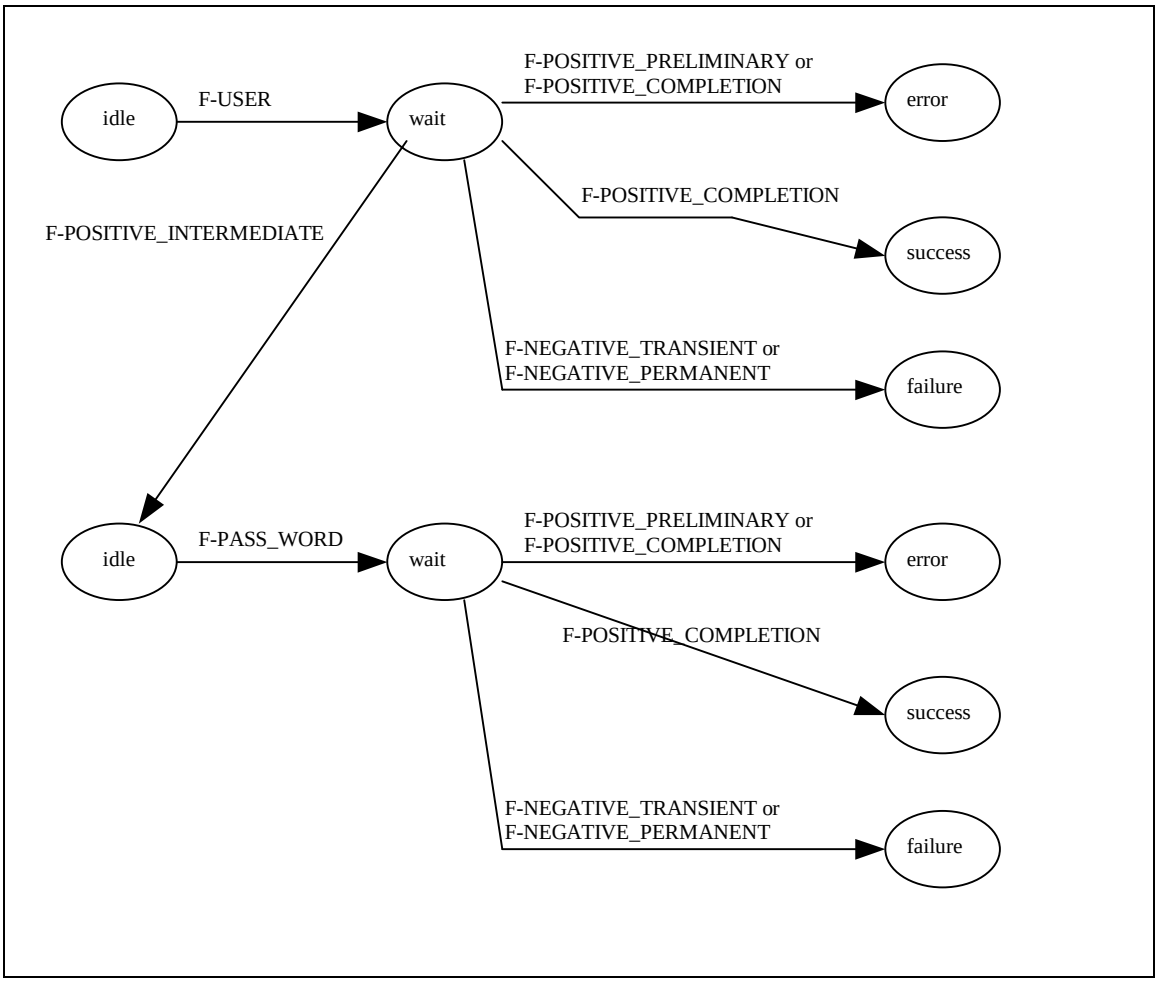


Figure D-3: State Diagram 3

D2.3.6 Figure D-3 shows the state transitions for F-USER/F-PASS_WORD indications.

D2.4 ACCESS PRIMITIVES

D2.4.1 F-CLOSE.request()

D2.4.1.1 The user issues F-CLOSE.request() to terminate a connection to the peer File Service. The initiating File Service issues T-CLOSE.request() to the Transport Service regardless of the response from the peer File Service.

D2.4.1.2 Upon success, the File Service issues F-POSITIVE_COMPLETION.response().

D2.4.1.3 Upon failure, the File Service issues F-NEGATIVE_COMPLETION.response().

