ABS Tech Note: SNA•ps08 3270 API (9/94)

Article Created: 26 April 1993
Article Reviewed/Updated: 27 September 1994

TOPIC ---

This technical note discusses Host-based File Transfer support as implemented in the Apple 3270 API 2.0.

DISCUSSION ---

Introduction

The Apple 3270 API 2.0 supports file-transfers that allow Macintosh applications to store and retrieve data from files located on an IBM host computer. This support allows an application to perform file transfer without regard to the underlying host connection or subsytem.

This Technical Note describes the use of these file transfer requests for SNA•ps Access/3270, Apple's implementation of the Apple 3270 API 2.0. The transfer of files takes place via the 3270 Data Stream and a family of host-based File Transfer programs collectively known as IND$FILE.

SNA•ps Access/3270

The Apple 3270 API 2.0 is a defined programming language interface between the 3270 data stream and Macintosh client applications. SNA•ps Access/3270 is Apple's implementation of the 3270 API, allowing connectivity to IBM hosts via the SNA•ps line of gateway products.

Host-Based File Transfer

The File Transfer between an IBM host and a Macintosh using SNA•ps Access/3270 is controlled by the host application. This application (IND$FILE) is initiated by SNA•ps Access/3270 sending a command to the host to start the file transfer. SNA•ps Access/3270 sends this command in response to a StartSend/StartReceive request from the Macintosh application. After the file transfer begins, IND$FILE issues requests to SNA•ps Access/3270, and waits for the response. SNA•ps Access/3270 may issue a message to cancel a file transfer operation, but it is up to IND$FILE as to whether/when the cancel takes effect.

Host Environments and Transport mechanisms

IND$FILE operates in three (3) major Host environments—TSO, CICS/VS, and VM/CMS. Each of these environments has a specific implementation of IND$FILE installed to support Host-based file transfer.

There are two mechanisms used to implement the transfer of file information using the 3270 Data Stream.

- PS-based File Transfer (CUT): The Presentation Space (PS) is used to transfer the file data to/from the Host. The PS has several specific formats which differ depending upon the direction of the file transfer. These are:
  - UPLOAD format - used when sending a file to the Host
  - DOWNLOAD format - used when retrieving a file from the Host
  - CONTROL format - used to pass control information to the Macintosh from the Host.

These formats are defined in the section "PS Screen Formats" as 'C' language structures.

- SF File Transfer (DFT): This mechanism uses Structured Fields (SF) to transfer the file to/from the host. The SFs used are part of the Distributed Data Management (DDM) architecture, and are known colloquially as D0(D-ZERO) SFs. Only a small set of the D0 SFs are used; this subset is defined in the section "D0 Structured Fields Subset".

Since there are three (3) possible host environments, and two (2) possible transport mechanisms for each host environment, there are a total of six (6) possible ways to do file transfer.

SNA•ps Access/3270 Requests for File Transfer

SNA•ps Access/3270 simplifies Host-based file transfer by providing a single set of file transfer requests which are independent of the transport protocol used. These requests are:

- StartSend () - allows an application to initiate file transfer to the Host.

- StartReceive () - allows an application to initiate file transfer from the Host.

- DoSend () - allows an application to send data to the Host, signal the end of data to be transferred (i.e., End of File), and abort the file transfer.

- DoReceive () - allows an application to receive data from the Host, receive messages from the Host (i.e. End of File), and indicate that the application wishes to abort the file transfer.

SNA•ps Access/3270 File Transfer Requests Validation

SNA•ps Access/3270 will validate the parameters to the File Transfer

requests and return an error code without performing the operation, if there is an error. If parameter validation fails, the following error codes may be returned:

StartSend

Return Code	Reason
kAPIFTTypeParm	TransferType not equal to kINDFILE.
kAPIFTFileNameParm	hostFileName not specified.
kAPIFTHostEnvirParm	verifyName is true and hostEnviron is not specified
kAPIFTFileTypeParm	fileType is not kTestFile or kBinaryFile
kAPIFTBadFEParam	fileExists is not kNewHostFile, kAppendHostFile, or kReplaceHostFile
kAPIFTRTParam	recordType is not kFixed, kVariable, or kUndefined
kAPIFTNotAvailable	There is not a session with the host.
kAPINotDisplaySession	The session is not a display session.
kAPIFTRecLenParm	The recordLength specified is zero(VM/CMS).
kAPIFTInProgress	There is already a File Transfer in progress.
kCapyResErr	An error occured trying to load a required resource.
kAPIInputInhibit	The 3270 keyboard is locked.
kAPIKeyCodeErr	There is an invalid character in the command line built by SNA•ps Access/3270. (If there is not a translation value for a character in the current character set, this error would be returned.)

