Hewlett Packard
Enterprise

Atalla Key Block Command Reference Manual

November 2016
Version 1.60

Part Number: AJ556-9005N

Atalla Key Block Command Reference Manual
November 2016

Contents

Document History

Part Number	Product Version	Published
587398-001	version 1.0	November 2009
AJ556-9005A	version 1.10	April 2010
AJ556-9005B	version 1.11	June 2010
AJ556-9005C	version 1.13	October 2010
AJ556-9005D	version 1.14	December 2010
AJ556-9005E	version 1.15	March 2011
AJ556-9005F	version 1.17	May 2011
AJ556-9005G	version 1.30	July 2011
AJ556-9005H	version 1.31	October 2011
AJ556-9005J	version 1.35	September 2012
AJ556-9005L	version 1.40	June 2014
AJ556-9005M	version 1.50	May 2016

Abstract

The Atalla Key Block Command Reference Manual defines the command and response syntax for the standard cryptographic functions in the A10160, A9160 and A8160. These devices are referred to as Network Security Processors (NSP)s. The Atalla Key Block (AKB) is a structure designed to securely store and control the use of TripleDES (3DES), AES, HMAC, and RSA public and private keys.

NSPs are designed to provide privacy, integrity, and performance in Automated Teller Machine (ATM), Electronic Funds Transfer (EFT), and Point Of Sale (POS) networks. They can also be used for other types of applications that require triple DES (3DES) support.

Custom commands, which are commands developed for specific customers, are not documented in this manual.

Throughout this manual the generic term “SCA” refers to the Secure Configuration Assistant-3.

What's New in This Manual

New and Changed Information

Version 1.60 Changes

- Command 357 has been modified.

Version 1.51 Changes

- Premium value commands 12F and 136 have been modified.

Version 1.50 Changes

- These premium value commands have been added: 136 and 139.
- These commands have been added: 119, 304, and 305.
- These utility commands have been added: 102, 103, 1104, and 1113.
- These premium value commands have been modified: 120, 121, 123, 124, 125, 12A, 12B, 12F, 3F1, and 39A.
- These commands have been modified: 3A, 97, 101, 365, 1101, and 1110.
- Option 28 has been added.
- The letters B, G, b, i, k, and w are allowed values in AKB Header Byte 1, Key Usage.
- The letter k is an allowed value in AKB Header Byte 7, Other Information.
- The internal logic which supports the processing of Atalla Key Blocks has been optimized resulting in an overall performance improvement.
- To support local network redundancy both NIC1 and NIC2 can now be connected to the same subnet.
- At startup, if the Network Security Processor's detects an error in the config.prm file it will attempt to use the previously saved last-known-good-config.prm file.
- The keywords PORT_ASCII and PORT_ASCII_2 can have a value of zero (disabled).
- The keyword DIAGTEST_TIME provides the ability to perform a daily Network Security Processor self-test at a specified time.

- The keywords ALLOW_IP and ALLOW_IP_2 provide the ability to specify the IP address of host systems which are allowed to connect to the Network Security Processor.
- The number of keyword/value pairs allowed in the config.prm file has been increased to 500.
- Additional diagnostic information has been added to the system log.
- Status and console messages display on the status port in standard system log format.
- The Ax160 NSP's TCP/IP stack has been upgraded to mitigate a denial of service attack as documented in CVE-2004-0230 (listed in the NIST National Vulnerability Database).
- These existing premium value commands have been added to this manual: 390 and 392.

Version 1.47 Changes

- These premium value commands have been added: 3F0 and 3F1.

Version 1.46 Changes

- Premium value command 38C has been added.
- Command 11A has been added.
- Premium value command 12F has been modified.

Version 1.45 Changes

- These premium value commands have been added: 3FC and 3FD.

Version 1.44 Changes

- Premium value command 376 has been modified.

Version 1.43 Changes

- These premium value commands have been added: 375, 376, and 377.

Version 1.42 Changes

- These premium value commands have been added: 135, 138, 324, 325, 326, 365, and 374.
- These commands have been added: 102, 1104, and 1113.
- These premium value commands have been modified: 131, 134, 351, and 358.
- These commands have been modified: 16F, 335, 350, 352, 1227, and 1228.

Version 1.40 Changes

- The procedure to enable printing commands has changed, see Enable and disable printing commands.
- These premium value commands have been added: 12A, 39A, 39B, and 39C.
- Command 16A has been added.
- The premium value option E5 has been added.
- These security relevant options have been added: 42 and 43.
- These commands have been modified: 9E, 354 and 359.
- These premium value commands have been modified: 121, 123, 124, 125, 351 and 3FA.
- The letter R is an allowed value in AKB Header Byte 1, Key Usage.
- The number 2 and the letter H are allowed values in AKB Header Byte 2, Algorithm.
- The command 350, 36B and 36C processing logic has been improved.
- The maximum value for the MAX_CLIENTS_ASCII, MAX_CLIENTS_ASCI_2 and MAX_CLIENTS_PRTCMD keywords has been increased to 64.
- The maximum value for a port number has been increased to 65534.
- The Network Security Processor's operating system has been upgraded.

Version 1.38 Changes

- The following premium value commands have been added: 36B and 36C.
- The letter I is an allowed value in AKB Header Byte 7, Other Information.

Version 1.37 Changes

- Customer specific premium value command 32E has been added.

Version 1.36 Changes

- Customer specific premium value commands 389, 38A, and 38B have been added.

Version 1.35 Changes

- The following commands which support the printing of PINs and key components have been added: 15E, 161, 162, 163, 16E, and 16F.
- To support the printing functionality, four new parameters have been added to the config.prm file. For information on these parameters, see section 4 of the *Installation and Operations Guide for the Atalla Ax160 NSP*.
- The lowercase letter p is an allowed value in AKB Header Byte 3, Mode of Use.
- The Network Security Processor will not start successfully if there are errors in the config.prm file.

Version 1.33 Changes

- Customer specific premium value commands B7, 341, 342, and 343 have been added.

Version 1.32 Changes

- Standard command 36A has been added.
- Customer specific premium value commands DB has been added.
- Customer specific premium value command 323 has been added.
- The number 4 is an allowed value in AKB Header Byte 2, Algorithm.
- Premium value command 12F has been modified.

Version 1.31 Changes

- Customer specific premium value command 322 has been added.

Version 1.30 Changes

- Option 4F has been added.

- Command 7E has been modified.
- Commands 35A and 35B have been modified to support CSC-2.
- In command 1216 the maximum value for the battery date counter has been reduced to 700 days.
- The performance of commands that return responses containing large amounts of unpacked ASCII data has been improved.

Version 1.17 Changes

- Customer specific premium value command 3A4 has been added.
- The accuracy of the NIC2 information in the system log has been improved.

Version 1.16 AKB Changes

- The list of headers allowed in command 392 has been expanded.

Version 1.15 AKB Changes

- Customer specific premium value commands 390, 391, and 392 have been added.
- An additional field has been added to the command 9A#ID response. This field is required to support the newly added 39x commands mentioned above.
- Command 9E supports multiple AKB lengths and the CMAC check digit method.
- Customer specific premium value option C1 has been added.
- Command BD documentation and examples have been updated to include the [MAC Type#] field.

Version 1.14 AKB Changes

- Customer specific premium value command 328 has been added.
- The error reporting logic in commands 117 and 118 has been improved.
- Utility command 1223 has been modified to support option 023.
- The list of allowed working key header values allowed in command 13 has been expanded.

Version 1.13 AKB and Variant Changes

- Customer specific premium value command 332 has been modified.

- Premium value option 87 has been added.
- Utility command 105 has been modified to allow lowercase characters in the serial number field.
- The operating system in the Atalla Cryptographic Engine has been updated.

Version 1.12 Changes

- Customer specific premium value commands 308 and 309 have been modified.

Version 1.11 Changes

- Standard command 35F has been added.
- Customer specific premium value commands 3A2 and 3A3 have been added.
- The letter O is an allowed value in the AKB header Byte 2, Algorithm.
- The ability to utilize the second Network Interface Connection (NIC2) on the Atalla Cryptographic Engine has been added. The *Installation and Operations Guide for the Atalla Ax160 NSP* contains detailed information; see section 2 for the location of NIC2 and section 4 for configuration parameters.

About This Manual

Who Should Read This Manual

This manual is written for host application programmers who need to add hardware DES cryptographic support to their applications.

This manual is organized into the following sections:

- Section 1, Introduction, provides an overview of the command and response format, data formats, cryptographic functions supported, and provides information on communicating with the Network Security Processor.
- Section 2, Using DES Keys, describes the different types of cryptographic DES keys used by the Network Security Processor. It also explains the differences between single and triple DES. Key parity, and the use of the Atalla Key Block (AKB) is also covered in this section.
- Section 3, Key Management, describes the commands and responses used to generate and or translate working keys for use in an ATM, POS, and EFT networks.
- Section 4, Processing Personal Identification Numbers, describes the commands and responses used to encrypt, generate, translate, and verify PINs.
- Section 5, Processing Transaction Data, describes the command and responses used to encrypt and decrypt data, and generate random numbers.
- Section 6, Authenticating Transaction Data, describes the commands and responses used to generate and verify message authentication codes.
- Section 7, Authorizing VISA, MasterCard, American Express and Discover Cards, describes the commands and responses used to generate and verify Card Verification Values (CVV), Card Validation Codes (CVC), and Card Security Codes (CSC).
- Section 8, Processing EMV and VISA Stored Value Cards, describes the commands and responses used to generate and verify S1, S2, and S3 signatures.
- Section 9, Storing Values in the Volatile Table, describes the commands and responses used to store, and delete DES keys, conversion tables, and rows of Diebold Number tables from the volatile table.
- Section 10, Using Public Key Technology to Initialize Remote Devices, describes the commands and responses used to key remote devices, such as ATMs, that support public key technology.

- Section 11, Printing Commands, describes the commands and responses used to print key components or PIN letters.
- Section 12, Utility Commands, describes the basic utility commands, and provides their calling and responding parameters.
- Section 13, Error Messages, defines the error response format and lists the application error types.
- Appendix A, Introduction to Cryptography and Cryptographic Keys, describes cryptographic standards and terms.
- Appendix B, Understanding Financial Interchange Networks, explains how to initialize a network.
- Appendix C, Summary of Commands and Options, is a reference that lists commands and where they are located in the manual.
- Appendix D, Obtaining technical support, provides email and telephone contact information.
- Glossary, provides definitions of terms used in this manual.

The manual is provided in “electronic” form, as a.PDF file. PDF files can be viewed with Adobe Acrobat. Hypertext links are included to allow you to quickly locate specific information.

Your Comments Invited

After using this manual, please take a moment to send us your comments via an email message. Be sure to include your name, company name, address, and phone number in your message. If your comments are specific to a particular manual, also include the part number and title of the manual. The email address is:

DataSecurity.Atalla.Support@hpe.com

Many of the improvements you see in manuals are a result of suggestions from our customers. Please take this opportunity to help us improve future manuals.

Related Documents

- *Installation and Operations Guide for the Atalla Ax160 NSP*

If you purchase a Secure Configuration Assistant, you will receive the following document:

- *Atalla Secure Configuration Assistant-3 Users Guide*

Type Conventions

Hypertext Links

Green text is used to indicate a hypertext link. By clicking a passage of green text, you are taken to the location described.

For example:
See Data formats for information on how to include these special characters in your command data.