D2.4.2 F-OPEN.request()

D2.4.2.1 The user issues F-OPEN.request() to establish a connection to the peer File Service.

D2.4.2.2 Upon success, the File Service issues F-POSITIVE_COMPLETION.response() if the connection is successfully established, and F-NEGATIVE_TRANSIENT.response() if the connection is not successfully established.

D2.4.3 F-PASS_WORD.request(password)

D2.4.3.1 The user issues F-PASS_WORD.request(password) to supply the password necessary to gain access to the peer File Service. The parameter is the password to send to the peer File Service.

D2.4.3.2 The File Service issues F-POSITIVE_COMPLETION.response() if the password is valid, indicating the user has access to the peer File Service.

D2.4.3.3 The File Service issues F-NEGATIVE_PERMANENT.response() if the password is not valid.

D2.4.4 F-USER.request(User_Name)

D2.4.4.1 The user issues F-USER.request(User_Name) to gain access to the peer File Service. The parameter User_Name is the name of an account on the remote computer system.

D2.4.4.2 The File Service issues F-POSITIVE_INTERMEDIATE.response() if User_Name is valid. The File Service must supply the valid password to gain access to the peer File Service.

D2.5 CONFIGURATION PRIMITIVES

D2.5.1 F-AUTORESTART.request()

D2.5.1.1 The user issues F-AUTORESTART.request() to cause the responding File Service to enter autorestart mode.

D2.5.1.2 The File Service issues F-POSITIVE_COMPLETION.response() in reply.

D2.5.2 F-BETS.request()

D2.5.2.1 The user issues F-BETS.request() to cause the responding File Service to use BETS on the data connection. The Best Effort Transport Service (BETS) is defined in the SCPS-TP Recommendation (reference [4]).

D2.5.2.2 The responding File Service replies by sending F-POSITIVE_COMPLETION.response().

D2.5.3 F-IDLE.request(Number_Of_Seconds)

D2.5.3.1 The user issues F-IDLE.request(Number_Of_Seconds) to set the value of the idle countdown timer. The parameter is the number of idle seconds the responding File Service waits before closing the connection.

D2.5.3.2 The responding File Service replies by sending the F-POSITIVE_COMPLETION.response().

D2.5.4 F-NOAUTORESTART.request()

D2.5.4.1 The user issues F-NOAUTORESTART.request() to cause the responding File Service to exit autorestart mode.

D2.5.4.2 The File Service issues F-POSITIVE_COMPLETION.response() in reply.

D2.5.5 F-NOBETS.request()

D2.5.5.1 The user issues F-NOBETS.request() to cause the responding File Service to turn off BETS during file transfers.

D2.5.5.2 The responding File Service replies by sending F-POSITIVE_COMPLETION.response().

D2.5.6 F-SENDPORT.request()

The user issues F-SENDPORT.request() to cause the client to issue the F-PORT.request(Address, Port) to establish the data connection prior to file transfer operations. The client issues F-POSITIVE.response().

D2.5.7 F-STATUS.request()

D2.5.7.1 The user issues F-STATUS.request() to cause the responding File Service to report the values of its configuration parameters.

D2.5.7.2 The responding File Service replies by sending F-POSITIVE_COMPLETION.response().

D2.5.8 F-STRUCT.request(Structure_ID)

D2.5.8.1 The user issues F-STRUCT.request(Structure_ID) to configure the file structure in the responding File Service.

D2.5.8.2 The responding File Service replies by sending F-POSITIVE_COMPLETION.response().

D2.5.9 F-SUPPRESS.request()

D2.5.9.1 The user issues F-SUPPRESS.request() to configure the responding File Service to send only the code of a reply, and not the text.

D2.5.9.2 The responding File Service replies by sending F-POSITIVE_COMPLETION.response().

D2.5.10 F-TYPE.request(Data_Type)

D2.5.10.1 The user issues F-TYPE.request(Data_Type) to configure the file data type in the responding File Service.

D2.5.10.2 If the data type is valid and supported, the File Service issues F-POSITIVE_COMPLETION.response().

D2.5.10.3 If the data type is either invalid or not supported, the File Service issues F-NEGATIVE_COMPLETION.response().

D2.5.11 F-UNSENDPORT.request()

The user issues F-UNSENDPORT.request() to cause the client to use the same address and port number for every file transfer operation. The client issues F-POSITIVE.response().

