Network Working Group
Request for Comments: 122
NIC 5834

James E. White
UC Santa Barbara
26 April 1971

NETWORK SPECIFICATIONS FOR UCSB's SIMPLE-MINDED FILE SYSTEM

CONTENTS

FIGURES

I. Preface

UCSB will provide file storage for Network users. UCSB's Simple Minded File System (SMFS) is addressed as socket number X'401', site 3. No accounting parameters are required. This document is intended to provide programmers with the information necessary to communicate with SMFS which conducts all Network transactions trough its NCP which operates under the Host-Host protocol of August 3, 1970.*

II. Implementation

The following information is not essential to use of SMFS but may be of interest. SMFS will store user's files on IBM 2316 disk packs, each with 29M 8-bit bytes of storage capacity. UCSB has two 2314 units, each with eight drives on-line. Initially, one drive will be allocated for Network storage, and the appropriate pack will always be mounted an that drive, and hence accessible to SMFS without operator intervention. UCSB estimates that for the next year it will have up to four drives that it can devote to Network use. The second, third, and fourth drives will be allocated only as the need arises. SMFS is written to accommodate any number of on-line drives without modification.

If necessary, UCSB will investigate the possibility of making one of the four drives a come-and-go drive on which one of a number of packs can be mounted as required. Hence, the potential exists for increased storage capacity with an accompanying increase in access time.

Files stored with SMFS will be backed up to tape daily. The back-up tape(s) will be off-line and available only in case the on-line copies are destroyed.

In no sense does USB expect to become _the_ file storage node of the Network; it hasn't the capacity. UCSB _is_ equipped, however, to make a limited amount of secondary storage immediately available to the Network community.

III. Login

SMFS can simultaneously service any number of Network users up to some assembly-parameters maximum (currently ten). A potential user must establish a pair of Network connections

*At the time of this writing, the NCP modifications of RFC #107 have not as yet been implemented at UCSB.

(i.e., one full-duplex connection) to SMFS by executing a standard ICP to socket X'401', site 3. SMFS always listens on that socket. It will accept any call it receives -- say from the user's receive socket 'm' -- and over the connection thus established transmit a 32-bit receive socket number (call it 'n'), and then close the connection. SMFS will then issue two connection requests -- one involving its receive socket 'n' and the user's send socket 'm+l', in other involving its send socket 'n+l' and the user's receive socket 'm'. Once these two connections have been established, the user will be considered logged in. A deviation from the Initial Connection Protocol will occur only if SMFS or its NCP has insufficient resources to support another connection.

SMFS will maintain its connections to the user indefinitely. It will voluntarily terminate its connections to the user only if (1) a bad op code is encountered in a user command (see Section VI), or (2) closing one of the connections is required to signal end-of-data (see Section V.D.). Barring such an occurrence, the user should close his connections to SMFS when through, at which time SMFS will consider the user logged out.

In the discussion to follow, the following terms are used. The connections on which the user transmits data to and receives data from SMFS are designated the input and output connections, respectively (i.e., SMFS's rather than the user's point of view is adopted). The string of bits which passes from the user to SMFS over the input connection during the life of that connection is called the _input stream_; the string of bits which passes from SMFS to the user over the output connection during the life of that connection is called the _output stream_.

IV. Service Offered

SMFS will provide storage for sequential, binary files of length greater than or equal to an assembly-parameter minimum (currently one bit) and less than or equal to an assembly-parameter maximum (currently 25 million bits). There is no restriction on the contents of the file.

