Adobe® PostScript®

PostScript Language File Transmission (PSFT) Specification

Adobe Developer Support

Version 1.0

02 April 1993

Adobe Systems Incorporated

Adobe Developer Technologies
345 Park Avenue
San Jose, CA 95110
http://partners.adobe.com/

PN LPS5011

Contents

List of Figures

List of Tables

PostScript Language File Transmission (PSFT) Specification

1 Introduction

Adobe™ Systems, Incorporated currently licenses PostScript™ language software to printer manufacturers for support of CCITT Group 3 facsimile products. PostScript fax printers allow users to:

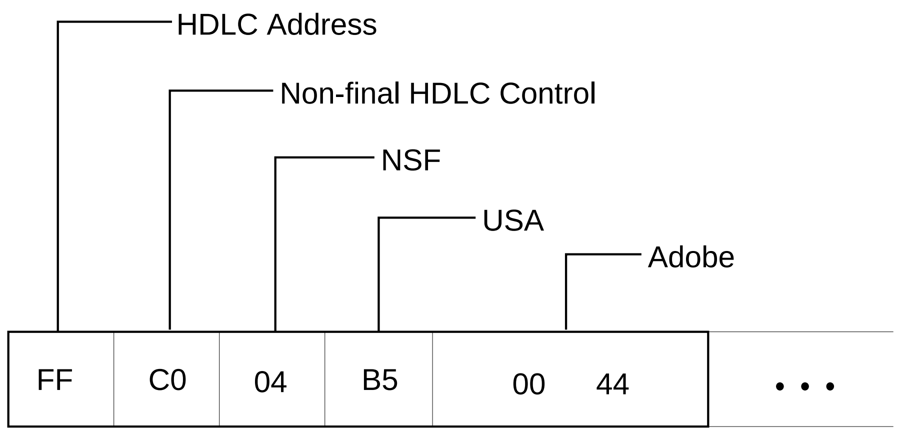
- Send fax documents from within any application on a Macintosh® or PC
- Receive fax documents on plain paper from any fax machine
- Transmit PostScript language files between PostScript fax printers.

This document describes the Adobe extensions to the standard CCITT T.30 protocol that enable PostScript language file transmission (PSFT). It is assumed that the reader is familiar with the material in CCITT Recommendation T.30, 1988 revision (blue book) or later. A copy can be obtained from Global Engineering Documents by calling 714-261-1455.

2 Adobe Non-Standard Facilities/ Non–Standard Setup

A PostScript fax printer incorporates an Adobe Non-Standard Facilities (NSF) HDLC optional frame in every binary coded signal burst that includes a Digital Information Signal (DIS) mandatory frame. Figure 1 shows an example of how the NSF frame begins.

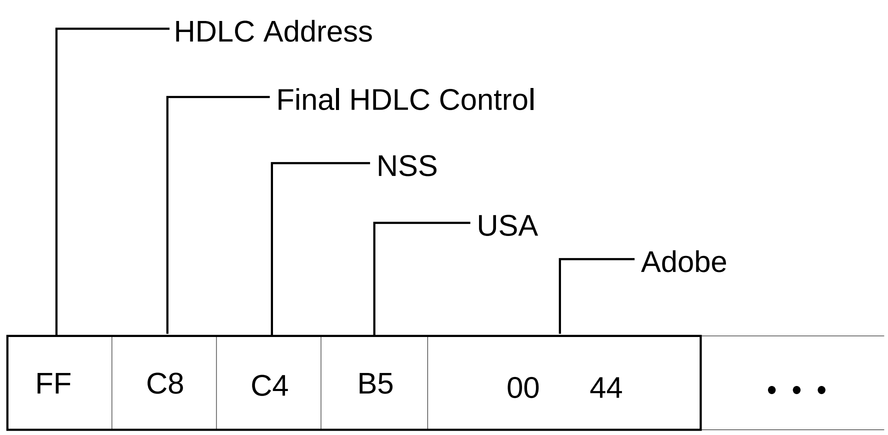
Figure 1 *Adobe NSF Frame Header*



The FF is the standard HDLC address field, the C0 is a non-final control field, 04 encodes NSF, B5 encodes the United States, and 00 44 encodes Adobe Systems Incorporated. This company identifier code was assigned by the chairman of EIA/TIA working group TR-29, who is authorized to assign such identifier codes to United States companies.