D2.5.12 F-UNSUPPRESS.request()

D2.5.12.1 The user issues F-UNSUPPRESS.request() to configure the responding File Service to send the reply text with the reply code.

D2.5.12.2 The responding File Service replies by sending F-POSITIVE_COMPLETION.response().

D2.6 FILE MANAGEMENT PRIMITIVES

D2.6.1 F-CHANGE_DIRECTORY.request(Directory_Name)

D2.6.1.1 The user issues F-CHANGE_DIRECTORY.request(Directory_Name) to change the current working directory in the responding File Service's file system.

D2.6.1.2 If successful, the responding File Service issues F-POSITIVE_COMPLETION.response().

D2.6.1.3 If unsuccessful, the responding File Service issues F-NEGATIVE_PERMANENT.response().

D2.6.2 F-COPY.request(File_Name1, File_Name2)

D2.6.2.1 The user issues F-COPY.request(File_Name1, File_Name2) to cause the responding File Service to copy the file specified by File_Name1 to the file specified by File_Name2.

D2.6.2.2 If successful, the responding File Service issues F-POSITIVE_COMPLETION.response().

D2.6.2.3 If unsuccessful, the responding File Service issues F-NEGATIVE_PERMANENT.response().

D2.6.3 F-DELETE.request(File_Name)

D2.6.3.1 The user issues F-DELETE.request(File_Name) to cause the responding File Service to delete the file specified by File_Name.

D2.6.3.2 If successful, the responding File Service issues F-POSITIVE_COMPLETION.response().

D2.6.3.3 If unsuccessful, the responding File Service issues F-NEGATIVE_PERMANENT.response().

D2.6.3.4 If the file cannot be deleted because it is in use by another application, the responding File Service issues F-NEGATIVE_TRANSIENT.response().

D2.6.4 F-GET_SIZE.request(File_Name)

D2.6.4.1 The user issues F-GET_SIZE.request(File_Name) to cause the responding File Service to report the size of the specified file.

D2.6.4.2 If successful, the responding File Service issues F-POSITIVE_COMPLETION.response().

D2.6.4.3 If unsuccessful, the responding File Service issues F-NEGATIVE_ PERMANENT.response().

D2.6.5 F-LIST.request(File_or_Directory_Specification)

D2.6.5.1 The user issues F-LIST.request(File_or_Directory_Specification) to cause the responding File Service to report the attributes of the specified file, the contents of the specified directory, or if a null parameter is supplied, the contents of the current working directory.

D2.6.5.2 The responding File Service establishes a data connection and issues F-POSITIVE_PRELIMINARY.response(). If a data connection could not be established, the responding File Service issues F-NEGATIVE_TRANSIENT.response().

D2.6.5.3 Upon establishing the data connection, the responding File Service uses it to transmit the report.

D2.6.5.4 When transmission is complete, the responding File Service issues F-POSITIVE_COMPLETION.response().

D2.6.5.5 If unsuccessful, the responding File Service issues F-NEGATIVE_TRANSIENT.response().

D2.6.6 F-MAKE_DIRECTORY.request(File_or_Directory_Specification)

D2.6.6.1 The user issues F-MAKE_DIRECTORY.request(File_or_Directory_Specification) to cause the responding File Service to create a directory with the specified name.

D2.6.6.2 If successful, the responding File Service issues F-POSITIVE_COMPLETION.response().

D2.6.6.3 If unsuccessful, the responding File Service issues F-NEGATIVE_PERMANENT.response().

D2.6.7 F-REMOVE_DIRECTORY.request(Directory_Name)

D2.6.7.1 The user issues F-REMOVE_DIRECTORY.request(Directory_Name) to cause the responding File Service to delete the directory with the specified name.

D2.6.7.2 If successful, the responding File Service issues F-POSITIVE_COMPLETION.response().

D2.6.7.3 If unsuccessful, the responding File Service issues F-NEGATIVE_PERMANENT.response().

D2.6.8 F-RENAME.request(File_Name1, File_Name2)

D2.6.8.1 The user issues F-RENAME.request(File_Name1, File_Name2) to cause the responding File Service to give the file specified by File_Name1 the name given in File_Name2.

D2.6.8.2 If successful, the responding File Service issues F-POSITIVE_COMPLETION.response().

D2.6.8.3 If unsuccessful, the responding File Service issues F-NEGATIVE_ PERMANENT.response().

D2.7 FILE TRANSFER PRIMITIVES

D2.7.1 F-ABORT.request()

D2.7.1.1 The user issues F-ABORT.request() to abort a transfer either to or from the responding File Service.