Key Presses

Keys you press are shown in boldface Helvetica type.

Example: Press the **clear** key to return to the Main Menu.

Emphasis

Words that are emphasized are shown in italic or bold.

Example: You *must* create a Master File Key (MFK).

Key Cryptogram Notation

Secret key values are sent to the Network Security Processor in an encrypted form and formatted in the Atalla Key Block. The following notation indicates that a Key Exchange Key (KEK) is encrypted under the Master File Key (MFK). The header defines the permissions associated with the KEK, and the MAC cryptographically links the header to the encrypted key value. The commas are required but are not used in the MAC calculation. All key cryptograms throughout this manual will be displayed in this format.

Header,$E_{MFK.E}$(KEK),MAC

To aid readability, long strings of characters, such as clear-text key values, will be split into groups of four characters. For example:

The clear-text key value: 0123456789ABCDEF, will be shown as:

```
0123 4567 89AB CDEF
```

Do not include these spaces when sending commands to the Network Security Processor.

Examples

Examples and explanations are shown in Courier type.

Example:

COMMAND

<10#Header#Header,$E_{MFK.E}$(KEK),MAC#[Key Length#]>

RESPONSE

<20#Header,$E_{MFK.E}$(WK),MAC#Header,$E_{KEK.E}$(WK),MAC#

Working Key Check Digits#>[CRLF]

Optional Fields

Fields in the command and response syntax descriptions that are surrounded by square brackets are optional. The location of the closing square bracket is significant. If the closing square bracket follows the field delimiter (#) the entire field is optional. If the closing square bracket precedes the field delimiter (#) the field is required but can be empty. For example:

The key length field is optional:

<10#Header#Header,$E_{MFK.E}$(KEK),MAC#[Key Length#]>

The key length field is required but can be empty:

<10#Header#Header,$E_{MFK.E}$(KEK),MAC#[Key Length]#>

Even Page Numbering

Each section in this manual ends on an even page, even if the page is blank. This practice enables each section to start on an odd-numbered page, which helps give the manual a consistent appearance.

1

Introduction

This section describes the cryptographic functions supported by the Network Security Processor, the command and response message format, error reporting, and data formats.

Cryptographic Functions

The Network Security Processor supports the following cryptographic functions:

- Key Management which includes key generation, and key translation.
- Processing Personal Identification Numbers which includes encrypting, translating, and verifying PINs.
- Processing Transaction Data which includes encrypting and decrypting data.
- Authenticating Transaction Data which includes generating and verifying Message Authentication codes.
- Authorizing VISA, MasterCard, American Express and Discover Cards which includes generating and verifying credit and debit authorization codes
- Processing EMV and VISA Stored Value Cards which includes generating and verifying Authorization Request Cryptograms, generating message authentication codes, and generating and verifying VCSC signatures.
- Using Public Key Technology to Initialize Remote Devices which includes generating keys, importing and exporting keys, and generating and verifying digital signatures.

Operating Overview

The Network Security Processor must be initialized with a Master File Key before it can process a cryptographic command. Utility Commands do not require the Network Security Processor to be initialized with a Master File Key.

Network Security Processor operation occurs in three phases:

- Command. The host application writes the command to the Network Security Processor.
- Processing. The Network Security Processor performs the requested actions based on the specific command received.
- Response. The Network Security Processor returns the response. The host application reads the response.

Command and Response

The application programming interface consists of a set of specific commands to which a response or error message is returned. The host application must send the command as a contiguous strings of characters. The TCP/IP message that contains the command to be processed by the Network Security Processor must end with the “#>” end-of-command characters.

note The host application must not send any additional characters after the “#>” end-of-command characters.

Commands are identified by an ID and have the following format:

```
<CMDID#FIELD 1#FIELD 2#FIELD N#[^Context Tag#]>
```

where:

<	Starts the command
CMDID	Is the two, three, or four character Command ID
#	Is a delimiter after each command field (including the last field)
Field	Is the command data (fields vary in length and number)
^	Context Tag (optional)
>	Ends the command

Characters preceding the “<” start of command character are ignored. Characters following the “>” end of command character are also ignored. These four characters (“<”, “#”, “^”, and “>”) have special meaning to the Network Security Processor, and therefore cannot be included in the command data fields. In Ethernet TCP/IP communications, the carriage return (CR) character is also a special character that is interpreted as an end-of-command, it also cannot be included in the command data. See to Data formats for information on how to include these special characters in your command data.

Any cryptographic command can include an additional field after all required fields. This field is optional, and can be used to supply "context" information which is returned with the response message. The first character of this field must be a ASCII hexadecimal 5E (^). The remaining data can be variable in length but it may not contain the #, >, <, or CR characters, or exceed the maximum command length of 5,000 characters.

The response format is identical to the command format with the exception that a carriage return CR (hexadecimal 0D) and a line feed LF (hexadecimal 0A) may follow the ">". The carriage return and line feed are denoted as [CRLF]. This capability is configurable, the default is CRLF is appended to the response; see option 023 to remove the CRLF from the response.

Input commands have odd-numbered first digits; the first digit of the corresponding response is a digit higher. For example, if 10 is the command ID, 20 is the response ID, if 37B is the command ID, 47B is the response ID. See Appendix C, "Summary of Commands and Options" for a listing of commands.

The following example shows the command and response for Command 10, notice that the CRLF is appended to the response:

Command

```
<10#1PUNE000#1KDNE000,E801D324A94F85BC5833A73B5E804ACF5B87CBA329D1A
74C,7CA7039EDE2AB473#>
```

Response

```
<20#1PUNE000,BAEB5B09AB8199B9732DB8E98BD649F66D1CC584B1204A5A,042BE
F3E6E59D9C9#1PUNE000,CE54B67FC9AD572E0BAECC1F7142F631485F54F6DDB91B
9F,66FE4A998CB954DF#DE9D#>CRLF
```

The following example shows the command and response for Command 10 using a context tag:

Command

```
<10#1PUNE000#1KDNE000,E801D324A94F85BC5833A73B5E804ACF5B87CBA329D1A
74C,7CA7039EDE2AB473#^Generate KPE for ATM 325#>
```

Response

```
<20#1PUNE000,BAEB5B09AB8199B9732DB8E98BD649F66D1CC584B1204A5A,042BE
F3E6E59D9C9#1PUNE000,CE54B67FC9AD572E0BAECC1F7142F631485F54F6DDB91B
9F,66FE4A998CB954DF#DE9D#^Generate KPE for ATM 325#>CRLF
```

Error Responses

If the Network Security Processor encounters a syntax error, an error response message is returned. Use the following information to decode the error response. If you are contacting HPE Security - Data Security technical support for assistance, be sure to provide the exact command and error response.

The format of the error response is:

```
<00#XXYYZZ#>
```

The response ID of 00 indicates an error is being returned.

`XX` – indicates the error number, Table 13-1 on page 13-1 lists the error number and its description.

`YY` – the first field found to be in error. The command ID field is field zero. If this field returns the value 00, any of the following may be true:

- The command specified an invalid command number.
- A necessary MFK is missing.
- In response to an echo command (Command (ID = 00).

`ZZ` – the software version of the cryptographic command processor. This field returns a two digit software version number, use Get ID of Current Image (Command 1101) to obtain more information on the software version.

note If a binary zero is present in an field that does not allow binary data, the context tag will not be present in the error response.

Here is an example of an invalid command 10, (field 2 contains 73 characters instead of 74).

Command

```
<10#1PUNE000#1KDNE00,E801D324A94F85BC5833A73B5E804ACF5B87CBA329D1A7
4C,7CA7039EDE2AB473#>
```

Response

```
<00#010210#>CRLF
```

The error indicates length out of range in field 2, software version is 1.00.

Detailed Errors

The detailed error is appended as a separate field after the error field (XXYYZZ). Detailed errors are included if option 021 is enabled. Table 13-2 on page 13-3 describes the detailed application error messages.

Here is an example of an invalid command that returns an error and a detailed error.

Command

```
<20#1PUNE000#1KDNE000,E801D324A94F85BC5833A73B5E804ACF5B87CBA329D1A
74C,7CA7039EDE2AB473#>
```

Response

```
<00#030010#0201##>CRLF
```

The error indicates value out of range in field 0, software version is 1.00. The detailed error (201) indicates Invalid Command.

Data formats

Commands and responses usually contain only hexadecimal characters. Each character is one byte. The Network Security Processor requires ASCII characters.

There are a limited number of commands that do not have fields that specify data type and length, the command Generate MAC and Encrypt or Translate Data (Command 59) is an example. In some instances, the data to be processed may be unprintable, or contain control characters for some protocols, or include the special characters (‘<‘, ‘#’, ‘^’, ‘>’, carriage return, and line feed). For example, to encrypt or authenticate this message:

Sell stock when price is > $50.

The “>” character causes the Network Security Processor to incorrectly terminate the message. There are two ways to resolve this issue:

- The host application converts each character of data to ASCII hexadecimal. This allows all possible characters to be processed. The example data ‘Sell stock when price is > $50.’ would be converted to:
  ```
  53656C6C2073746F636B207768656E207072696365206973203E20
  2435302E
  ```

- Use a command that supports binary data and includes data type and data length fields, such as, Generate MAC (Command 98).

Programming Guidelines

If your host application is running on an HP NonStop Server, you can use Boxcar or the Atalla Resource Manager (ARM) to manage the host to Network Security Processor TCP/IP Ethernet interface. For more information, see *the Boxcar Reference Guide*.

The remainder of this section applies to a Unix host environment. Communicating with the Network Security Processor using Ethernet TCP/IP involves the following basic steps:

- Setting up the application
- Opening the socket interface
- Connecting the socket to the Network Security Processor
- Sending the command
- Receiving the response
- Closing the socket

The following subsections explain each of the above programming steps.

Setting Up the Application

The host application should interact with the Network Security Processor in a single-threaded manner. This means you must send a command and wait for a response. Also, if you are using the C programming language in a UNIX environment, be sure to include the following files:

- sys\types.h
- sys\socket.h
- netinet\in.h
- stdio.h

Opening the Socket Interface

The **socket** system call requires the domain, socket type, and protocol. The Network Security Processor socket should be coded to use Internet domain (AF_INET), a stream socket (SOCK_STREAM), and the default protocol (0) for SOCK_STREAM.

Connecting a Socket to the Network Security Processor

The **connect** system call is used to establish the connection. When connecting the socket to the Network Security Processor, you need to specify the IP address of the Network Security Processor and the TCP port number. The maximum number of socket connections allowed for each NIC is based on the config.prm file parameters MAX_CLIENTS_ASCII and RECONNECT. Refer to the Installation and Operations Guide for documentation on these parameters. Requests to open additional sockets, beyond the maximum, will immediately receive a close socket from the Network Security Processor.

Sending Commands to the Network Security Processor

Use the **send** system call to send the command to the Network Security Processor.

Receiving Responses

Use the **recv** system call to received the Network Security Processor response. Since a response from the Network Security Processor can exceed the length of a single Ethernet frame, it is important to code your application in a way that ensures that you have received the entire response. All commands end with #>. If option 23 is disabled, a carriage return and line feed 0x0D and 0x0A will be appended to the end of the response. A response to a command that contains binary data may have the #> as part of the response data. Check the response data length field to be sure it contains the entire response.

Closing a Socket

The **close** call is used to terminate the session with the Network Security Processor. The application should close the Network Security Processor sockets before the application terminates. Sockets that are not closed remain open, thereby reducing the number of sockets available. Opening and closing a socket for each command is not recommended.