Every file stored with SMFS has a _filename_, which may be any string of from one to 36, 8-bit characters chosen from the set:

{ A,...,Z,0,...9,blank }

Graphic		EBCDIC Code (Hex)		ASCII Code (Hex)	
UC	LC	UC	LC	UC	LC
A	a	C1	81	41	61
B	b	C2	82	42	62
C	c	C3	83	43	63
D	d	C4	84	44	64
E	e	C5	85	45	65
F	f	C6	86	46	66
G	g	C7	87	47	67
H	h	C8	88	48	68
I	i	C9	89	49	69
J	j	D1	91	4A	6A
K	k	D2	92	4B	6B
L	l	D3	93	4C	6C
M	m	D4	94	4D	6D
N	n	D5	95	4E	6E
O	o	D6	96	4F	6F
P	p	D7	97	50	70
Q	q	D8	98	51	71
R	r	D9	99	52	72
S	s	E2	A2	53	73
T	t	E3	A3	54	74
U	u	E4	A4	55	75
V	v	E5	A5	56	76
W	w	E6	A6	57	77
X	x	E7	A7	58	78
Y	y	E8	A8	59	79
Z	z	E9	A9	5A	7A
0	-	F0	-	30	-
1	-	F1	-	31	-
2	-	F2	-	32	-
3	-	F3	-	33	-
4	-	F4	-	34	-
5	-	F5	-	35	-
6	-	F6	-	36	-
7	-	F7	-	37	-
8	-	F8	-	38	-
9	-	F9	-	39	-
blank	-	40	-	20	-

Figure 1

Filename/Password Character Sets

Filenames may be specified by the user in either EBCDIC or ASCII (see Figure 1), and the characters A,...,Z may be either upper- or lower-case. However, the acceptance by SMFS of both upper- and lower-case, and both EBCDIC and ASCII, is provided only as a convenience to the user. In particular, such distinctions don't increase the number of unique filenames that can be generated; the filenames 'FILE NUMBER 1' and 'file number 1', in EBCDIC or ASCII, designate the same file.

Every file stored with SMFS may optionally be protected against unauthorized retrieval and/or modification. When a file is created, the user may associate with it a _modification password_ and/or an _access password_. Thereafter, SMFS will demand that the appropriate password be supplied before the file is modified or retrieved, respectively. Since SMFS protects each file independently against unauthorized modification and retrieval, a group of users can be given access to a file while a single individual retains the exclusive right to modify it. If no password is defined for a particular type of reference to a file, then such references are unrestricted. Passwords have the same attributes as filenames -- same length restrictions and same character sets.

Because of the manner in which SMFS writes files onto secondary storage, it must insure that while one user is modifying a file, no other user is simultaneously either modifying or retrieving the same file. This requirement is effected by a mechanism internal to SMFS and hence transparent to users, with the exception that when a user attempts to retrieve or modify a file currently being modified by another user, SMFS will delay action upon the request until the current modification is complete. There is no restriction on the number of users which may concurrently retrieve the same file.

V. Primitive File Operations

SMFS recognizes and will execute the following primitive file operations:

V.A. Allocate File (ALF)

SMFS regards the reservation of filename, the assignment of passwords, and the reservation of secondary storage as an operation distinct from that of transmitting the file's contents. The operation is called _file allocation_, abbreviated ALF. In allocating a file, the user specifies the filename to be assigned to it, the access password (if any), and the estimated size of the file in bits. SMFS checks the proposed filename to insure that it doesn't duplicate that of an existing file. SMFS also checks to insure that it has sufficient secondary storage available to accommodate the new

file. If both requirements are met, SMFS allocates the file; the filename is reserved, secondary storage is reserved, and the password information is recorded.

In reserving secondary storage for a file, SMFS adds its estimate of its overhead in storing the file to the user-declared size of the file. In general, the user should slightly over-estimate the size of his file at allocation. SMFS allocates a fixed amount of storage on the basis of that estimate, an amount which cannot be increased later. SMFS's actual overhead in storing a file is a function of the manner in which the contents of the file are transmitted by the user. The overhead is minimal when the file is transmitted in a single series of operations (see Section VI) and increases as the number of operations increases. It is the overhead associated with single-series transmission that SMFS adds to the file size specified by the user to determine the amount of storage to allocate. Hence, for multiple-series transmission, the overhead will have been underestimated.