D2.7.1.2 If a transfer is not in progress the responding File Service issues F-NEGATIVE_PERMANENT.response().

D2.7.1.3 If a transfer is in progress, the sending File Service terminates the transmission. The responding File Service issues F-POSITIVE_COMPLETION.response(), in reply to the F-ABORT.request, and issues F-POSITIVE_COMPLETION.response() in reply to the operation that was in progress.

D2.7.2 F-GET.request(File_Name1, File_Name2)

D2.7.2.1 The user issues F-GET.request(File_Name1, File_Name2) to cause the responding File Service to send the specified file specified by File_Name2. The initiating File Service stores the incoming file under the name given by File_Name1.

D2.7.2.2 The responding File Service establishes a data connection and issues F-POSITIVE_PRELIMINARY.response(). If the data connection could not be established, the responding File Service issues F-NEGATIVE_TRANSIENT.response().

D2.7.2.3 Upon establishing the data connection, the responding File Service uses it to transmit the contents of the file.

D2.7.2.4 When transmission is complete, the responding File Service issues F-POSITIVE_COMPLETION.response().

D2.7.2.5 If unsuccessful, the responding File Service issues F-NEGATIVE_TRANSIENT.response().

D2.7.3 F-INTERRUPT.request()

D2.7.3.1 The user issues F-INTERRUPT.request() to interrupt a transfer either to or from the responding File Service.

D2.7.3.2 If a transfer is not in progress the responding File Service issues F-NEGATIVE_PERMANENT.response().

D2.7.3.3 If a transfer is in progress, the sending File Service terminates the transmission. The responding File Service issues F-POSITIVE_COMPLETION.response(), in reply to the F-INTERRUPT.request, and issues F-POSITIVE_COMPLETION.response() in reply to the operation that was in progress.

D2.7.4 F-PORT.request(Address, Port)

The client File Service issues F-PORT.request(Address, Port) to set the address and port on which the data connection will be established. The responding File Service checks the syntax of the parameters. If the syntax is correct, the server File Service stores the address and port number for later use and issues F-POSITIVE_COMPLETION.response(). Otherwise it issues F-NEGATIVE_COMPLETION.response().

D2.7.5 F-PUT.request(File_Name1, File_Name2)

D2.7.5.1 The user issues F-PUT.request(File_Name1, File_Name2) to cause the initiating File Service to send the contents of the file specified by File_Name1 to the responding File Service. The responding File Service stores the file under the name given in File_Name2.

D2.7.5.2 The responding File Service establishes a data connection and issues F-POSITIVE_PRELIMINARY.response(). If the data connection could not be established, the responding File Service issues F-NEGATIVE_TRANSIENT.response().

D2.7.5.3 Upon receiving the F-POSITIVE_PRELIMINARY.response(), the initiating File Service sends the contents of the specified file on the data connection.

D2.7.5.4 When transmission is complete, the responding File Service issues F-POSITIVE_COMPLETION.response().

D2.7.5.5 If unsuccessful, the responding File Service issues F-NEGATIVE_TRANSIENT.response().

D2.7.6 F-RECORD_READ.request(File_Name1, File_Name2, Record_Set_Specification, Force_Read)

D2.7.6.1 The user issues F-RECORD_READ.request(File_Name1, File_Name2, Record_Set_ Specification, Force_Read), to cause the responding File Service to send only the selected records of the file specified by File_Name1. The initiating File Service stores the received records in the file named by File_Name2. The Record_Set_Specification file lists the records selected for transfer. The responding File Service ignores read errors if the Force_Read argument is ‘TRUE’.

D2.7.6.2 The responding File Service establishes a data connection and issues F-POSITIVE_PRELIMINARY.response(). If the data connection could not be established, the responding File Service issues F-NEGATIVE_TRANSIENT.response().

D2.7.6.3 Upon establishing the data connection, the responding File Service uses it to transmit the selected records.

D2.7.6.4 When transmission is complete, the responding File Service issues F-POSITIVE_COMPLETION.response().

D2.7.6.5 If unsuccessful, the responding File Service issues F-NEGATIVE_TRANSIENT.response().

D2.7.7 F-RECORD_UPDATE.request(File_Name1, File_Name2, File_Name3, File_Name4, Record_Update_Script)

D2.7.7.1 The user issues F-RECORD_UPDATE.request(File_Name1, File_Name2, File_Name3, File_Name4, Record_Update_Script) to cause the responding File Service to execute the Record_Update_Script on the file specified by File_Name1. The Record_Update_Script contains commands to add records, change records, and delete records.