Sample program

The following sample program can be used to communicate with a Network Security Processor over TCP/IP.

```
/ * Example code for Network Security Processor TCP/IP communication */

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define MAX_MSG 8192
```

```
/*
* Assume the response will be returned in 10 byte chucks. This
* demonstrates how to look for the end of a command across multiple
* packets.
*/

#define PKT_READ_SIZE 10
int
main(void)
{
  char ipaddr[40];               /* IP address */
  int portnumber = 0;            /* IP Port number */
  char message[MAX_MSG];        /* Buffer of message being sent */
  char msgrsp[PKT_READ_SIZE];   /* Buffer of message being read back */
  char retmsg[MAX_MSG];          /* Buffer that contains response */
  int msglen = 0;
  int rcvlen = 0;
  int msg_done = 0;
  int rsp_start = 0;
  int socketnum = 0;
  struct sockaddr_in aname;
  int status = 0;
  int rsp_ptr = 0;
  /*
   * Load IP address, port number, and message
   */
  sprintf(ipaddr, "%s", "192.168.1.100");
  portnumber = 7000;
  sprintf(message, "%s", "<1101#>");
  msglen = strlen(message);
  /*
   * Create a socket
   */
  socketnum = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
  if (socketnum < 0)
  {
    printf("Unable to obtain socket number\n");
    exit(2);
  }
  /*
   * Set socket infomation in the socket structure
   */
  aname.sin_family = AF_INET;
  aname.sin_port = htons(portnumber);
  aname.sin_addr.s_addr = inet_addr(ipaddr);
  /*
   * Connect to the target Network Security Processor
   */
  if (connect(socketnum, &aname, sizeof(aname)) < 0 )
  {
    printf("Connection error");
    close(socketnum);
    exit(2);
  }
   /*
   * Send the message
```

```
 */
status = send(socketnum, message, msglen, 0);
if (status < 0)
{
  printf("Unable to send to socket\n");
  close(socketnum);
  exit(2);
}
/*
 * Fetch the response
 */
rsp_start = 0;
do
{
  int i = 0;
  rcvlen = recv(socketnum, msgrsp, (size_t)(PKT_READ_SIZE), 0);
  if (rcvlen < 0)
  {
    printf("Unable to receive from socket\n");
    close(socketnum);
    exit(2);
  }
  if (rcvlen == 0)
  {
    printf("Received 0 length message\n");
    close(socketnum);
    exit(2);
  }
  i = 0;
  if (rsp_start == 0)
  {
     /*
     * Search for the start of the response
     */
    for (; i<rcvlen; i++)
    {
     if (msgrsp[i] == '<')
     {
      /*
        * Found the start of the response
      */
       rsp_start = 1;
       break;
    }
    }
  }
  if (rsp_start != 0)
  {
    /*
     * Process the response, copy characters into the output buffer
     */
    for (; i<rcvlen; i++)
    {
     /*
      * Error if response get too big for the buffer allocated
      */
```

```
          if (rsp_ptr >= MAX_MSG - 1)
          {
            printf("Error: response would overflow buffer\n");
            exit(2);
          }
          retmsg[rsp_ptr++] = msgrsp[i];
          if (msgrsp[i] == '>')
          {
           msg_done = 1;
           break;
          }
         }
       }
         /*
        * Continue to perform socket reads to get the whole response
        */
     } while (msg_done == 0);
     /*
      * Null terminate the response string for printf
      */
     retmsg[rsp_ptr] = 0;
     /*
      * Output the response
      */
     printf("Message:  %s\n", message);
     printf("Response: %s\n", retmsg);
     close(socketnum);
   }
```

2

Using DES Keys

A secure financial system network has several types of DES keys, each of which are used for a specific purpose. The majority of these keys are generated by the Network Security Processor, and returned to the host application in two forms. One form is for use by the Network Security Processor. In this form, the DES key is encrypted under the Master File Key and stored in the host application's key database. The second form is for use by the remote system. In this form, the DES key is encrypted under the Key Exchange Key. The most common uses of these DES keys are to encrypt, translate, and verify PINs, encrypt and decrypt data, generate and verify message authentication codes, and generate and verify card verification values.

The Network Security Processor will not accept or return clear-text DES key values. All DES keys must be supplied encrypted under the Master File Key which resides within the secure boundary of the Network Security Processor. When importing a DES key that was generated on a remote system, the DES key must be encrypted under a Key Exchange Key.

Single length DES keys contain 64 bits, in this manual the name of this type of key is 1key-3DES (single-length). Double length DES keys contain 128 bits, in this manual the name of this type of key is 2key-3DES (double-length). Triple length DES keys contain 192 bits, in this manual the name of this type of key is 3key-3DES (triple-length). In the variant personality only the Master File key can be a triple-length key, all other DES keys must be either single or double length. For more information on DES keys, see Key Attributes.

For information on RSA keys, see Atalla Key Block (AKB) formats for RSA Keys.

To skip this introduction, go to Atalla Key Block for information on headers and how an Atalla key block is generated.

Master File Key

The Master File Key (MFK) encrypts Key Exchange Keys and working keys. The MFK is never used to encrypt PINs or data, it is never shared with another node. The length of the Master File Key must be equal to or greater than the length of the Key Exchange Keys and working keys it protects. Security Officers use the SCA to create components of the Master File Key, and then send them to the Network Security Processor. These components are combined to form a secret MFK key which is stored in the Network Security Processor's Non-volatile Key

Table. To minimize downtime when replacing the Master File Key, a Pending MFK (PMFK) can be loaded into the Network Security Processor using the same procedure. For more information, see Procedure to replace the current MFK with the pending MFK.

Key Exchange Key (KEK)

To maintain secrecy, working keys are encrypted under a key called a Key Exchange Key (KEK) before they are sent from one node to another. Key Exchange Keys are unique for each network node. The length of the Key Exchange Key must be equal to or greater than the length of the working keys it protects.

Working Keys

Working keys are types of keys used to perform specific cryptographic operations. PIN Encryption Keys and Message Authentication Keys are two examples of working keys used for a specific purpose. Working keys are *not* stored in the non-volatile key table.

The Network Security Processor default security policy does not allow 1key-3DES (single-length) working keys. If you must support 1key-3DES working keys, you must enable options 6C and 4A in the Network Security Processor's security policy. Generally, the length of the working keys can be either 1key-3DES, 2key-3DES, or 3key-3DES, however certain commands require that the working key be a specific length.

A secure system should not allow working keys to be used for other than their intended purpose. For example, it is a security violation to allow a Data Encryption Key to be used to decrypt PINs. The Atalla Key Block prevents keys from being misused.

In many financial networks, working keys are changed on a frequent basis and many devices, such as ATMs, have unique working keys. It is not uncommon for thousands of these working keys to exist at any one time, far too many to store within the Network Security Processor. In practice, the working keys are encrypted under the secret MFK which resides within the Network Security Processor, the resulting cryptogram is stored on a host database. This encrypted form of a key is called an Atalla Key Block. When a particular working key is needed to process a transaction, the host sends the Atalla Key Block of the working key to the Network Security Processor where it is decrypted using the MFK and then used to process the transaction data.

Atalla Key Block

The Atalla Key Block (AKB) has been designed to meet the financial services key management standards, such as ANS X9.24. The Network Security Processor also provides support for the TR-31 key block, see commands 119 and 11A. For information on the AKB for RSA keys, see Atalla Key Block (AKB) formats for RSA Keys.

You will see throughout the command syntax sections of this manual notations similar to this:

Header,$E_{MFK.E}$(Working Key),MAC

This represents the working key encrypted under the Master File Key and formatted in the Atalla Key Block.

The AKB consists of three fields. Commas are used to separate the fields, and aid in readability.

Header Field– This header is a flexible mechanism that controls the use of the key. This header must contain attributes such as the algorithm in which the key may be used, whether the key functions as a working or a key-encrypting-key, etc. The header field is comprised of eight printable ASCII characters, see Key Header.

Encrypted Key Field – The 3DES-CBC algorithm is used to encrypt the key block. The ASCII hexadecimal representation of the eight header characters is used as the Initialization Vector. To hide their length, 1key (single-length) keys are right padded with “53”, 2key (double-length) 3DES keys are right padded with “44”. Each byte of the Master File Key is exclusive or’d with “45”, the resulting key is used as the encryption key. The Encrypted Key field consists of 48 ASCII hexadecimal characters.

MAC Field – The MAC field contains a 3DES-CBC Message Authentication Code (MAC) which is computed across the encrypted key block and the header. The two commas in the AKB are not used in the MAC calculation. Each byte of the Master File Key is exclusive or’d with “4D”, the resulting value is used as a MAC key to generate the MAC. The MAC cryptographically binds the header and the encrypted key together. This prevents an adversary from manipulating key blocks contained in the Encrypted Key Field, and also allows the Network Security Processor to control key usage in a manner consistent with the header information. The MAC Field consists of 16 ASCII hexadecimal characters.

Key Header

This section contains tables listing the possible values for each of the eight bytes used in the header. Each byte describes attributes of the key. Before a key in AKB format is used in an Atalla Network Security Processor, the content of the header block is validated.

Table 2-1 Definition of the Header Bytes

Byte	**Definition**
0	Version Number
1	Key Usage
2	Algorithm
3	Mode of Use

Table 2-1 Definition of the Header Bytes (continued)

Byte	Definition
4	Exportability
5	Reserved
6	Special Handling
7	Other Information

Byte 0, Version Number

The version number of the header. Its value must be 1.

Byte 1, Key Usage

Defines the type of key.

note The “Variant” column refers to variant values required in non-AKB versions of Atalla products, this column is only relevant for key database migration purposes.

Table 2-2 Byte 1, Key Usage

Value	Definition	Variant
A	ATM Master Key	1, 5
B	CMAC 3DES Key	n/a
C	Card Verification Value / Card Validation Code	3
D	Data Encryption	2, 16, 17, 21, 22, 23, 24, 26
G	Gilbarco	n/a
I	Initialization Vector, Byte 7 = ‘M’= CV-KMAC, ‘P’= CV-KPE	6, 10, 20
J	Rijndael - AES	n/a
K	Key Encryption Key	0, 1, 28, 29, 31
M	Message Authentication Code, Byte 7 = ‘C’ = Challenge/Response	3, 7, 12, 18, 19
P	Pin Encryption Key	1, 5, 10, 14, 20, 25
R	Root Signing Key (used in RSA commands)	n/a
S	Signing Key (used in RSA commands)	n/a
T	Token Key	21, 22, 23, 24

Table 2-2 Byte 1, Key Usage (continued)

Value	Definition	Variant
V	PIN verification, KPV	4
b	Signing Token	n/a
c	Communication Key, Byte 7 = ‘B’ = PIN encrypt and MAC	1, 3
d	Derivation Key	8
i	DUKPT Initial PIN Encryption Key	n/a
k	TR-31/TR-34 Key Wrapping Key	n/a
m	Master Key, Byte 7 = ‘M’= MAC, ‘P’ = PAC, ‘m’ = MF-MAC, ‘p’ = MF-PAC	2, 3, 4, 8, 9, 11, 13, 26, 27
n	Diebold Number Table, Burroughs Number Table, Decimalization/Conversion Table.	5, 6
p	RSA Private Key	n/a
s	Signing Key (used in RSA commands)	n/a
w	TR-34 Key Message Signing Key	n/a

Byte 2, Algorithm

This byte defines what algorithms can be used with this key.