V.B. Update File (UDF)

The operation of transmitting part or all of a previously allocated file's contents for storage by SMFS is called _updating_ the _file_ (UDF). The user specifies the filename of the file to be updated, the modification password if required, the amount of data in bits to be added to the file, and finally the data itself. SMFS locates the file on secondary storage, checks the password for validity, if appropriate, and adds the data to the file. SMFS considers the update complete when either the specified number of bits have been extracted from the input stream and stored, or when the user terminates transmission by closing the connection.

The data transmitted in a UDF operation is _concatenated_ to the current contents of the file. Boundaries between updates are transparent to the user when the file is retrieved. Hence, for example, the contents of a file might be transmitted to SMFS in two distinct UDF operations, and later retrieved in a single RTF operation (see Section V.D.). The user should view a file stored with SMFS as a potentially very long bit string which may be transmitted to SMFS in any number of variable-length _segments_, and is retrievable in any number of variable-length segments, with the manner of segmentation chosen during retrieval independent of that selected during the updating process.

The user may optionally request that SMFS 'remember' the manner in which a file was updated, i.e., along with the data, store sufficient information to reconstruct segment boundaries at retrieval time. Such a file is said to be _formatted_. In retrieving a formatted

file, the user, rather than requesting that SMFS transmit the next 'n' bits of the file as he would do for an unformatted file (see Section V.D.), requests that SMFS transmit the next segment of the file; it is then SMFS's responsibility to supply the length of the segment. Hence, the notion of a _logical record_ is introduced.

Of course, since the user may format the contents of a file in any way he chooses, he can embed record-length information in the data itself. Hence, the user can implement a record structure in a way that's transparent to SMFS. This scheme, however, requires during retrieval that, for each logical record retrieved, the user fetch first the length field and then, using the length as an operand, fetch the data itself. In this kind of arrangement, the retrieval rate is apt to suffer. However, by allowing SMFS knowledge of logical-record boundaries, the feedback loop is effectively shortened (SMFS being closer to the file); hence, the potential exists for an increased retrieval rate.

If the user intends that a file be formatted, he must so specify in every update and every retrieve operation referencing that file. SMFS in no way flags a file to indicate that it is formatted. Hence, if the user invokes the option during retrieval without having done so when the file was stored, results will be erroneous. Furthermore, if an update of a formatted file is terminated before the bit count for the operation is exhausted (i.e., because the user closed the connection), retrieval results will again be erroneous.

V.C. Replace File (RPF)

The replace-file (RPF) operation is identical to UDF, except that the new file segment, rather than being concatenated to the existing file, _replaces_ the entire contents of the file. The previous contents of the file are lost, and the new segment becomes the only segment in the file.

RPF may be used to rewrite an existing file. If the rewritten file is to contain just a single segment, that segment may be transmitted to SMFS in an RPF operation. Otherwise, the first segment of the new file must be transmitted in an RPF operation, and all succeeding segments in UDF operations. Alternately, a dummy (bit count of zero) RPF operation may be inserted before the first real segment is transmitted; all segments of the file may then be transmitted in UDF operations.

V.D. Retrieve File (RTF)

The operation which retrieves all or part of a file's contents is called file retrieval (RTF). The user specifies the filename of the file to be retrieved, the access password if required, and the amount of data in bits to be fetched from the file. SMFS locates the file on secondary storage, checks the password for validity (if appropriate), and copies the bit count and the requested file segment into the output stream. SMFS considers the retrieval complete when either the requested number of bits have been placed in the output stream, or when the contents of the file are exhausted. In this latter case, SMFS closes the connection to signal end-of-data to the user.

Successive RTF operations referencing the same file cause successive segments of the file to be transmitted, provided that the operations are juxtaposed in the input stream (however, NOP's may be interspersed anywhere in the input stream). When a series of RTF operations referencing a particular file is broken by an operation referencing another file, or by a different type of operation involving the same file, the next RTF operation designating the original file will cause the _first_ segment of that file to be transmitted. The manner in which the user segments a file for a series of retrieve operations need bear no relationship to the segmentation scheme employed when the file was updated, nor to that employed in previous retrievals.