Note *This document uses the uniform big-endian sequence. Data octets are generally carried to and from telecommunications channels through UARTs, which historically use little-endian sequence within an octet. If you use a standard UART, you may need to assemble and disassemble octets backwards from the way they are shown in this document, or use an octet-reversal table.*

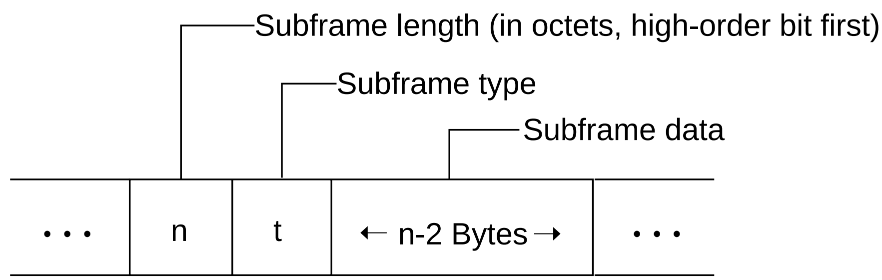
Figure 2 *Adobe NSS Frame Header*



As can be noted in Figure 2, the Adobe Non–Standard Setup (NSS) frames begin similarly to the NSF frames.

The remaining data, if any, of an Adobe NSF/NSS frame is a sequence of subframes. Each subframe is preceded by two octets: a subframe length octet and a subframe type octet. The subframe length octet expresses the length of the subframe in octets, including the length octet itself. Figure 3 shows an example of this.

Figure 3 *Basic Adobe NSF/NSS Subframe Format*



Adobe subframe types are intended to be an expanding set with time, much as the Facsimile Control Fields of T.30 have been. At this point Adobe has defined a small set. If an older implementation receives a newer Adobe subframe type that it does not understand, it uses the subframe length octet to skip over that ignored subframe. Adobe reserves the right to define all subframe types in the range [0x00..0xC0]; subframe types outside that range will not be used in Adobe products, and may be used by Adobe and others on an uncontrolled experimental basis.

The random chance of connecting to a PostScript fax product will be low until PostScript fax products infiltrate the growing installed base of standard fax machines. As a result, the Adobe NSF frame will likely be ignored by a calling standard fax machine. To minimize the duration of the negotiating protocol, the callee restricts itself to transmitting an NSF frame with no subframes, until the callee hears an Adobe NSS frame. This process necessitates an additional NSS/ NSF cycle if the caller wants to attempt transmission of PostScript language files.

2.1 ADOBE_INFO (subframe type 3)

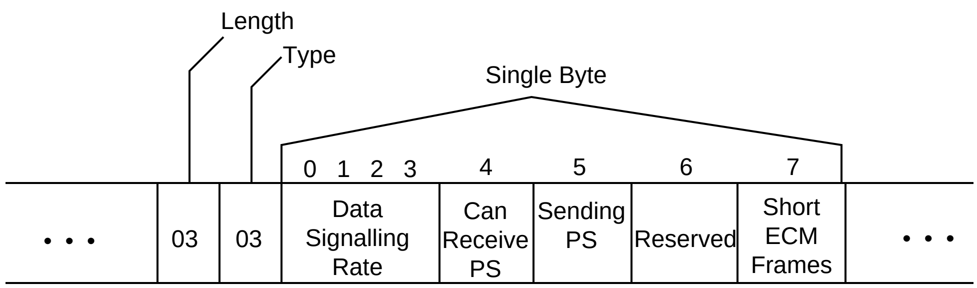
The **ADOBE_INFO** subframe is a bit sequence similar to that used in the T.30 DIS/DCS frame. The bits are numbered beginning with 0, the first bit transmitted. If an **ADOBE_INFO** subframe is transmitted, all bits in octets beyond those actually transmitted are defined as 0. The following descriptions refer to Figure 4.

Bits [0..3] are identical to T.30 DIS/DCS frame bits [11..14]. In an NSF frame they define (redundantly with the following DIS frame) the modem types available for receiving data. In an NSS frame, they define the data signalling rate to be used.