StartReceive

Return Code	Reason
kAPIFTTypeParm	fileTransferType not equal kINDFILE.
kAPIFTFileNameParm	hostFileName not specified.
kAPIFTHostEnvirParm	verifyName is true and hostEnviron is not specified. (Check the list below for errors that may be returned when the access method verifies the host file name.)
kAPIFTFileTypeParm	fileType is not kTestFile or kBinaryFile.
kAPIFTNotAvailable	There is not a session with the host.
kAPINotDisplaySession	The session is not a display session.
kAPIFTInProgress	There is already a File Transfer in progress.
kCapyResErr	An error occurred trying to load a required resource.
kAPIInputInhibit	The 3270 keyboard is locked.
kAPIKeyCodeErr	There is an invalid character in the command line built by SNA•ps Access/3270. (If there is not a translation value for a character in the current character set, this error would be returned.)

DoSend

Return Code	Reason
kAPIFTNotAvailable	There is not a session with the host.
kAPINotDisplaySession	The session is not a display session.

kAPIFTNotInProgress	There is not a File Transfer in progress.
kAPIFTRespParm	The appResponse field is not set to kContinueFT or kCancelFT.
kAPISendBufParm	The sendBufPtr is zero, and the sendBufLen is not zero.
kAPIFTNoSendInRcv	The File Transfer in progress is a receive.
kAPIFTCancelBad	The appResponse was kCancelFT, and an error occurred while trying to issue the cancel.
kAPIFTNotRdyToSend	There wasn't a kDoSendEvent generated. If the File Transfer is being made via the PS, then errors associated with CopyToPS may also be encountered. (kAPIScrnSizeChng, kAPIInputInhibit, kAPIByteCountParm, kAPISrcBufParm, kAPIKeyCodeErr, kAPIEndOfPS, kAPIWriteProtFldErr, or kAPIWriteAttrErr.)

DoReceive

Return Code	Reason
kAPIFTNotAvailable	There is not a session with the host.
kAPINotDisplaySession	The session is not a display session.
kAPIFTRespParm	The appResponse field is not set to kContinueFT or kCancelFT.
kAPIReceiveBufParm	The receiveBufPtr is zero, and the numBytesToReceive is not equal to zero, and the appResponse is kContinueFT.
kAPIFTNotInProgress	There is not a File Transfer in progress.
kAPIFTNoDataToRecv	There is no data to be received.

For both StartSend and StartReceive requests you can specify that the Access Method should verify that the host name being passed is a valid name for the specified system. To do this, in the StartSendBlk set verifyName to True and set hostEnviron to the value matching the host environment being used. Name validation may return the following errors:

TSO

Return Code	Reason
kAPIFTBadFirst	Error with resource file or bad first character of hostFileName.
kAPIFTTSOLong	Data Set Name is too long.
kAPIFTQualLon	Qualifier too long.
kAPIFTNoQuoteMemb	Quote around member, but no parenthesis.
kAPIFTBadChar	Invalid character found in the name.
kAPIFTMemberChar	A period or quote was found in a member name.
kAPIFTMisParen	Matching parenthesis is missing.
kAPIFTTSOSpace	A space character was found in the hostFileName.
kAPIFTMemberLong	Member name is too long.
kAPIFTMemberBad	Missing quote or parenthesis.
kAPIFTMisplacedQte	Missing/misplaced quote.
kAPIFTMismatchedQte	Quotes not placed correctly.

VM/CMS

Return Code	Reason
kAPIFTBadFirst	Error with resource file or bad first character of hostFileName.
kAPIFTCMSDot	A period appeared in the hostFileName.
kAPIFTBadChar	An invalid character appeared in hostFileName.
kAPIFTNoType	No "Type" appeared in the hostFileName.
kAPIFTCMSLong	The "Name" field is too long.
kAPIFTCMSTLong	The "type" field is too long.
kAPIFTCMSMLong	The "mode(format)" field is too long.
kAPIFTCMSExtra	An extra field appeared in hostFileName.

CICS/VS

Return Code	Reason
kAPIFTBadFirst	Error with resource file or bad first character of hostFileName.
kAPIFTCICSTLong	hostFileName is too long.
kAPIFTCICSDot	A period appeared in the hostFileName.
kAPIFTCICSSpace	A Space appeared in the hostFileName.
kAPIFTBadChar	An invalid character appeard in hostFileName.

SNA•ps Access/3270 File Transfer Requests Usage

The SNA•ps Access/3270 File Transfer Requests may be used to send/receive a file to/from the Host. It is assumed in the following paragraphs that the Macintosh application/User has logged onto one of the supported host environments (TSO, CICS, VM/CMS).