If the user elected to have his file formatted by SMFS, he should re-invoke the option in the RTF operation, in which case SMFS will supply the length of the segment, and place both it and the segment itself into the output stream.

V.E. Space File (SPF)

Files stored with SMFS are sequential in organization. That is the n+1th segment of the file cannot be retrieved without first processing the nth segment. The user may, however, upon occasion, wish to retrieve only selected segments of a file. This he could do, effectively, by retrieving each segment of the file and flushing those with which he was currently unconcerned. To avoid needless Network traffic, SMFS provides a mechanism for flushing file segments locally. The operation is called _spacing_ a file (SPF). It is identical to RTF with the exception that transmission of data (but not bit count) is suppressed. SPF operations may be freely inserted anywhere within a series of RTF operations designating a particular file, with the desired results.

V.F. Delete File (DLF)

A file may be deleted at any time after allocation. The user specifies the filename of the file to be deleted and the modification password if required. SMFS locates the file on secondary storage, checks the password for validity (if appropriate), and, if the password is correct, deletes the file. The filename is made available for reassignment, and the secondary storage allocated to the file is reclaimed by SMFS. The contents (if any) of the file are lost.

V.G. Rename File (RNF)

A file stored with SMFS may be renamed at any time after allocation. The user specifies the current filename of the file to be renamed, the modification password if any, and the proposed new filename. SMFS locates the file on secondary storage, checks the password for validity (if appropriate), and assures that the proposed new filename is not already assigned to another file. If these requirements are met, the file is renamed, and all subsequent references to the file must be by the newly-assigned filename.

RNF provides a means for protecting a file that must be rewritten in its entirety against failures in the Net, or in the sending or receiving host. The strategy is as follows. Allocate a new file, assigning it some temporary name. Transmit the revised file contents in one more UDF and/or RPF operations. Then delete the original file and, using RNF, replace the newly-created file’s temporary filename with that of the original file.

V.H. File no Operation (FNO)

FNO is a dummy operation which is provided for use in terminating a series of RTF operations. Should the user desire to retrieve the contents of a file twice in succession, he may do so with a series of RTF/SPF operations, followed by a FNO followed by a second series of RTF/SPF operations. Each RTF/SPF operation in the first series will retrieve/flush the next segment of the file. The first operation of the second string, since it _is_ the first of a string, will, as explained in Section V.D., retrieve/flush the _first_ segment of the file. The remaining operations in the second string will, of course retrieve/flush the 2nd, 3rd, etc., segments of the file. Hence, the contents of the file are transmitted twice. FNO, when it terminates such a string of operations, effectively repositions the user to the first segment of the file.

FNO may appear anywhere within the input stream.

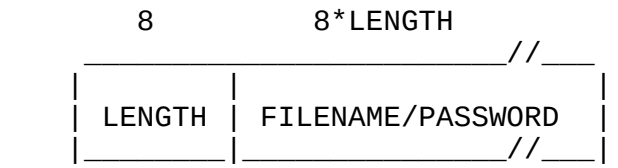
V.I. No Operation (NOP)

This operation is provided _solely_ to aid the user in formatting the input stream, and is discarded without further processing whenever it is encountered. In particular, a NOP embedded in a series of RTF operations does not terminate the string as FNO does.

VI. Input Stream Format

The input stream shall consist of a contiguous string of commands to SMFS. A command type is defined for each of the primitive file operations of Section V. Each command has the following general format:

where the lengths of fixed-length fields have been indicated in bits. Each of the fields 'FILENAME','ACCESS PASSWORD', 'MODIFICATION PASSWORD', and 'NEW FILENAME' is further divided into the following subfields:



where the 'LENGTH' subfield contains the length in 8-bit characters of the 'FILENAME/PASSWORD' subfield.

This is the _general_ format for all SMFS commands. No one command type requires all of the fields specified above. A particular subset of these fields is defined for each type of command, and only those fields should appear. The defined fields for each command type are indicated in Figure 3.