Table 2-3 Byte 2, Algorithm

Value	Definition
0	SDI Security Dynamic Algorithm
2	HMAC-SHA256
3	IBM 3624
4	ARIS
7	IBM 4731
B	Atalla Bilevel
C	Conversion Table
D	DES
E	EMV Key Derivation
F	Key Derivation
H	HMAC-SHA1

Table 2-3 Byte 2, Algorithm (continued)

Value	Definition
I	Identikey
J	Rijndael - AES
N	NCR
O	Discover
P	Preem
R	RSA
U	unknown or unspecified
V	VISA
X	American Express
Z	Key Derivation
a	Atalla 2x2
b	Key Derivation
d	Diebold Number Table
i	ICC Key Derivation
r	Routex Derivation
s	custom
u	Unisys (Burroughs)

Byte 3, Mode of Use

Defines the operation the key can perform.

Table 2-4 Byte 3, Mode of Use

Value	Definition
D	Decrypt only
E	Encrypt only
G	Generate only
I	Import RSA key
N	No special restrictions
R	RSA

Table 2-4 Byte 3, Mode of Use (continued)

Value	Definition
V	Verify only
X	Export RSA key
p	PIN Printing Key

Byte 4, Exportability

note Exportability refers to export from the Atalla Network Security Processor. Not exportable means the key can not be translated from encryption under the MFK to encryption under the KEK.

Table 2-5 Byte 4, Exportability

Value	Definition
E	Exportable under trusted key
N	Not exportable
S	Sensitive, exportable under untrusted key

Byte 5, Reserved

Not used. Its value must be 0 (zero).

Byte 6, Special Handling

Used to distinguish between working keys, key exchange keys, and key components.

Table 2-6 Byte 6, Special Handling

Value	Definition
0	No special considerations
1	AKB contains a derivation key component
2	AKB contains a key component of a 2 component key
3	AKB contains a key component of a 3 component key
4	AKB contains a key component of a 4 component key
C	AKB contains a key component - not a key

Table 2-6 Byte 6, Special Handling (continued)

Value	Definition
E	KEK can be used only to export a working key
I	KEK can be used only to import a working key
K	KEK can be used only to import a KEK
M	Master File Key
d	Derivation key
e	AKB contains a export key exchange key component
i	KEK can be used only to import a key component - not a key
k	AKB contains a key exchange key component
n	AKB contains a key component - not a key

Byte 7, Other Information

This byte further defines the key usage.

Table 2-7 Byte 7, Other Information

Value	Definition
0	Initialization/Control Vector. Byte 1 = I
1	ISO 16609 MAC algorithm 1 (using TDEA) ISO 9797-1 MAC algorithm 1
2	ISO 9797-1 MAC algorithm 2
3	ISO 9797-1 MAC algorithm 3
4	ISO 9797-1 MAC algorithm 4
5	ISO 9797-1 MAC algorithm 5
B	PIN and MAC Key. Only valid if Byte 1 = c
C	Challenge/Response Key. Only valid if Byte 1 = M.
E	EMV key if byte 1 = m, EMV Master Key - Confidentiality
I	IVR key
M	MAC key if byte 1 = I or m, EMV Master Key - Integrity
P	PIN Key if byte 1 = I or PAC key if byte 1 = m.
S	Sermepa Derivation

Table 2-7 Byte 7, Other Information (continued)

Value	Definition
T	SSL
a	EMV Master Key - Application Cryptogram
b	Bank Master key, Data Decrypt and MAC key
d	EMV Master Key - Data Authentication
e	EMV Master Key - Dynamic Numbers
f	EMV Master Key - Card Personalization
g	EMV Master Key - Other
k	TR-34 Wrapping
m	MAC Key. Only valid if byte 1 = m
n	Nordea Derivation Method
p	PAC Key. Only valid if byte 1 = m
r	Registration Master key

Working Key Headers

Table 2-8 on page 2-10, lists many of the supported working key headers. **Always refer to the command syntax for the specific headers allowed in a command**. Headers used in customer specific commands may not be included in this table.

Any header in this table can also have byte 4 set to “N”. For example 1KDNN000 is a valid KEK header. Certain commands will not allow byte 4 to have a value of “N”, for example the working key in field 3, in commands 11 and 1A. Check the individual command syntax to see if “N” is allowed in byte 4 of the header.

Any header listed in this table can also be an import KEK header when byte 6 is set to “I”. For example, 1PUNE0I0 would be the import KEK header for a PIN encryption key (1PUNE000). These import KEK headers are only used in command 11B.

note The “Variant” column refers to variant values required in non-AKB versions of Atalla products, this column is only relevant for database migration purposes.

Table 2-8 Working Key Headers

Type of Key	Key Name Abbreviation	Header	Variant
Master File Key	MFK	1KDDN0M0	0
Key Encryption Key Local Master Key Zone Master Key TMK Transport Key Initial ATM Master Key (IBM 4731)	KEK LMK ZMK TMK KTRAN IMK	1KDNE000	0
KEK, Encrypt Only (for key export)	KEK-EO	1KDEE000	0
KEK, Decrypt Only (for key import)	KEK-DO	1KDDE000	0
PIN Encryption Key	KPE	1PUNE000	1
PIN Encryption Key (DES)	KPE-DES	1PDNE000	1
KPE Encrypt only	KPE-EO	1PUEE000	1,10
KPE Encrypt only (DES)	KPE-DES-EO	1PDEE000	1
KPE Decrypt only	KPE-DO	1PUDE000	1,20
KPE Decrypt only (DES)	KPE-DES-DO	1PDDE000	1
ATM Communication Key	KC	1cDNE000	1
Communication Key - Encrypt Only	KC-EO	1cDEE000	1
Communication Key - Decrypt Only	KC-DO	1cDDE000	1
ATM A-Key (IBM 3624)	KATM	1K3NE000	1
ATM Master Key (IBM 3624)	KM	1A3NE000	1
ATM Master Key (IBM 4731)	KM	1A7NE000	1
Terminal PIN Key	TPK	1PUNE000	1
Zone PIN Key	ZPK	1PUNE000	1
Data Encryption Key	KD	1DDNE000	2
Data Encryption Key - Encrypt Only	KD-EO	1DDEE000	2,16
Data Encryption Key - Decrypt Only	KD-DO	1DDDE000	2,17
Communication Key (IBM 3624 PIN Block)	KC3624	1DDNE000	2
KC3624 Encrypt Only	KC3624-EO	1DDEE000	2

Table 2-8 Working Key Headers (continued)

Type of Key	Key Name Abbreviation	Header	Variant
KC3624 Decrypt Only	KC3624-DO	1DDDE000	2
MF-MAC-MK	MAC-MK	1mFNE00m	2
Message Authentication Key	KMAC	1MDNE000	3
KMAC-Generate Only	KMAC-GO	1MDGE000	3,18
KMAC-Verify Only	KMAC-VO	1MDVE000	3,19
KC used to encrypt IBM 4731 PIN Block	KC4731	1c7NE000	3
KC4731-Encrypt Only	KC4731-EO	1c7EE000	3
KC4731-Decrypt Only	KC4731-DO	1c7DE000	3
PIN Encrypt and MAC	KC	1cDNE00B	3
American Express Card Security Code	KCSC	1mXNE000	3
KCSC-Generate Only	KCSC-GO	1mXGE000	3
KCSC-Verify Only	KCSC-VO	1mXVE000	3
CVV/CVC Key	KCVV	1CDNE000	3
KCVV-Generate Only	KCVV-GO	1CDGE000	3
KCVV-Verify Only	KCVV-VO	1CDVE000	3
Terminal Authentication Key	TAK	1MDNE000	3
Zone Authentication Key	ZAK	1MDNE000	3
PIN Verification Key	KPV	1VUNE000	4
PIN Verification Key -Generate Only	KPV-GO	1VUGE000	4
PIN Verification Key -Verify Only	KPV-VO	1VUVE000	4
SecureID Card Seed Encryption Key	KCSE	1V0NE000	4
SecureID Card Seed Encryption Key - Generate Only	KCSE-GO	1V0GE000	4
SecureID Card Seed Encryption Key - Verify Only	KCSE-VO	1V0VE000	4
Bank ID & Comparison Id. (Identikey)	BID	1VINE000	4
Bank ID & Comparison Id. (Identikey) - Generate Only	BID-GO	1VIGE000	4
Bank ID & Comparison Id. (Identikey)	BID-VO	1VIVE000	4
PIN Verification Key (IBM 3624)	KPV-3624	1V3NE000	4
PIN Verification Key (IBM 3624) - Generate Only	KPV-3624-GO	1V3GE000	4

Table 2-8 Working Key Headers (continued)

Type of Key	Key Name Abbreviation	Header	Variant
PIN Verification Key (IBM 3624) - Verify Only	KPV-3624-VO	1V3VE000	4
VISA PIN Verification Key pair	KPV-PAIR	1VVNE000	4
VISA PIN Verification Key pair - Generate Only	KPV-PAIR-GO	1VVGE000	4
VISA PIN Verification Key pair - Verify Only	KPV-PAIR-VO	1VVVE000	4
PIN Verification Key (Atalla BiLevel)	KPV-BI	1VBNE000	4
PIN Verification Key (Atalla BiLevel) - Generate Only	KPV-BI-GO	1VBGE000	4
PIN Verification Key (Atalla BiLevel)- Verify Only	KPV-BI-VO	1VBVE000	4
PIN Verification Key (NCR)	KPV-NCR	1VNNE000	4
PIN Verification Key (NCR) - Generate Only	KPV-NCR-GO	1VNGE000	4
PIN Verification Key (NCR) - Verify Only	KPV-NCR-VO	1VNVE000	4
PIN Verification Key (Atalla 2x2)	KPV-2x2	1VaNE000	4
PIN Verification Key (Atalla 2x2) - Generate Only	KPV-2x2-GO	1VaGE000	4
PIN Verification Key (Atalla 2x2) - Verify Only	KPV-2x2-VO	1VaVE000	4
MF_PAC_MK	PAC_MK	1mFNE00p	4
ATM Master Key	AMK	1ADNE000	5
Diebold Number Table Row	DNT	1ndNE000	5
Key for ePIN Offset	KOP	1PUNE000	5
Burroughs Number Table	BNT	1nuNE000	5
Initialization Vector or MAB	IV	1IDNE000	6
Key Data Header	KEY-DATA	1nUNE000	6
Decimalization/Conversion Table	DecTable	1nCNE000	6
KMAC used in Challenge/Response	KMACR	1MDNE00C	7
KMAC - Generate Only	KMACR-GO	1MDGE00C	7
KMAC - Verify Only	KMACR-VO	1MDVE00C	7
Derivation Key (KDERV, DK, KCD)	KDREV	1dDNE000	8
Master Key	KGK	1mZNE000	8
MAC Terminal Master Key	MAC-MK-SL	1mFNE00M	8
VISA Stored Value Card Master Key	VSVCMK	1mVNE000	9

Table 2-8 Working Key Headers (continued)