Bit 4 is 1 if this device is capable of receiving PostScript Level 2 language files.

Bit 5 is 1 in an NSS frame if a PostScript language file will be transmitted following a successful Training Check and Confirmation Cycle (TCF-CFR).

Figure 4 *Short ADOBE_INFO NSF/NSS Subframe*



Bit 7 is 1 in an NSF frame if the Error Correcting Mode (ECM) transport layer is permitted to transmit PostScript language data with HDLC frames containing 64 octets of data (instead of the usual 256). Bit 7 is 1 in an NSS frame if the ECM transport layer will use HDLC frames containing 64-octets of data (rather than 256); this is permitted only if a previous NSF frame from the correspondent unit allowed it. Under noisy line conditions, using shorter frames sometimes results in shorter overall transmission times.

The receiver uses bit 8 in the NSF frame to communicate fall back capability to the sender for T.4 raster. If the receiver has set bit 8 to 1, the receiver can accept either PostScript language data or fall back to receive T.4 raster in the same phone call after seeing an Adobe command from the sender. The sender may choose to fall back to raster, for example, if it does not know the password of a protected receiver or it does not think the receiver has the memory resource for the PostScript language file. The sender can only fall back and send raster in the same phone call, if bit 8 is 1. If the sender wants to fall back to raster and sees bit 8 is 0, any attempt to fall back to raster will fail, and the sender should place a second call to send raster.

Adobe reserves the right to define all **ADOBE_INFO** subframe bits beyond 8.

2.2 ADOBE_RCV_RESERVATION (subframe type 4)

The **ADOBE_RCV_RESERVATION** subframe is an optional 32-bit non-negative integer field in the NSF frame (high-order bit transmitted first) that is a hint of the number of octets of buffer space in the receiver that are available for holding a PostScript language file. A 0 in this field means no information is being given. The sender can use a non-zero value in this field to help decide whether sending a PostScript language job would likely succeed, or whether normal Group 3 compressed raster would have a better chance.

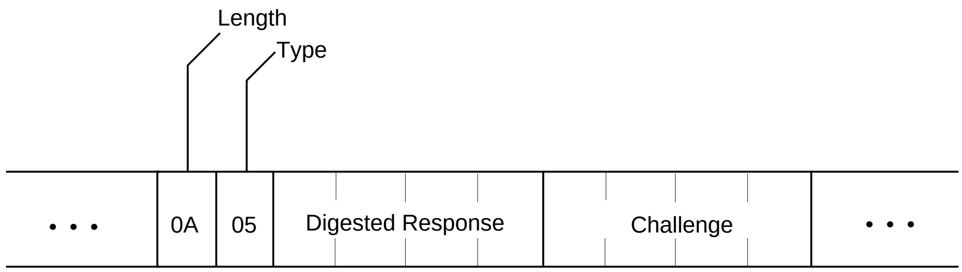
2.3 ADOBE_SECURITY (subframe type 5)

The PostScript language is a general-purpose programming language, and can thus encode *trojan horses*: programs that appear to do something straightforward, but whose real purpose is hidden, and possibly damaging. Adobe has designed a mechanism whereby a receiver has the option to accept a PostScript language job only when there is strong evidence that the sender and receiver are in possession of a shared secret key. At the same time, the mechanism does not help an agent who overhears a successful PostScript language file transfer to transmit his own subsequent PostScript language file to the same receiver.

The mechanism is based on the MD5 function, which was devised and released to the public domain by RSA Data Security, Inc., of Menlo Park, California. MD5 is a one-way hash function, that maps any sequence of octets into a 128-bit "digest." MD5 is designed so that finding any sequence of octets that maps into a particular digest should be very difficult. MD5_32 is simply the most significant 32 bits of the 128-bit MD5 result.

The NSF frame contains a sequence of two 32-bit non-negative integers (transmitted high-order bits first). The second is a randomly-chosen challenge, for example derived from the system clock, thermal noise, or whatever. The first of these integers (digested response) is the result of mixing that challenge with the 32-bit digested secret key (using bitwise exclusive-OR) and iterating the MD5_32 function twice on the result. Refer to Figures 5 and 6.