Furthermore, not all of the fields which are defined for a particular command type need always appear _explicitly_. The user should envision that SMFS maintains filename, password, and bit-count accumulators. Every time a filename (or new filename),

access/modification password, or bit count appears explicitly in the input stream, it is saved in the appropriate accumulator (a null password -- designated by setting Bits 0,3 or Bits 8,11 to zero (Figure 4) -- should be thought of as appearing explicitly). The user may cause a defined field to _default_ to the current contents of the appropriate accumulator by turning on the appropriate bin in the flags field (see Figure 4). When a field defaults in this manner, that field is said to appear _implicitly_ in the command.

NOP	0	No operation.
FNO	1	File no operation.
ALF	2	Allocate file.
UDF	3	Update File.
RPF	4	Replace File.
RTF	5	Retrieve File.
SPF	6	Space File.
DLF	7	Delete File.
RNF	8	Rename File.

Figure 2
Command Op codes

The three accumulators are initially empty and hence an attempt to default a field in the first command in the input stream illicits an error indication. A field of the appropriate type must appear once explicitly in the input stream before the corresponding accumulator is considered defined. Furthermore, whenever SMFS detects an invalid filename or password (i.e., improper length or deviation from the character set) in the input stream, the appropriate accumulator is left empty again.

SMFS allows operations on several files to be interleaved in the input stream by including in its command formats provision for explicitly specifying filename and password information in each command. When many operations involving the same file appear sequentially in the input stream, the user need only let the appropriate fields default in all but the first command, avoiding re-transmission of what would otherwise be redundant parameters.

	OP CODE	FLAGS	FILENAME	ACCESS PASSWORD	MODIFICATION PASSWORD	NEW FILENAME	BIT COUNT	DATA
ALF	X	X	X	X	X		X	
UDF	X	X	X		X		X	X
RPF	X	X	X		X		X	X
RTF	X	X	X	X			X	
SPF	X	X	X	X			X	
DLF	X	X	X		X			
RNF	X	X	X		X	X		
FNO	X							
NOP	X							

Figure 3

Defined Command Fields

Note: Command fields marked with an ’X’ are defined.

0	ACCESS PASSWORD DEFAULTS	The access password for this operation defaults to the access or modification password which appeared explicitly most recently in the input stream; hence, it does not appear explicitly in the current command.
1	BIT COUNT DEFAULTS	The bit count for this operation defaults to that which appeared explicitly most recently in the input stream; hence it does not appear explicitly in the current command.
2	FILENAME DEFAULTS	The filename for this operation defaults to the filename or new filename which appeared explicitly most recently in the input stream; hence it does not appear explicitly in the current command.
3	ACCESS PASSWORD APPEARS EXPLICITLY	The access password for this operation appears explicitly in the current command. (Bits 0, 3 = 0 indicates that no access password was/is-to-be defined for the file.)
4	ECHO OP CODE AND FILENAME	SMFS shall echo the op code and filename (whether it appears explicitly or not) by copying them into the output stream ahead of any other response to the current command.
5-7	undefined	Not examined; should be zeros.
8	MODIFICATION PASSWORD DEFAULTS	Same as Bit 0, but applied to the modification password, rather than the access password.

Figure 4

Definition of Command FLAGS Bits

9	FILE FORMATTED	FOR UDF/RTF: this segment is part of a formatted file; hence SMFS should record the bit count. For RTF/SPF: the referenced file is formatted; hence the bit count does not appear explicitly in the current command
10	NEW FILENAME DEFAULTS	same as Bit 2, but applied to the new filename, rather than the filename.
11	MODIFICATION PASSWORD APPEARS EXPLICITLY	Same as Bit 3, but applied to the modification password, rather than the access password.
12-15	undefined	Not examined; should be zeros.

Figure 4(continued)

Definition of Command FLAGS Bits

Note: The sixteen bits of FLAGS are numbered 0-15 from left to right.