Type of Key	Key Name Abbreviation	Header	Variant
EMV ACC Master Key	MK	1mENE000	9
ICC Intermediate Master Key	IMK	1miNE000	9
Encrypt Only KPE	KPE-EO	1PUEE000	10
Control Vector for MAC derivation	CV-MAC	1IDNE00M	10
MAC Terminal Master Key	MAC_MK	1mFNE00M	11
Reference PIN MAC Gen-only key	PMK/MAK	1MDGE000	12
Master Key	MK	1mZNE000	13
EMV Message Auth. Master Key	KMAC-MK	1mENE00M	13
Reference PIN Encryption Key	PSK	1PUNE000	14
EMV PIN Encryption Key	KENC-MAC	1mENE00E	14
KD - Encrypt Only	ENC-KD	1DDEE000	16
KD - Decrypt Only	DEC-KD	1DDDE000	17
KMAC, generate only	GMAC	1MDGE000	18
KMAC, verify only	VMAC	1MDVE000	19
Decrypt Only KPE	KPE-DO	1PUDE000	20
Control Vector for KPE derivation	CV-PAC	1IDNE00P	20
Token Key type 1	TK	1TDNE001	21
Token Key type 2	TK	1TDNE002	22
Token Key type 3	TK	1TDNE003	23
Token Key type 4	TK	1TDNE004	24
PTK	PTK	1PUEE000	25
Terminal PAC Master Key	PAC-MK	1mFNE00P	26
Data Encryption Key	KD	1DDNE000	26
Terminal PAC Master Key	PAC-MK	1mFNE00P	27
Export KEK	Export-KEK	1KDEE000	28
Import KEK	Import-KEK	1KDDE000	29
KEK for outgoing key only	KEK-Out	1KDEE000	31
Application KEK for KD	KEK-App	1KDEE000	31

Generating Working Keys

A common use of the Network Security Processor is to protect sensitive information as it travels through an insecure network. Triple-DES encryption is typically used for this purpose. A random Triple-DES key is used to encrypt the sensitive information at the origin, and the same triple-DES key must be used at the receiving node to successfully decrypt the information. This means that both the origin and destination must share the same triple-DES key. When establishing working keys, a special purpose key called a Key Exchange Key (KEK) is created, and exchanged out-of-band; that is, it is not transmitted over the network. Once both nodes have the same KEK, they can use it to encrypt working keys for transmission between the two nodes.

Working keys such as PIN Encryption Keys (KPEs), Data Encryption and Decryption (KDs) and Message Authentication Keys (KMACs) should be system generated, this insures no one individual knows the clear-text key value. The Network Security Processor supports a generic key generation command Generate 3DES Working Key, Any Type (Command 10). The generated key is encrypted in two forms one for local storage and use (encrypted under the MFK) and one for export to the remote node (encrypted under the KEK).

In many networks, two different keys are used to protect data such as a customer PIN. The key used is determined by the direction the PIN is moving in the network. For example, a customer PIN being sent from an institution to a switch uses a PIN Encryption Key called an Aquirer's Working Key (AWK). A customer PIN sent from the switch to the institution is called an Issuer's Working Key (IWK). In this example, the institution is only encrypting PINs using the AWK, and only decrypting PINs using the IWK. Therefore the institution should define the AWK an encrypt-only PIN Encryption Key, and the IWK as a decrypt-only PIN Encryption Key. This same approach can be used when defining other types of working keys including but not limited to Data Encryption Keys, and Message Authentication Keys. Byte 3 of the working key header is used to define the mode of use for the working key.

Importing and Exporting Working Keys

The Key Exchange Key (KEK) is used to encrypt a working key that is to be imported or exported. The Network Security Processor's default security policy is no keys can be imported or exported, and separate KEKs must be used to import and export working keys. Use Option E0 to define what type of keys can be imported, use Option E1 to define what type of keys can be exported. These options, check the header of the working key to be sure only the appropriate type of working key can be imported or exported. Option E2 defines if the KEK AKB can be used to *both* import and export working keys, that is the KEK header byte 3 contains the value "N". These options are enabled using the SCA. You can use Network Security Processor Status ID (Command 9A) to determine the types of working keys allowed to be imported and exported in your Network Security Processor.

To import a working key from a remote node, it must be translated from encryption under the KEK to encryption under the MFK. If the remote node uses an Atalla-AKB Network Security Processor, see Import a Working Key in AKB Format (Command 13). You must use the SCA to obtain the AKB for the KEK.

If the working key to be imported is not in AKB format, see Import Non-AKB Formatted Working Keys (Command 11B). The KEK header bytes 0, 1, 2, 3, 5, and 7 must match that of the working key to be imported. The KEK header byte 6 must contain the value “I”. If the same KEK value is used to import several different types of working keys from a network node that does not support the AKB, you will be required to generate the appropriate KEK AKB for each type of working key. For example, if you will be importing KPEs, KCVVs, and Visa KPVs, all encrypted under the same KEK, you will need to create three different KEK-AKBs with headers 1PUNE0I0, 1CDNE0I0, and 1VVNE0I0, respectively.

To export an encrypted working key to a remote node it must be translated from encryption under the MFK to encryption under the KEK. If the remote node uses an Atalla-AKB Network Security Processor, see Export a Working Key in AKB Format (Command 11). Use the SCA to obtain the AKB for the KEK. The working key header byte 4 must contain the value “E”. If the working header byte 3 is either “E” encrypt-only, “D” decrypt-only, “G” generate-only, “V” or verify-only, the header for the working key encrypted under the KEK will have the opposite value. For example, if the working key is an encrypt-only PIN Encryption Key 1PUEE000, it will become a decrypt-only PIN Encryption Key 1PUDE000 when exported under the KEK.

If the remote node does not support an AKB formatted working key, see Export a Working Key to non-AKB Format (Command 1A). Use the SCA to obtain the AKB for the KEK. The working key header byte 4 must contain the value “E”.

Migrating the non-AKB formatted Working Key Database

If you are upgrading your Atalla Network Security Processor to support the Atalla Key Block, you will need to perform the following steps to migrate the working key database which is encrypted under variant specific forms of the Master File Key. This procedure assumes that the Atalla Network Security Processors are already members of a security association.

1. Define and send a 3-key Master File Key to the Network Security Processor-AKB.
2. At the non-AKB Network Security Processor, enter the first two key blocks of the 3-key Master File key as a Pending Master File Key.
3. At the non-AKB Network Security Processor, execute command 9E to translate the working keys from encryption under the MFK to encryption under the Pending MFK.
4. At the Network Security Processor-AKB, execute Convert Atalla DES Key to 3DES AKB Format (Command 11C). The AE response fields 1 and 2 from the 9E command executed in step 3 above, will be used as fields 3 and 4 respectively, in the Command 11C. The 21C response will contain the working key in AKB format.

The recommended migration procedure is:

1. Define a 3-key Master File Key and load it into the Network Security Processor-AKB.
2. Using the SCA, generate an AKB with this header 1KDDN0M0 for the old MFK.
3. At the Network Security Processor-AKB execute Convert Atalla DES Key to 3DES AKB Format (Command 11C).

If you are converting from another vendor's key protection method, you must use command Import Non-AKB Formatted Working Keys (Command 11B) to import the encrypted working keys.

Non-volatile Key Table

The Network Security Processor has a non-volatile key table that stores the Master File Key and Pending Master File Key. Keys stored in the non-volatile key table are maintained without external power for up to five years. Once loaded into the non-volatile key table, they cannot be extracted, transmitted, or downloaded in clear-text form. Securing the Network Security Processor involves using the SCA to either add a Network Security Processor to an existing security association or create a new security association for the Network Security Processor, and defining and sending a Master File Key to the Network Security Processor. See Network Security Processor Status Key (Command 9A) for the command syntax to determine what keys are stored in the non-volatile key table. See the *SCA Users Guide* for the procedures to load keys into the non-volatile key table.

Volatile Table

Early model Network Security Processors supported only asynchronous or bisynchronous communications at a maximum baud rate of 19,200 bits per second. Performance was limited by the communications interface. To minimize the number of characters sent in a command, a volatile table was created. The host application was able to preload keys into the table. When a specific key was needed, the index into the table was provided in the command, instead of the 16 character key cryptogram, reducing the length of the command. The benefit was better performance, however the host application became more complex as it had to manage the table. The Ethernet TCP/IP interface is fast enough such that there is no performance benefit in using the table. The Verify PIN – Diebold (Command 32), is the only command that requires the use of the volatile table. See Storing Values in the Volatile Table for more information.

Procedure to replace the current MFK with the pending MFK

All working key AKBs encrypted under the current MFK must be translated to encryption to the new MFK. You can do this manually with the SCA. A more efficient process is to follow this procedure. The new MFK must be loaded as a pending MFK. This procedure assumes you already have a current MFK in the Network Security Processor and working keys are encrypted under it:

1. Using the SCA, define and load the pending MFK into the Network Security Processor.
2. Translate all working keys from encryption under the current MFK to the Pending MFK. See Translate Working Key from Current MFK to Pending MFK (Command 9E) for the command syntax required to perform this task.
3. Replace the current MFK with the pending MFK. See Replace the Current MFK with the Pending MFK (Command 9F) for the command syntax required to perform this task.
4. Configure the host application to use the new key cryptograms generated in step 2.

Security Precautions

The Network Security Processor is only as secure as you and your procedures make it. Many attempts to obtain confidential information are performed by employees or other individuals with access to, or knowledge of, security related equipment. Here are some recommendations on keeping your Network Security Processor secure:

- Always keep production cryptographic keys secret. Key components should recorded and stored in a secure location.
- Always define production keys with multiple key components. Never let one individual have access to an entire production key.
- Never use the same key value for multiple purposes.
- Make sure that key component holders are restricted to their one key component. They should never be allowed to assume the role of another key component holder which would give them access to the entire secret key value.
- Never allow a test key to be used in the same system that has production keys.
- Before migrating a test unit into production, always ensure that all test keys have been deleted.
- Whenever possible, choose 2key-3DES (double-length) or 3key-3DES (triple-length) keys.

- Keep the SCA locked in a secure location.
- Keep the SCA smart card PINs secret, change them frequently. Make sure that SCA users are restricted from switching roles.
- Keep the Network Security Processor front bezel door locked. Store the keys in a secure location. Do not keep the keys in the locks.
- Never use the SCA Calculate AKB feature to encrypt a key that is known by a single individual. Always validate the source of the key and the business requirement, before you allow it to be entered into your system.
- Configure and secure your system such that only authorized individuals and host applications have access to the Network Security Processor.
- Only enable commands and options that you have confirmed are required by your host application, all other commands and options should be disabled in the Network Security Processor's security policy. Do not enable commands and options until you have confirmed that there is a legitimate business requirement to do so.

3

Key Management

This section contains the information on commands used to support the initialization and management of symmetric cryptographic keys in a financial interchange network. See Initializing the Financial Interchange Network for an overview. Information on key management using RSA keys is provided in Using Public Key Technology to Initialize Remote Devices.

Quick Reference

Table 3-1 identifies each command by number, name, and purpose. While the table organizes the initialization commands by category, the commands themselves are presented in numerical order.

Table 3-1 Initialization Commands

Command	Name	Purpose
Key generation commands		
10	Generate 3DES Working Key, Any Type	Generates a variety of Working Keys. The command returns the generated key in two forms: one for storing locally, encrypted under the MFK, and one for transmitting to another Atalla-AKB node, encrypted under the KEK.
1E	Generate New Initial Key for PIN Pad Using VISA DUKPT	This command supports an obsolete/proprietary technique for key loading and is not compatible with any current POS terminals. It is not required or recommended in any DUKPT environment.
38C	Derive DUKPT IPEK	This command derives the DUKPT Initial PIN Encryption Key.
39A	Generate AES or HMAC Key	Generates an 80 - 256 bit symmetric key.
Key Import and Export commands		
11	Exports a Working Key in AKB format.	Exports a Working Key. This command translates a Working Key in AKB format from encryption under the MFK, to encryption under the KEK. The exported Working Key is in AKB format, intended for another Atalla-AKB Network Security Processor.
13	Imports a Working Key in AKB format.	Imports a Working Key. This command translates a Working Key in AKB format from encryption under the KEK to encryption under the MFK.