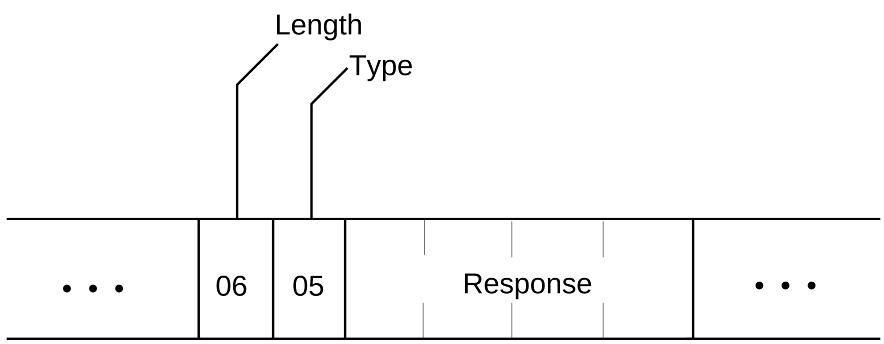
Figure 5 *ADOBE__SECURITY NSF Subframe*



The 32-bit digested secret key is simply the result of applying MD5_32 to an arbitrary-length ASCII secret key, except that if the ASCII secret key is of length 0, the 32-bit digested secret key is defined as 0.

The NSS reply frame contains a single 32-bit non-negative integer response that is the result of mixing the most-recently-heard challenge with the 32-bit digested secret key and then applying the MD5_32 function.

Figure 6 *ADOBE_SECURITY NSS Subframe*



The method is designed so that the sender can know as soon as he hears the challenge whether his secret key can pass the challenge. Thus, the sender does not have to wait through a cycle of command messages following a security failure before opting to transmit normal T.4 compressed raster (depending on the state of **ADOBE_INFO**).

Note *See Appendix B: Security Flow Chart.*

3 Example Sequences

The following sections describe some captured traces of successful and unsuccessful transmissions of PostScript language data. The times shown are offsets in milliseconds from an arbitrary time, so only differences between these numbers are meaningful.

3.1 Example One

The trace of Table 1, which is from the point of view of the caller/sender, shows the simplest possible transmission of PostScript language data. No security challenge was presented.

Table 1 *Communication Data Trace #1*

Offset	*Frame Description*	*Frame Data*
34359	Received V.21 command 0x04	FF C0 04 B5 00 44
35092	Received V.21 command 0x02	FF C0 02 0C 8C EC CC 4C 6C 9C AC 8C 2C 04 8C D4 A2 42 F2 22 82 04
35391	Received V.21 command 0x01	FF C8 01 00 72 0F 60
35394	Sending V.21 command 0xc4	FF C8 C4 B5 00 44
38466	Received V.21 command 0x04	FF C0 04 B5 00 44 03 03 CA
39198	Received V.21 command 0x02	FF C0 02 0C 8C EC CC 4C 6C 9C AC 8C 2C 04 8C D4 04 A2 42 F2 22 82 04
39497	Received V.21 command 0x01	FF C8 01 00 72 0F 60
39538	Sending V.21 command 0xc4	FF C8 C4 B5 00 44 03 03 44
41528	Sending V.29 (9600 bps) TCF ...	
43132	...sent	
45444	Received V.21 command0x21	FF C8 21
45056	Sending V.29 (9600 bps) error-corrected...	
106803	...sent 256 FCD's and 3 RCP's	
106829	Sending V.21 command 0xFD	FF C8 FD 00 00 00 FF
109733	Received V.21 command 0x31	FF C8 31
110247	Sending V.29 (9600 bps) error-corrected...	
131431	...sent 86 FCD's and 3 RCP's	
131434	Sending V.21 command 0xFD	FF C8 FD F4 00 80 AA
134341	Received V.21 command 0x313	FF C8 31
134344	Sending V.21 command 0xDF	FF C8 DF

In the first DIS burst the callee's NSF frame (0x04) initially only indicated that the callee understood Adobe's non-standard fax control protocol. The caller then issued a matching NSS frame (0xC4), indicating that it, too, understood the protocol. The caller gave no further information, so the callee re-issued its DIS burst, but this time it filled in its Adobe NSF frame more completely. The one shown has one subframe: **ADOBE_INFO**. In that subframe, the callee expressed willingness to accept transmission of PostScript language data. The caller then sent another NSS frame, indicating an intention to begin transmission of PostScript language data using a V.29 (9600 bps) modem, followed by a TCF burst. The receiver accepted this intention and the quality of the received signal, and so sent a CFR frame. Thereafter, an 87411-octet PostScript language program was transported in two ECM blocks: a full one and a partial one. Even when the PostScript language program will generate many pages of data, the PostScript language program is transported as a single ECM page.