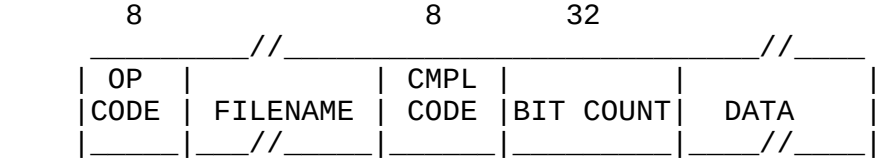
When a series of RTF/SPF operations referencing the same file are juxtaposed in the input stream (as discussed in Section V.D.), they cause successive segments of the file to be transmitted _only_ if both filename and access password default (Bits 0,2 = 1) (a null password is also acceptable) in those operations following the first in the series. If the user specifies either parameter explicitly in a command in the series -- even if the explicitly stated value is the same as what would have been the default value -- SMFS considers the series terminated, as if a FNO had been encountered, and hence the command in question returns, or flushes, the first segment of the file. Allowing both filename and password to default has the added effect, in both RTF/SPF and UDF series, of decreasing the processing time required by SMFS to execute the operations which comprise the series. Under such circumstances, SMFS executes such initial functions as file location and password verification only once at the beginning of the series, rather than for each operation. Hence, a potential for increased transmission rates exists. Furthermore, in such a series of UDF/RPF operations, SMFS is able to conserve secondary storage by concatenating file segments before they are written out.

Whenever SMFS aborts the processing of a command in the input stream (e.g., the filename is invalid, an incorrect password is supplied, etc), SMFS flushes the entire command. Suppose, for example, that the file specified in a UDF operation does not exist (i.e., has not been allocated). If the data field for the operation is very long, SMFS may well detect the non-existence of the file before the data field has been transmitted by the user. In such cases, SMFS will accept and flush whatever remains of the aborted command (in this case, including the very long data field) until it reaches the point in the input stream at which it expects to find the next command, which it will process normally. SMFS will, however, notify the user that the command was aborted by placing an appropriate indicator in the output stream, and it will do this as soon as it detects the error (and hence, in this case, before the erroneous command has been flushed from the input stream). Hence, the user has the option of aborting the process by closing the connection.

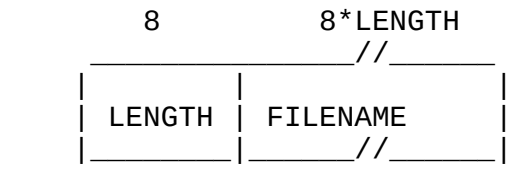
SMFS considers a command with an invalid op code as an especially severe error, since it has no way of locating the start of the next command. Accordingly, it places a special character (X'FF') in the output stream, follows it with the invalid op code, and then closes its connections to the user.

VII. Output Stream Format

SMFS will respond to each command it extracts from the input stream -- every command except FNO and NOP -- by placing a command response in the output stream. Command responses have the following general format:



where the lengths of fixed-length fields have been indicated in bits. The field 'FILENAME' is further divided into the following subfields:



where the 'LENGTH' subfield contains the length in 8-bit characters of the 'FILENAME' subfield.

This is the general format for SMFS command responses. For responses to particular commands, not all fields may be present. A particular subset of these fields is defined for each type of command response; no other fields will appear. The defined fields for each command response type are indicated in Figure 5.

The fields 'OP CODE' and 'FILENAME' are the op code and filename extracted by SMFS from the input stream and are echoed by SMFS in the output stream. The filename is always echoed explicitly, even if it appeared implicitly in the input stream. 'OP CODE' and 'FILENAME' are suppressed and hence do not appear in the command response it Bit 4 of the 'FLAGS' field of the corresponding command is set to 0.

'CMPL CODE' contains an indication of the outcome of the operation. If the operation was completed successfully, 'CMPL CODE' contains a value equal to the op code of the command executed. Hence, if echoing of 'OP CODE' and 'FILENAME' is not suppressed, the operation was successful if and only if 'OP CODE' and 'CMPL CODE' are identical. If the operation as unsuccessful, 'CMPL CODE' contains an indication of the error encountered by SMFS in processing the command. Completion codes are summarized in Figure 6.