Table 3-1 Initialization Commands (continued)

Command	Name	Purpose
1A	Export a Working Key in AKB format to Non-AKB format.	Exports a Working Key. This command translates a Working Key in AKB format from encryption under the MFK, to encryption under the KEK. The exported Working Key is not in AKB format.
9E	Translate Key using Pending MFK	Translates a Working Key from encryption under the MFK to encryption under the PMFK.
113	Translate Key between modes of DES	Translates a KEK encrypted Working Key from ECB to CBC, or CBC to ECB, mode of DES.
117	Import a TR-31 formatted key	Translates a TR-31 formatted Working Key that is encrypted under a KEK, to AKB format encrypted under the MFK.
118	Export a Working Key in TR-31 format	Translates a Working Key that is encrypted under the MFK, to TR-31 format encrypted under the KEK.
119	Import a TR-31 version A, B, or C formatted key	Translates a TR-31 version A, B, or C formatted Working Key that is encrypted under a KEK, to AKB format encrypted under the MFK.
11A	Export a Working Key in TR-31 format Version A, B, or C	Translates a Working Key that is encrypted under the MFK, to TR-31 version A, B, or C format encrypted under the KEK.
11B	Import Non-AKB Keys	Translates a Working Key that is not in AKB format, and is encrypted under a KEK, to encryption under the MFK. The imported Working Key is converted to AKB format.
11C	Convert DES Key to AKB Format	Converts 3DES keys, from non-AKB Atalla Network Security Processors, to AKB format.
ATM-key loading commands		
14	Load ATM Master Key – IBM 3624	Encrypts the ATM Master Key for downloading to IBM 3624 ATMs. The master key is not in AKB format.
14	Load ATM Master Key – IBM 4731	Encrypts the ATM Master Key for downloading to IBM 4731 ATMs. The master key is not in AKB format.
ATM-key changing commands		
15	Change ATM Communications Key – Docutel	Encrypts a Communications Key for downloading to Docutel ATMs. The Communications Key is not in AKB format.
15	Change ATM Communications Key – IBM 3624	Encrypts a Communications Key for downloading to an IBM 3624 ATM. The Communications Key is not in AKB format.
15	Change ATM Communications Key – IBM 4731	Encrypts a Communications key for downloading to an IBM 4731 ATM. The Communications Key is not in AKB format.
Generate Check Digit Command		
7E	Generate Check Digits	Generates check digits for a key encrypted under the MFK.
392	Generate AES Key Check Digits	Uses either the SHA-256 or CMAC algorithm to generate check digits.

Table 3-1 Initialization Commands (continued)

Command	Name	Purpose
Promote Pending MFK Command		
9F	Make Pending MFK Current	Replaces the current MFK with the Pending MFK.

Generate 3DES Working Key, Any Type (Command 10)

Command 10 generates a variety of 3DES Working Keys. The Working Key can be generated in two forms: one for storing locally, encrypted under the Master File Key (MFK), and optionally one for transmitting to another network node encrypted under the Key Exchange Key (KEK). If the key length field is not present, a 2key-3DES (double-length) key will be generated. This command is enabled in the Network Security Processor's default security policy.

This command can generate a 1key-3DES (single-length) key only if 6C is enabled.

Command

```
<10#Header#[Header,EMFK.E(KEK),MAC]#[Key Length#]>
```

Response

```
<20#Header,EMFK.E(WK),MAC#[Header,EKEK.E(WK),MAC]#
Working Key Check Digits#>[CRLF]
```

Calling Parameters

`10`

Field 0, the command identifier.

`Header`

Field 1, the header to be used for the generated Working Key, thus establishing its function. This field is eight bytes long and can contain printable ASCII characters. See Working Key Headers for a partial list of supported headers.

Byte 4, Exportability is not evaluated in the command processing logic. Therefore, if field 2 is present, the generated Working Key will be encrypted under the KEK, even if header byte 4 is set to a value of "N" (not exportable). If the generated Working Key should not be exportable, do not include field 2 in the command.

`[Header,EMFK.E(KEK),MAC]`

Field 2, the KEK encrypted under the MFK. This key is used to encrypt the randomly generated Working Key. This field contains a 74 byte value, or a volatile table location, or can be empty. If this field is empty, field 2 of the response will also be empty.

The length of the KEK must be equal-to or greater-than the length of the Working Key being generated. For example, if the generated Working Key is a 2key-3DES key, this field must contain either a 2key-3DES or 3key-3DES key, or a reference to a volatile table location that contains a 2key-3DES or 3key-3DES key. Option 6C must be enabled if this field will contain a 1key-3DES (single-length) key.

The following headers are supported: 1KDEE000 and 1KDEN000. If option E2 is enabled, these headers are supported: 1KDEE000, 1KDEN000, 1KDNE000 and 1KDNN000. If option E3 is enabled, these headers are supported: 1KDEE000, 1KDEN000, 1ADEE000 and 1ADEN000. When both options E2 and E3 are enabled, these headers are supported: 1KDEE000, 1KDEN000, 1KDNE000, 1KDNN000, 1ADEE000, 1ADEN000, 1ADNE000 and 1ADNN000.

`[Key Length#]`

Field 3, an optional field, used to specify the length of the generated Working Key. If this field is not included, a 2key-3DES (double-length) Working Key will be generated.

Value	Key Length
S	1key-3DES key (single-length)
D	2key-3DES (double-length)
T	3key-3DES (triple-length)

Table 3-2 Command 10: Generate Working Key, Any Type

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	10
1	Header	8	printable ASCII
2	[Header,E$_{MFK.E}$(KEK),MAC]*	0, 74	printable ASCII
3	[Key Length]	0, 1	S, D, T

* Can be a volatile table location.

Responding Parameters

`20`

Field 0, the response identifier.

Header,E$_{MFK.E}$(Working Key),MAC

Field 1, the Working Key encrypted under the MFK. The host application stores this key block on its local database for subsequent use. This field contains a 74 byte value.

Header,E$_{KEK.E}$(Working Key),MAC

Field 2, the Working Key encrypted using the KEK specified in field one of the command. The host application transmits this key block to the network node. This field contains a 74 byte value, or if field 2 of the command is empty, this field will be empty.

If Byte 3 of the Working Key header defined in field 1 of the command contains either E, D, G, or V, the **opposite** value will be used for byte 3 of the header in this field. For example, if an Encrypt-Only KPE header is defined in field 1 with a header value of 1PUEE000, the Working Key header in this field will be 1PUDE000, indicating this key can only be used to decrypt PINs. The generate and verify modes work the same way. For example, if a KMAC Working Key is defined as Verify-Only 1MDVE000 in field 1 of the command, the header in this field will be 1MDGE000, indicating that this key can only be used to generate MACs.

`Working Key Check Digits`

Field 3, check digits; the first four digits that result from encrypting zeros using the Working Key. If option 88 is enabled, this field will contain the six check digits.

Table 3-3 Response 20: Generate Working Key, Any Type

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	20
1	Header,$E_{MFK.E}$(Working Key),MAC	74	printable ASCII
2	[Header,$E_{KEK.E}$(Working Key),MAC]	0, 74	printable ASCII
3	Working Key Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- Generate the KEK and obtain the key block of it encrypted under the MFK. Store this value in your host application database.
- This command uses the Network Security Processor's Deterministic Random Bit Generator (DRBG) to generate odd-parity keys.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

The 2key-3DES KEK used in these examples is:
0123 4567 89AB CDEF FEDC BA98 7654 3210, check digits = 08D7.

The KEK in AKB format:
1KDEE000,00B12295FA353767B3E2F92C51F8C4799EE072B479C8C889,815074BF95E27120

This command generates a random Working Key, therefore your results will not match these examples.

Generate a 1key-3DES PIN Encryption Key

The command looks like this:

```
<10#1PUNE000#1KDEE000,00B12295FA353767B3E2F92C51F8C4799EE072B479C8C
889,815074BF95E27120#S#>
```

The Network Security Processor returns a response similar to this:

```
<20#1PUNE000,CAB52BB0EE74A388D602F3426B7C3CC04E59917FBD6BA192,F4837
093CBD2C05E#1PUNE000,8FE933EF0A53BA6D4847CCD897198DAB1AD22AC21FC589
B1,D3C169E0E864E6B3#BE9F#>
```

Generate a 2key-3DES Encrypt-Only PIN Encryption Key

The command looks like this:

```
<10#1PUEE000#1KDEE000,00B12295FA353767B3E2F92C51F8C4799EE072B479C8C
889,815074BF95E27120#D#>
```

The Network Security Processor returns a response similar to this:

```
<20#1PUEE000,7B8A26B5FCB083AED95AE5D5C6BC2E3F997ADC6491836767,51A28
5CDAC1E378C#1PUDE000,262EBCE518B41DA681C6BD6CA9C6A8C4DD1BFCB41122A0
5F,B3067437DA538EE4#2BA7#>
```

Generate a 3key-3DES PIN Encryption Key

```
<10#1PUNE000#1KDEE000,00B12295FA353767B3E2F92C51F8C4799EE072B479C8C
889,815074BF95E27120#T#>
```

The Network Security Processor returns the following response:

```
<00#070310#>
```

because the KEK is only a 2key-3DES key. To generate a 3key-3DES Working Key, the KEK must be a 3key-3DES key.

Generate a 2key-3DES PIN Verification Key (that is not encrypted under a KEK)

The command looks like this:

```
<10#1VUNE000##D#>
```

The Network Security Processor returns a response similar to this:

```
<20#1VUNE000,499D7190CFCC3045CDD1E6D26D472A884BD0769C4DBE71F2,73A64
E0E0F37391B##08D7#>
```

Export a Working Key in AKB Format (Command 11)

Command 11 translates a Working Key of any type, from encryption under the MFK to encryption under the KEK. Use this command to export a Working Key to another node that uses Atalla Network Security Processors that support the Atalla Key Block. Your node and the remote node must have the same KEK value.

Option E1 must be enabled with the Byte 1, Key Usage value of the Working Key to be exported. For example, if a PIN Encryption Key (header 1PUNE000) is to be exported, option E1 must contain the letter “P”.

This command can export a 1key-3DES (single-length) key only if 6C is enabled.

This command is not enabled in the Network Security Processor’s default security policy. To use this command, you must add it to the Network Security Processor’s security policy.

Command

<11#Reserved#Header,$E_{MFK.E}$(KEK),MAC#Header,$E_{MFK.E}$(WK),MAC#>

Response

<21#Header,$E_{KEK.E}$(WK),MAC#Working Key Check Digits#>[CRLF]

Calling Parameters

`11`

Field 0, the command identifier.

`Reserved`

Field 1, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

Header,$E_{MFK.E}$(KEK),MAC

Field 2, the KEK encrypted under the MFK. The KEK is used to encrypt the Working Key. This field contains a 74 byte value, or a volatile table location.