3.2 Example Two

The trace of Table 2 shows a successful transmission of PostScript language that involves a security challenge. This trace is also from the point of view of the caller.

Table 2 *Communication Data Trace #2*

Offset	*Frame Description*	*Frame Data*
33765	Received V.21 command 0x04	FF C0 04 B5 00 44
34505	Received V.21 command 0x02	FF C0 02 0C 8C EC CC 4C 6C 9C AC 8C 2C 04 8C D4 04 A2 42 F2 22 82 04
34796	Received V.21 command 0x01	FF C8 01 00 72 0F 60
34800	Sending V.21 command 0xc4	FF C8 C4 B5 00 44
38155	Received V.21 command 0x04	FF C0 04 B5 00 44 03 03 CA 0A 05 E4 39 F1 42 2B 07 D6 B6
38886	Received V.21 command 0x02	FF C0 02 0C 8C EC CC 4C 6C 9C AC 8C 2C 04 8C D4 04 A2 42 F2 22 82 04
39186	Received V.21 command 0x01	FF C8 01 00 72 0F 60
39228	Sending V.21 command 0xc4	FF C8 C4 B5 00 44 03 03 44 06 05 65 89 58 E0
41585	Sending V.29 (9600 bps) TCF ...	
43166	...sent	
44535	Received V.21 command0x21	FF C8 21
45532	Sending V.29 (9600 bps) error-corrected...	
106433	...sent 256 FCD's and 3 RCP's	
106460	Sending V.21 command 0xFD	FF C8 FD 00 00 00 FF
109318	Received V.21 command 0x31	FF C8 31
110321	Sending V.29 (9600 bps) error-corrected...	
130679	...sent 86 FCD's and 3 RCP's	
130682	Sending V.21 command 0xFD	FF C8 FD F4 00 80 AA
133565	Received V.21 command 0x31	FF C8 31
133569	Sending V.21 command 0xDF	FF C8 DF

The receiver’s administrator gave his secret key to the printer as

```
(%Fax%) << /PostScriptPassword (CCITT) >> setdevparams
```

The printer ran the string CCITT through MD5_32, which yielded a 32-bit digested secret key of C9 5C 58 FD.

This time when the callee expanded and re-issued its Adobe NSF frame in response to the caller's NSS frame, the Adobe NSF frame contained two subframes: **ADOBE_INFO** and **ADOBE_SECURITY**. In the **ADOBE_SECURITY** subframe, the callee issued the random security challenge 2B 07 D6 B6, and hinted that mixing the challenge with its 32-bit digested secret key and then applying MD5_32 twice to the result would give the result E4 39 F1 42.

The caller decided that he could pass the challenge, and issued a second Adobe NSS command. This time, he requested transmission of PostScript language data in the **ADOBE_INFO** subframe; and in the **ADOBE_SECURITY** subframe, claimed that the result of mixing the challenge with the secret key and then applying MD5_32 once to the result is 65 89 58 E0.

The callee checked that applying MD5_32 once to 65 89 58 E0 yields E4 39 F1 42, checked the TCF, and since all was well, issued a CFR. If the callee decided that the caller had failed the security challenge, he could issue an immediate DCN, or he could re-issue the DIS burst with the **ADOBE_INFO** subframe now indicating no ability to receive transmission of PostScript language data. See Table 1 for limitations on whether the caller is permitted to send T.4-compressed raster immediately.

3.3 Example Three

In the trace of Table 3, an unsuccessful transmission of PostScript language data is shown that involves a security challenge. The trace is also from the point of view of the caller, and the callee's secret key is CCITT as before.