	OP CODE	FILE NAME	COMPLETION CODE	BIT COUNT	DATA
NOP					
FNO					
ALF	X	X	X		
UDF	X	X	X		
RPF	X	X	X		
RTF	X	X	X	X	X
SPF	X	X	X	X	
DLF	X	X	X		
RNF	X	X	X		

Figure 5
Defined Command Response Fields

Note: Command response fields marked with an 'X' are defined.

An invalid op code in the input stream constitutes a special type of error. SMFS's response is as follows. A special command response is constructed. It consists of the value X'FF' in an eight-bit field, followed by the erroneous op code, also in an eight-bit field. The command response is placed in the output stream and connections to the user are closed.

2	ALLOCATION SUCCESSFUL	The file was successfully allocated.
3	UPDATE SUCCESSFUL	The file was successfully updated.
4	REPLACE SUCCESSFUL	The file was successfully replaced.
5	RETRIEVE SUCCESSFUL	The file segment was successfully retrieved.
6	SPACE SUCCESSFUL	The file segment was successfully flushed.
7	DELETION SUCCESSFUL	The file was successfully deleted.
8	RENAME SUCCESSFUL	The file was successfully renamed.
20	NO DEFAULT FILENAME	The user attempted to default the filename (or new filename), and the filename accumulator was empty.
21	ZERO-LENGTH FILENAME	The length of the filename (or new filename) was specified as zero.
22	FILENAME TOO LONG	The length of the filename (or new filename) exceeded 36 characters.
23	INVALID FILENAME	The filename (or new filename) contained character(s) that do not appear in the character set.
24	NO DEFAULT PASSWORD	The user attempted to default either the access or modification password, and the password accumulator was empty.
25	ZERO-LENGTH PASSWORD	The length of either the access of modification password was specified as zero.

Figure 6
Completion Codes

26	PASSWORD TOO LONG	The length of either the access or modification password exceeded 36 characters.
27	NO DEFAULT BIT COUNT	The user attempted to default the bit count, and the bit-count accumulator was empty.
28	INVALID PASSWORD	Either the access or modification password contained character(s) that do not appear in the character set.
29	DUPLICATE FILENAME	Either the filename (in an ALF operation) or new filename (in a RNF operation) is already assigned to another file.
30	INSUFFICIENT SPACE	(In an ALF operation) The requested amount of secondary storage is unavailable.
31	ALLOCATION I/O ERROR	(In an ALF operation) An irrecoverable I/O error was encountered by SMFS while attempting to allocate the file.
32	FILE NOT FOUND	The referenced file does not exist.
33	SEARCH I/O ERROR	An irrecoverable I/O error was encountered by SMFS while attempting to locate the referenced file.
34	FILE FULL	(In a UDF/RPF operation) The secondary storage allocated to the file has been exhausted.
35	INCORRECT PASSWORD	The access or modification password supplied by the user does not match that declared when the file was allocated.
36	FILE SIZE TOO SMALL	(In an ALF operation) The bit count specified is less than the minimum file size accepted by SMFS.

Figure 6 (continued)
Completion Codes

37	FILE SIZE TOO BIG	(In an ALF operation) The bit count specified exceeded the maximum file size accepted by SMFS.
38	WRITE I/O ERROR	An irrecoverable I/O error as encountered by SMFS. (In an ALF operation) SMFS was attempting to record password information, or (in a UDF/RPF operation) SMFS as attempting to add data to the file.
39	READ I/O ERROR	An irrecoverable I/O error was encountered by SMFS attempting to retrieve either password information or data.
40	RENAME I/O ERROR	An irrecoverable I/O error was encountered by SMFS while attempting to rename the file.
41	DELETE I/O ERROR	(In a DLF operation) An irrecoverable I/O error was encountered by SMFS while attempting to delete the file.
42	END-OF-DATA	(In a RTF/SPR operation) The end of the file was reached before the requested segment had been transmitted/flushed.

Figure 6 (continued)
Completion Codes