The length of the KEK must be equal-to or greater-than the length of the Working Key being exported. For example, if the Working Key is a 2key-3DES key, this field must contain either a 2key-3DES or 3key-3DES key, or a reference to a volatile table location that contains a 2key-3DES or 3key-3DES key. The following headers are supported: 1KDEE000 and 1KDEN000. If option E2 is enabled, these additional headers are supported: 1KDNE000 and 1KDNN000.

`Header,E`$_{MFK.E}$`(WK),MAC`

Field 3, the Working Key encrypted under the MFK. The Working Key can have any valid header, except that header byte 4 cannot contain the value “N”. See Working Key Headers for a partial list of supported headers. This field contains a 74 byte value, or a volatile table location.

Table 3-4 Command 11: Export a Working Key in AKB Format

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	11
1	Reserved	0, 1, 2	0 - 31
2	Header,E$_{MFK.E}$(KEK),MAC*	74	printable ASCII
3	Header,E$_{MFK.E}$(WK),MAC*	74	printable ASCII

* Can be a volatile table location.

Responding Parameters

`21`

Response identifier.

`Header,E`$_{KEK.E}$`(WK),MAC`

Field 1, the Working Key encrypted under the KEK. This field contains a 74 byte value.

If Byte 3 of the Working Key header, defined in field 3 of the command, contains either E, D, G, or V, the **opposite** value will be used for byte 3 of the header in this field. For example, if an Encrypt-Only KPE header is defined in field 3 with a header value of 1PUEE000, the Working Key header in this field will be 1PUDE000, indicating that this key can only be used to decrypt PINs. The generate and verify modes work the same way, for example if a KMAC Working Key is defined as Verify-Only 1MDVE000 in field 3 of the command, the header in this field will be 1MDGE000, indicating that this key can only be used to generate MACs.

`Working Key Check Digits`

Field 2, check digits; the first four digits that result from encrypting zeros using the Working Key. If option 88 is enabled, this field will contain the six check digits.

Table 3-5 Response 21: Export a Working Key in AKB Format

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	21

Table 3-5 Response 21: Export a Working Key in AKB Format (continued)

Field	Contents	Length (bytes)	Legal Characters
1	Header,$E_{KEK.E}$(WK),MAC	74	printable ASCII
2	Working Key Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- This command is used for transmitting a Working Key from one network node to another. Both nodes must use Atalla Network Security Processors that support the Atalla Key Block and have the same KEK.
- Generate the Working Key to be transmitted, and the appropriate header for the KEK required to transmit this Working Key.
- Option E1 must contain the key type being exported.
- The key to be exported must have an AKB header whose first 6 bytes match one of the allowed AKB headers. Use command <9A#HEADERS#> to obtain a list of allowed AKB headers.
- Do not use this command to export HMAC-SHA1 or HMAC-SHA256 keys that are longer than 176-bits.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Export a PIN Encryption Key (KPE)

- The 2key-3DES KEK used in this example is:
  0123 4567 89AB CDEF FEDC BA98 7654 3210, check digits = 08D7
  The KEK in AKB format:
  1KDEE000,00B12295FA353767B3E2F92C51F8C4799EE072B479C8C889,815074BF95E27120
- The 2key-3DES KPE used in this example is:
  4321 8765 4321 8765 FEDC BA98 FEDC 8765, check digits = C2BA
  The KPE in AKB format:
  1PUNE000,D57B60F2B33E1E09455043445C18B8E3F4377A860FB6C6B9,29D9CE150C987366

The command looks like this:

```
<11##1KDEE000,00B12295FA353767B3E2F92C51F8C4799EE072B479C8C889,8150
74BF95E27120#1PUNE000,D57B60F2B33E1E09455043445C18B8E3F4377A860FB6C
```

```
6B9,29D9CE150C987366#>
```

The Network Security Processor returns the following response:

```
<21#1PUNE000,8E032A21466685B81D78BDEB280430D06A631F2092AC87F1,ABCA4
0663880916B#C2BA#>
```

Import a Working Key in AKB Format (Command 13)

Command 13 translates a Working Key from encryption under a KEK to encryption under the MFK. Use this command to import a Working Key from another node that uses Atalla Network Security Processors that support the Atalla Key Block. Your node and the remote node must have the same KEK.

Option E0 must be enabled with the Byte 1, Key Usage value of the Working Key being imported. For example, if a PIN Encryption Key (header 1PUNE000) is to be imported, option E0 must contain the letter “P”.

This command can import a 1key-3DES (single-length) key only if 6C is enabled.

This command is not enabled in the Network Security Processor’s default security policy. To use this command, you must add it to the Network Security Processor’s security policy.

Command

<13#Reserved#Header,$E_{MFK.E}$(KEK),MAC#Header,$E_{KEK.E}$(WK),MAC#>

Response

<23#Header,$E_{MFK.E}$(WK),MAC#Working Key Check Digits#>[CRLF]

Calling Parameters

`13`

Field 0, the command identifier.

`Reserved`

Field 1, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

Header,$E_{MFK.E}$(KEK),MAC

Field 2, the KEK encrypted under the MFK. This key is used to protect the Working Key during a key exchange with the transmitting node. This field contains a 74 byte value or a volatile table location.

The length of the KEK must be equal-to or greater-than the length of the Working Key being imported. For example, if the Working Key is a 2key-3DES key, this field must contain either a 2key-3DES or 3key-3DES key, or a reference to a volatile table location that contains a 2key-3DES or 3key-3DES key. The following headers are supported: 1KDDE000 and 1KDDN000. If option E2 is enabled, these additional headers are supported: 1KDNE000 and 1KDNN000.

`Header,E`$_{KEK.E}$`(WK),MAC`

Field 3, the Working Key encrypted under the KEK. The Working Key can have any valid header. See Working Key Headers for a partial list of supported headers. This field contains a 74 byte value. In versions 1.14 and above, the header byte 4 can be any valid character when header byte 6 contains the capital letter "I".

Table 3-6 Command 13: Import a Working Key in AKB Format

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	13
1	Reserved	0, 1, 2	0 - 31
2	Header,E$_{MFK.E}$(KEK),MAC*	74	printable ASCII
3	Header,E$_{KEK.E}$(WK),MAC	74	printable ASCII

* Can be a volatile table location.

Responding Parameters

`23`

Field 0, the response identifier.

`Header,E`$_{MFK.E}$`(WK),MAC`

Field 1, the Working Key encrypted under the MFK. This field contains a 74 byte value.

`Working Key Check Digits`

Field 2, check digits; the first four digits that result from encrypting zeros using the Working Key. If option 88 is enabled, this field will contain six check digits.

Table 3-7 Response 23: Import a Working Key in AKB Format

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	23
1	Header,E$_{MFK.E}$(WK),MAC	74	printable ASCII
2	Working Key Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- This command is used to receive a Working Key that has been transmitted from another node that uses Atalla Network Security Processors that support the AKB format.

- Generate the Key Exchange Key.

- Option E0 must specify the key byte usage for the key type to be imported.
- Do not use this command to import HMAC-SHA1 or HMAC-SHA256 keys that are longer than 176-bits.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate a PIN Encryption Key

- The 2key-3DES KEK used in this example is:
  0123 4567 89AB CDEF FEDC BA98 7654 3210, check digits = 08D7
  The KEK in AKB format, encrypted under the MFK:
  1KDDE000,71D2B6C6C47EB96240DF41F6432047F20958029A5018D141,B7F73C6625A54B26
- The 2key-3DES KPE used in this example is:
  4321 8765 4321 8765 FEDC BA98 FEDC 8765, check digits = C2BA
  The KPE in AKB format, encrypted under the KEK:
  1PUNE000,8E032A21466685B81D78BDEB280430D06A631F2092AC87F1,ABCA40663880916B

The command looks like this:

```
<13##1KDDE000,71D2B6C6C47EB96240DF41F6432047F20958029A5018D141,B7F7
3C6625A54B26#1PUNE000,8E032A21466685B81D78BDEB280430D06A631F2092AC8
7F1,ABCA40663880916B#>
```

The Network Security Processor returns the following response:

```
<23#1PUNE000,481C952CE8C7347E1196896FE57526A3C461632195D59963,9D55D
8F8453922D8#C2BA#>
```

Load ATM Master Key – IBM 3624 (Command 14)

Command 14 – IBM 3624, encrypts the ATM master key for downloading to an IBM 3624 ATM.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

Command

```
<14#3#Header,E_MFK.E(MK),MAC#Header,E_MFK.E(K1),MAC#
Header,E_MFK.E(K2),MAC#Message#>
```

Response

```
<24#3#IBM 3624 Message#>[CRLF]
```

Calling Parameters

`14`

Field 0, the command identifier.

`3`

Field 1, the ATM identifier; IBM 3624.

Header,$E_{MFK.E}$(MK),MAC

Field 2, the ATM Master Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1ADNE000 and 1ADNN000.

Header,$E_{MFK.E}$(K1),MAC

Field 3, the ATM A key (K1) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location.The following headers are supported: 1K3NE000 and 1K3NN000.

Header,$E_{MFK.E}$(K2),MAC

Field 4, the ATM Communications Key (K2) encrypted under the MFK. This field contains a 74-byte hexadecimal value, or a volatile table location. The following header is are supported: 1DDNE000, 1DDNN000, 1DDEE000, and 1DDEN000.

`Message`

Field 5, bytes five to eight in the IBM 3624 request message, represented as eight hexadecimal characters.

Table 3-8 Command 14: Load ATM Master Key – IBM 3624

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	2	14
1	ATM identifier (IBM 3624)	1	3
2	Header,$E_{MFK.E}$(MK),MAC*	74	printable ASCII
3	Header,$E_{MFK.E}$(K1),MAC*	74	printable ASCII
4	Header,$E_{MFK.E}$(K2),MAC*	74	printable ASCII
5	Message	8	0 - 9, A - F

* Can be a volatile table location.

Responding Parameters

`24`

Field 0, the response identifier.

`3`

Field 1, the ATM identifier; IBM 3624.

`IBM 3624 Message`

Field 2, the result of the partial double-encryption process defined in IBM key management. This result is formed using the following steps.

1. First, the ATM master key is encrypted using the ATM A key. This is denoted as E_{K1}(KM), where K1 is the ATM A key and KM is the ATM master key to be downloaded.

2. Let L4 represent the leftmost 4 bytes of E_{K1}(KM) and let R4 represent the rightmost 4 bytes of E_{K1}(KM). Each value – L4 and R4 – is 8 hexadecimal characters long.

   The four variable bytes in Field 5 of the command are concatenated to the left of L4, forming an eight byte (that is, 16 hexadecimal character) field, denoted as follows.

   ```
   [(4 Var Bytes) || L4]
   ```

   This field is then encrypted using the ATM communication key, K2. The result of this encryption is denoted as follows.

```
E_K2[(4 Var Bytes) || L4]
```

3. R4 is concatenated to the right of the encrypted result of step two, denoted as follows.

```
E_K2[(4 Var Bytes) || L4] || R4
```

This result is the 12 byte (that is, 24 hexadecimal character) field that is sent to the IBM 3624 ATM.

Table 3-9 Response 24: Load ATM Master Key – IBM 3624

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	24
1	ATM identifier (IBM 3624)	1	3
2	IBM 3624 Message	24	0 - 9, A - F

Usage Notes

- Generate the ATM Master Key (MK), ATM A key (K1) and the ATM Communications Key (K2).