Table 3 *Communication Data Trace #3*

Offset	*Frame Description*	*Frame Data*
27502	Received V.21 command 0x04	FF C0 04 B5 00 44
28236	Received V.21 command 0x02	FF C0 02 0C 8C EC CC 4C 6C 9C AC 8C 2C 04 8C D4 04 A2 42 F2 22 82 04
28535	Received V.21 command 0x01	FF C8 01 00 72 0F 60
28538	Sending V.21 command 0xc4	FF C8 C4 B5 00 44
31890	Received V.21 command 0x04	FF C0 04 B5 00 44 03 03 CA 0A 05 69 C7 29 33 25 0A DC 93
32621	Received V.21 command 0x02	FF C0 02 0C 8C EC CC 4C 6C 9C AC 8C 2C 04 8C D4 04 A2 42 F2 22 82 04
32920	Received V.21 command 0x01	FF C8 01 00 72 0F 60
32957	Sending V.21 command 0xDF	FF C8 DF

This scenario is functionally the same as the previous one, until the caller realized that his secret key, after mixing with the challenge and being transformed twice by MD5_32, did not yield the same result as in the **ADOBE_SECURITY** subframe of the NSF frame. At that point, because the callee's **ADOBE_INFO** bit 8 was 0, T.4 transmission of raster data was no longer permitted (he might also have had no T.4 raster ready), he issued a DCN frame and disconnected.

If the caller had persisted, and attempted to send a PostScript language file in the face of a security violation, the callee would have responded with another DIS burst, in which bit 4 of the **ADOBE_INFO** subframe of the NSF frame would have been 0, indicating an unwillingness to accept a PostScript language file. This unwillingness would have persisted until the end of the telephone connection.

Appendix A: Font Resources

All PostScript fax printers will be at least PostScript Level 2 language capable. PostScript language programs generated by printer drivers typically include any fonts that the printer driver does not expect to be available to the interpreter. The most common fonts available to the interpreter are those contained in ROM on the printer controller. Printer drivers assume that the standard 35 Adobe fonts listed below are in the ROM of a PostScript fax printer.