D2.7.7.2 The initiating File Service executes the Record_Update_Script against the local file whose name is given in File_Name3. The initiating File Service assumes the contents of the file named File_Name3 and the file named File_Name1 are identical. If the update fails, the initiating File Service issues F-NEGATIVE_PERMANENT.response(). If it succeeds, the initiating File Service calculates the CRC of the resulting file and stores it in local RAM and issues F-UPDATE.request() to the responding File Service.

D2.7.7.3 The responding File Service executes the Record_Update_Script on the file whose name is given in File_Name1 and writes the result to a file whose name is given in File_Name2. If the update fails, the responding File Service issues F-NEGATIVE_TRANSIENT.response(). Otherwise it calculates the CRC and issues F-POSITIVE_COMPLETION.response().

D2.7.7.4 The initiating File Service compares the CRC stored in local RAM to the one passed in the reply from the responding File Service. If it matches, it issues F-POSITIVE_COMPLETION.response() to the user. Otherwise it issues F-NEGATIVE_TRANSIENT.response() to the user.

D2.7.8 F-RESTART.request(Restart_Point)

D2.7.8.1 The user issues F-RESTART.request(Restart_Point) to set the restart point of the next transfer. The responding File Service stores the Restart_Point in RAM and issues F-POSITIVE_ INTERMEDIATE.response().

D2.7.8.2 The initiating File Service must now issue a transfer request such as F-PUT or F-GET.

D2.7.8.3 This request is used to restart failed transfers.

D3 SERVICES ASSUMED FROM THE LOWER LAYER

- T-PASSIVE_OPEN.request;
- T-ACTIVE_OPEN.request;
- T-SEND_DATA_OVER_CONNECTION.request;
- T-RECEIVE_DATA_FROM_CONNECTION.request;
- T-CLOSE.request;
- T-ABORT.request;
- T-CONNECTION_STATUS.request;
- T-OPEN_ID.confirm;
- T-OPEN_FAILURE.confirm;
- T-OPEN_SUCCESS.confirm;
- T-DELIVER_CONNECTION_DATA.confirm;
- T-CLOSING.indication;
- T-TERMINATE.indication;
- T-CONNECTION_ERROR.indication.

D4 SERVICES ASSUMED FROM THE OPERATION ENVIRONMENT

D4.1 OVERVIEW

D4.1.1 This subsection summarizes the file system, operating systems, and transport layer services on which the client FP and server FP rely.

D4.1.2 The typical SCPS-FP user service(s) that would use the file system or operating system service is specified for each service. An implementation may employ additional services that are not listed here and/or some of the same services for different user commands.

D4.2 FILE SYSTEM DEPENDENCIES

The File Service relies on the resident file system to provide the following services. If the host lacks a file system, then software may have to be developed to provide equivalent services.

FILE SYSTEM SERVICE
Change current working directory
Change root directory for a command
Change the name of a file
Control a device
Create a directory
Create/Open File or Close a file
Determine accessibility of file
Get current working directory pathname
Move the read or write file pointer to a specific location
Obtain status information about a file
Open a file or Close a file
Open directory, Read next directory entry, or Close directory
Open directory, Read next directory entry, or Close directory
Open system log
Read from file
Remove (unlink) files or directories
Set file creation mode mask
Write to file

D4.3 OPERATING SYSTEM DEPENDENCIES

D4.3.1 The File Service relies on the resident operating system to provide the following services. If the host operating system does not provide these services, then software may have to be developed to provide equivalent services.

OPERATING SYSTEM SERVICE
Block signals
Create an interprocess communication channel
Encode or decode a file
Execute a file/command
Find name of terminal
Get effective user ID
Get login name
Get password file entry
Get process identification
Get system time of day
Get user identity
Open or Close a pipe for I/O from or to a process
Send a signal to terminate a process
Set current signal mask
Set effective group ID (UNIX)
Set effective user ID
Spawn a new process in virtual memory (vfork)
Suspend execution for interval
Wait for process to terminate or stop

D4.3.2 The following operating system services may be needed in response to any of the user commands:

- Get system error;
- Get system time;
- Non-local Goto (jumps);
- Schedule signal after specified time (alarm);
- Simplified software signal facilities;
- Write system error to log.