To Send a file to the Host:

- Issue the StartSend request. If an error occurs, correct the error, and resubmit the request.
- Issue PollSessions.
- If the event returned was a kDoSendEvent, send one block of data to the host using DoSend. Note that SNA•ps Access/3270 will automatically segment this block if it exceeds the transport mechanism's maximum size.
- If the event returned was a kDoReceiveEvent, the file transfer is not successful, and the DoReceive request will likely return an error message that was received from the Host. Issue a DoReceive request.
- If the event returned was a kPSUpdateEvent, the host application has written a message into the PS, which may be displayed to the user. For example, this will occur in TSO if you exceed the disk space allocated for the file. Issue GetUpdate() to allow the file transfer to continue. OIA updates, which also occur in this manner, will be a fairly constant event during the course of the file transfer.
- Repeat the above sequence (starting at PollSessions) until the end of the file, then issue the DoSend request with a zero-length for the data and a null send buffer pointer.
- Note that at any time the host application may cancel the file transfer. Be prepared to receive kDoReceiveEvents from the PollSessions call.
- Note that if a "Cancel" is issued, the host application may or may not return a message (as a kDoReceiveEvent) indicating the file transfer was canceled. You must receive these events or subsequent StartSend and

StartReceive requests will fail with the error kAPIFTInProgress.

To receive a file from the Host:

- Issue the StartReceive. If an error occurs, correct the error and resubmit the request.
- Issue PollSessions.
- If the event returned was a kDoReceiveEvent, issue a DoReceive request. This request may contain an error message from the host (file not found, for example) or the first block of data.
- If the event returned was a kPSUpdateEvent, the host application has written a message into the PS, which may be displayed to the user. For example, this will occur in TSO if you exceed the disk space allocated for the file. Issue GetUpdate() to allow the file transfer to continue. OIA updates, which also occur in this manner, will be a fairly constant event during the course of the file transfer.
- Repeat the above sequence (starting at PollSessions) processing the kDoReceiveEvents until the host application sends a block with the host Reply field set to kMsgContent. The host normally sends only one (1) message after the end of the file. If an error occurs during the file transfer, the host application may send an error message.
- If a DoReceive request is issued specifying Cancel, SNA•ps Access/3270 will return kAPINoErr after sending the appropriate cancel command to the host application. The host application may then send a kClose event and a message, or it may simply send an error message. You must receive these events or subsequent StartSend and StartReceive requests will fail with the error kAPIFTInProgress.

See the section "Sample File Transfer Flowchart for an Apple 3270 API 2.0 User Program" for a more complete description.

File Transfer Request Formats

The next two sections detail the format of the requests/responses which flow between SNA•ps Access/3270 and the host application. These sections are provided for your use while attempting to decode a line trace of a file transfer request which may be failing.

PS Screen Formats

CUT-based file transfer uses three types of screens to communicate data from the host to the Mac. Each screen may be distinguished by the first character.

```
// character          Meaning
//   'A'  -    This is a Download format screen
//   'B'  -    This is an Upload format Screen
//   'C'  -    This is a Control format screen
#define kMAX_PSFT_DOWN   1909   // Max Data from Host
#define kMAX_PSFT_UP     1912   // Max Data to Host
// Types of control frames
#define   FT_Ack         'a'    // Acknowledgement
#define   FT_Info        'i'    // Information (Message)
#define   FT_Mesg        'm'
```

```
     #define   FT_Quit        'q'    // Quit/Abort/Quiesce
```

Control Screen Format

```
Definition of the Control Format of the PS:
typedef struct psft_ctlscreen{
     unsigned char  psftc_control; // Screen Format indicator
     unsigned char  psftc_attr1;   // Unprotected Attr.
     unsigned char  psftc_code1;   // Code 'a'
     unsigned char  psftc_code2;   // Code 'a', 'q', 'i', 'm'
     unsigned char  psftc_code3;   // Code 'R', ' ', '\0'
} PSFT_CTLSCRN;
```

UPLOAD Format

```
Definition of the Upload Format of the PS
typedef struct psft_upscreen{
     unsigned char  psftu_control;           // Screen format indicator
     unsigned char  psftu_attr1;             // Unprotected attr.
     unsigned char  psftu_code;              // Data Code 'A'
     unsigned char  psftu_sequence;          // Block Sequence
     unsigned char  psftu_chksum;            // Checksum of screen
     unsigned char  psftu_len1;              // Length of data MSB
     unsigned char  psftu_len2;              // Length of screen LSB
     unsigned char  psftu_data[kMAX_PSFT_UP];   // Data being uploaded
     unsigned char  psftu_attr2;             // Protected Attr.
} PSFT_UPSCRN;
```