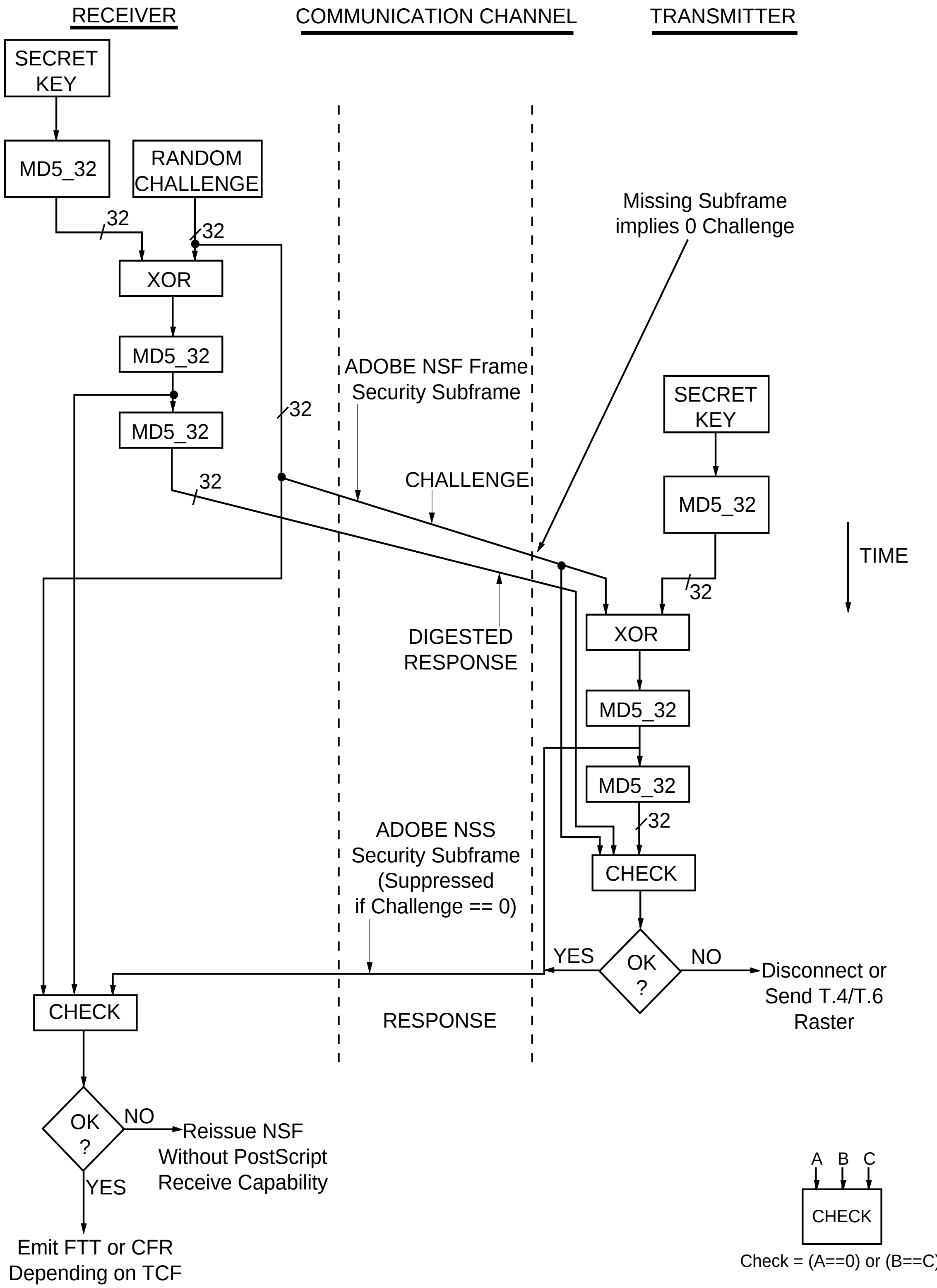
```
(AvantGarde-Gothic)
(AvantGarde-BookOblique)
(AvantGarde-Demi)
(AvantGarde-DemiOblique)
(Bookman-Demi)
(Bookman-DemiItalic)
(Bookman-Light)
(Bookman-LightItalic)
(Courier)
(Courier-Bold)
(Courier-BoldOblique)
(Courier-Oblique)
(Helvetica)
(Helvetica-Bold)
(Helvetica-BoldOblique)
(Helvetica-Narrow)
(Helvetica-Narrow-Bold)
(Helvetica-Narrow-BoldOblique)
(Helvetica-Narrow-Oblique)
(Helvetica-Oblique)
(NewCenturySchlbk-Bold)
(NewCenturySchlbk-BoldItalic)
(NewCenturySchlbk-Italic)
(NewCenturySchlbk-Roman)
(Palatino-Bold)
(Palatino-BoldItalic)
(Palatino-Italic)
(Palatino-Roman)
(Symbol)
(Times-Bold)
```

```
(Times-BoldItalic)
(Times-Italic)
(Times-Roman)
(ZapfChancery-MediumItalic)
(ZapfDingbats)
```

There may indeed be additional fonts available to the PostScript interpreter, besides those in controller ROM. Fonts on a printer-attached hard disk are available to the interpreter, as are fonts that have been downloaded into printer memory. The PostScript printer driver will not include these fonts in the PostScript language file sent from computer to printer. If that printer then transmits the PostScript language file via the PSFT protocol to another PostScript fax printer, the required fonts for the final print job may not be available to the receiving PostScript interpreter. The result will be font faulting, usually in the Courier font.

Appendix B: Security Flow Chart

The following diagram charts the flow of communication between a receiver and transmitter during the security check for a shared secret key. Acomplete description of the **ADOBE_SECURITY** subframe (subframe type 5) is found in section 2.3.



RECEIVER
COMMUNICATION CHANNEL
TRANSMITTER
SECRET KEY
MD5_32
RANDOM CHALLENGE
32
32
XOR
MD5_32
MD5_32
32
32
Missing Subframe implies 0 Challenge
ADOBE NSF Frame Security Subframe
CHALLENGE
SECRET KEY
MD5_32
TIME
32
XOR
DIGESTED RESPONSE
MD5_32
MD5_32
32
ADOBE NSS Security Subframe (Suppressed if Challenge == 0)
CHECK
YES
OK ?
NO
Disconnect or Send T.4/T.6 Raster
CHECK
RESPONSE
OK ?
NO
Reissue NSF Without PostScript Receive Capability
YES
Emit FTT or CFR Depending on TCF
A B C
CHECK
Check = (A==0) or (B==C)

Index