DOWNLOAD Format

```
Definition of the Download Format of the PS :
typedef struct psft_dwnscreen{
     unsigned char  psftd_control;           // Screen format indicator
     unsigned char  psftd_sequence;          // Block Sequence
     unsigned char  psftd_chksum;            // Checksum of screen
     unsigned char  psftd_len1;              // Length of data MSB
     unsigned char  psftd_len2;              // Length of screen LSB
     unsigned char  psftd_data[kMAX_PSFT_DOWN];   // Data being downloaded
     unsigned char  psftd_attr1;             // Unprotected Attr.
     unsigned char  psftd_resp[4];           // Area for PC to respond in
     unsigned char  psftd_attr2;             // Protected attr.
} PSFT_DWNSCRN;
```

D0 Structured Fields Subset

```
DFT-based File Transfer uses a subset of the D0 structured fields defined
as part of DDM.  These D0 structured fields flow as a Write Structured
Field (WSF) from the host (and with inbound AID of 0x88 to the host).   A
structured field has the following format:

  struct SF {
     unsigned short length;        /* Length of SF */
     unsigned char   sf_id1;       /* First/only ID byte */
```

```
    unsigned char   sf_id2;       /* Second byte of ID or */
                                  /* first  byte of SF info */
    unsigned char   sf_data[1]    /* Contents determined by SF type */
  };
```

The following D0 structured fields are used by the DFT-based File Transfer mechanism:

- 0xD000 - Open
  The Open function is used to make a logical connection between the file transfer and a specific file. The following is the format of the Open SF:

```
   00 23                          // Length of this field
   D0 00                          // SF Type (Open)
   12 01 06 01 01 04 03 0a 0a 00  // Fixed Information
   00 00 00 11 01 01 00 50 05 52  // Fixed Information
   03                             // Fixed Information
   F0                             // No Compression
   03 09                          // Fixed Information
   C'FT:DATA' or C'FT:MSG'        // Dummy file name
```

  Only two (2) file names are used in the Open request. 'FT:DATA' is used to generically represent the file to be up/down-loaded. The actual name of the file was communicated on the command line. 'FT:MSG' is used by the host application as the name of the file that messages from the host application are to be stored into. These messages come your application as 'kMsgContent' DoReceive events with SNA•ps Access/3270.

- 0xD041 - Close
  This request terminates the logical connection established between the local file and the remote (host) file. It is not always sent by all implementations of IND$FILE. The format of the Close request is:

```
   00 05     // Length
   D0 41     // SF ID (Close)
   12        // Fixed Information
```

- 0xD045 - Set Cursor
  This request is used by the host application to place the 'cursor' to the beginning of the next block to be sent. It is used for file uploads only. The format of the Set Cursor request is as follows:

```
   00 0F                          // Length
   D0 45                          // SF ID
   11 01 05 00 06 00 09 05 01 03  // Fixed Information
   00                             // Fixed Information
```

  The Set Cursor request is followed immediately by the Get request.

- 0xD046 - Get
  The Get request flows in both directions for file uploads. In the Outbound (from the host) direction, it flows immediately after the Set Cursor request. SNA•ps Access/3270 will send the next block of data from the 'file' using a Get request (which is really a response at this

point). SNA•ps Access/3270 will check an internal queue for data to be sent. If no data is presently queued, a kDoSendEvent will be returned to the next PollSessions request issued which requests kDoSendEvents. The format of Get request is as follows:

```
From host application:
----------------------
00 09           // Length
D0 46           // SF ID (Get)
11 01 04 00 80  // Fixed Information

From SNA•ps Access/3270:
------------------------
XX XX     // Length (Depends on amount of data sent and segment size)
D0 46     // SF ID
NN NN NN NN    // Sequence number of this block
C0        // Fixed Information
80        // Data not compressed
DD DD     // Length of data that follows
DATA      // Data from DoSend command, possibly segmented
```

• 0xD047 - Insert
This command is used by the host during file transfer to the Macintosh. It inserts the next block of data into the file. The block is always inserted at end of file. This always flows as two (2) Insert requests. The first is of fixed size and information. It informs SNA•ps Access/3270 that the insert is to be a sequential insert. The format of the Insert request is:

```
Fixed Insert:
-------------
00 0A           // Length
D0 47           // SF ID
01 05 00 80 00  // Fixed Information

Variable Insert:
----------------
XX XX     // Length
D0 47     // SF ID
C0        // Fixed Information
80        // Data not compressed
61        // Fixed Information
YY YY     // Length of data
DATA      // Data to be stored as next block of file
```

Article Change History:
27 Sep 1994 - Reviewed.

Support Information Services

Keywords: knts

==========

This information is from the Apple Technical Information Library.

19960215 11:05:19.00

Tech Info Library Article Number: 11694