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Load an ATM master key

- Clear-text Master Key: 1111 1111 1111 1111
  The Master Key in AKB format:
  1ADNE000,30E6A07C9E4C487EB08F46AF01E64CAED0992A87B336AC61,A66804B787FC9CDD

- Clear-text ATM A Key (K1): 2222 2222 2222 2222
  The ATM A Key in AKB format:
  1K3NE000,84450846E3E1C9A3B0B35F9BFADEA29D5694D31E8E8227B7,42A93B9E88DCE2B2

- Clear-text ATM Communications Key (K2): 3333 3333 3333 3333
  The ATM Communications Key in AKB format:
  1DDNE000,09FFCD303017DA193AE10630F7A97FB04EAC17E4CA01A1FC,966113F44A28E527

- Message: 56789ABC.

The command looks like this:

```
<14#3#1ADNE000,30A334A2FE749CB53E376FCF3CB32BDA3FC3F90C6D082A4A,DA9
F8D4941C49AB2#1K3NE000,84450846E3E1C9A3B0B35F9BFADEA29D5694D31E8E82
27B7,42A93B9E88DCE2B2#1DDNE000,09FFCD303017DA193AE10630F7A97FB04EAC
17E4CA01A1FC,966113F44A28E527#>
```

The Network Security Processor returns the following response:

```
<24#3#2B41AE49E5C8E28F811DA672#>
```

Load ATM Master Key – IBM 4731 (Command 14)

Command 14 – IBM 4731, generates an IBM 4731 ATM master key (KM).

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

Command

```
<14#4#Header,EMFK.E(KX),MAC#Header,EMFK.E(KI),MAC#Message#
[Key Length#]>
```

Response

```
<24#4#Header,EMFK.E(KM),MAC#Header,EMFK.E(EKI(KM)),MAC#
Header,EKX.E(EKI(KM)),MAC#
E(EInitial Master Key(ATM Master Key))(message type/date)#
Exchange Key Check Digits#Initial Master Key Check Digits#
ATM Master Key Check Digits#
ATM Master Keyencrypted under Initial Master Key Check Digits#>
[CRLF]
```

Calling Parameters

`14`

Field 0, the command identifier.

`4`

Field 1, the ATM identifier; IBM 4731.

Header,$E_{MFK.E}$(KX),MAC

Field 2, the Exchange Key encrypted under the MFK. This key is used to encrypt the cryptogram of the generated ATM master key encrypted under the Initial Master Key. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1KDNE000, 1KDNN000, 1KDEE000, and 1KDEN000.

Header,$E_{MFK.E}$(KI),MAC

Field 3, the Initial Master Key encrypted under the MFK. This key is used to encrypt the generated ATM Master Key. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1KDNE000, 1KDNN000, 1KDEE000, and 1KDEN000.

`Message`

Field 4, the message type/date in binary form. This field contains an 8 byte binary value, where each character represents one byte.

`[Key Length#]`

Field 5, length of the generated IBM 4731 ATM master key. This is an optional field. If used, it can be one byte, containing the numbers 1 (1key-3DES key) or 2 (2key-3DES key). If this field is not present, a 1key-3DES key will be generated.

Table 3-10 Command 14: Load ATM Master Key – IBM 4731

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	14
1	ATM identifier (IBM 4731)	1	4
2	Header,$E_{MFK.E}$(KX),MAC*	74	printable ASCII
3	Header,$E_{MFK.E}$(KI),MAC*	74	printable ASCII
4	Message	8	Binary
5	[Key Length#]	0, 1	empty, 1 or 2

* Can be a volatile table location.

Responding Parameters

`24`

Field 0, the response identifier.

`4`

Field 1, the ATM identifier; IBM 4731.

Header,$E_{MFK.E}$(KM),MAC

Field 2, the generated ATM Master Key encrypted under the MFK. This field contains a 74 byte value.

Header,$E_{MFK.E}$(E_{KI}(KM)),MAC

Field 3, the generated ATM Master Key encrypted under the Initial Master Key. This cryptogram is then encrypted under the MFK. This field contains a 74 byte value.

Header,$E_{KX.E}$(E_{KI}(KM)),MAC

Field 4, the generated ATM Master Key encrypted under the Initial Master Key. This cryptogram is then encrypted under the Exchange Key. This field contains a 74 byte value.

`E(E`$_{\text{Initial Master Key}}$`(ATM Master Key))(message type/date)`

Field 5, the generated ATM Master Key encrypted under the Initial Master Key. This cryptogram is then used to encrypt the message type/date. This field contains a 16 byte hexadecimal value.

`Exchange Key Check Digits`

Field 6, check digits; the first four digits that result from encrypting zeros using the Exchange Key. If option 88 is enabled, this field will contain six check digits.

`Initial Master Key Check Digits`

Field 7, check digits; the first four digits that result from encrypting zeros using the Initial Master Key. If option 88 is enabled, this field will contain six check digits.

`Master Key Check Digits`

Field 8, check digits; the first four digits that result from encrypting zeros using ATM Master Key. If option 88 is enabled, this field will contain six check digits.

`Cryptogram Check Digits`

Field 9, check digits; the first four digits that result from encrypting zeros using the cryptogram of the ATM Master Key encrypted under the Initial Master Key. If option 88 is enabled, this field will contain six check digits.

Table 3-11 Response 24: Load ATM Master Key – IBM 4731

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Response identifier	2	24
1	ATM identifier (IBM 4731)	1	4
2	Header,$E_{MFK.E}$(KM),MAC	74	printable ASCII
3	Header,$E_{MFK.E}$(E_{KI}(KM)),MAC	74	printable ASCII
4	Header,$E_{KX.E}$(E_{KI}(KM)),MAC	74	printable ASCII
5	E($E_{\text{Initial Master Key}}$(ATM Master Key))(message type/date)	16	0 - 9, A - F
6	Exchange Key Check Digits	4 or 6	0 - 9, A - F
7	Initial Master Key Check Digits	4 or 6	0 - 9, A - F
8	ATM Master Key Check Digits	4 or 6	0 - 9, A - F
9	Cryptogram Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- Generate the Initial Master Key (KI) and the KEK (KX).

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

This example generates a random ATM Master Key your result will be different.

Generate an IBM 4731 ATM Master Key

- Clear-text Exchange Key (KX): 0123 4567 89AB CDEF
  The Exchange Key in AKB format:
  1KDNE000,7BC98D21224A2C853A3C48A2A09AF7905CDD901CD903FAE1,FBBDF27401D4CCD8
- Clear-text Initial Master key (KI): 1111 2222 3333 4444
  The Initial Master Key in AKB format:
  1KDNE000,5872409C4F44A5B161D37ED2D2F13F8A4C353D985F2D46E0,24CADA741B2754C3
- Message: 01234567

The command looks like this:

```
<14#4#1KDNE000,7BC98D21224A2C853A3C48A2A09AF7905CDD901CD903FAE1,FBB
DF27401D4CCD8#1KDNE000,5872409C4F44A5B161D37ED2D2F13F8A4C353D985F2D
46E0,24CADA741B2754C3#01234567#>
```

The Network Security Processor returns a response similar to this:

```
<24#4#1KDNE000,742C3CCD01F290B24A274591A4B547CDF1D162C348D96E52,D44D
ED381F71FB7A#1DDNE000,331BCC5079164766F23D65F8BDF1B2B8921B67D7CE5A15
7E,E641E7CE2E37D3CA#1DDNE000,57551BFCEF5173C27021241C9302FC1D4D56E49
091FB1285,CCFA62D9283BD08A#97A99756709F90E8#D5D4#F7A6#B9EF#24C6#>
```

Change ATM Communications Key – Docutel (Command 15)

Command 15 – Docutel, encrypts a Communications Key for downloading to a Docutel ATM.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

Command

<15#2#Header,$E_{MFK.E}$(KC),MAC#Header,$E_{MFK.E}$(KM),MAC#>

Response

<25#2#$E_{Master\ Key}$(Communications Key)#
Communications Key Check Digits#>[CRLF]

Calling Parameters

`15`

Field 0, the command identifier.

`2`

Field 1, the ATM identifier; in this command, Docutel.

Header,$E_{MFK.E}$(KC),MAC

Field 2, the Communications Key encrypted under the MFK. This is the key to be downloaded to the ATM. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1cDNE000, 1cDNN000, 1cDEE000, 1cDEN000, 1cDDE000, and 1cDDN000.

Header,$E_{MFK.E}$(KM),MAC

Field 3, the ATM Master Key encrypted under the MFK. This key is used to encrypt the new Communications Key. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1ADNE000 and 1ADNN000.

Table 3-12 Command 15: Change ATM Communications Key – Docutel

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	15
1	ATM identifier (Docutel)	1	2

Table 3-12 Command 15: Change ATM Communications Key – Docutel (continued)

Field	Contents	Length (bytes)	Legal Characters
2	Header,$E_{MFK.E}$(KC),MAC*	74	printable ASCII
3	Header,$E_{MFK.E}$(KM),MAC*	74	printable ASCII

* Can be a volatile table location.

Responding Parameters

`25`

Field 0, the response identifier.

`2`

Field 1, the ATM identifier; in this command, Docutel.

`E`$_{\texttt{Master Key}}$`(Communications Key)`

Field 2, the Communications Key encrypted under the ATM master key. This field contains a 16 byte hexadecimal value.

`Communications Key Check Digits`

Field 3, the result of encrypting 0123 4567 89AB CDEF using the Communications Key. This field contains a 16 byte hexadecimal value.

Table 3-13 Response 25: Change ATM Communications Key – Docutel

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	25
1	ATM identifier (Docutel)	1	2
2	E_{KM}(KC)	16	0 - 9, A - F
3	E_{KC}(0123456789ABCDEF)	16	0 - 9, A - F

Usage Notes

- The Communications Key on a Docutel ATM is used to encrypt PINs; therefore, although the term Communications Key is used for this key, it is supported in the Atalla key management scheme as a PIN Encryption Key (KPE).
- Generate the ATM master key.
- Generate the new Communications Key (KC). The Communications Key can be randomly generated using Command 10.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Change the ATM Communications Key

- Clear-text Communications Key: 1111 1111 1111 1111
  The Communications Key in AKB format:
  1cDNE000,513AC567D16856035019FA23000EBB31E5A2F7D3211204AC,7410337A13F34146

- Clear-text ATM Master Key: 2222 2222 2222 2222
  The ATM Master Key in AKB format:
  1ADNE000,D64FD16F5811E399A0FD379B5EC338D6917E1491D022EA0F,3B545D817E6523D4

The command looks like this:

```
<15#2#1cDNE000,513AC567D16856035019FA23000EBB31E5A2F7D3211204AC,741
0337A13F34146#1ADNE000,D64FD16F5811E399A0FD379B5EC338D6917E1491D022
EA0F,3B545D817E6523D4#>
```

The Network Security Processor returns the following response:

```
<25#2#08024FCF811DA672#8A5AE1F81AB8F2DD#>
```

Change ATM Communications Key – IBM 3624 (Command 15)

Command 15 – IBM 3624, encrypts a Communications Key for downloading to an IBM 3624 ATM.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

Command

```
<15#3#Header,EMFK.E(KC),MAC#Header,EMFK.E(P),MAC#
Header,EMFK.E(KC-1),MAC#Reserved#>
```

Response

```
<25#3#IBM 3624 Message#>[CRLF]
```

Calling Parameters

`15`

Field 0, the command identifier.

`3`

Field 1, the ATM identifier; IBM 3624.

Header,$E_{MFK.E}$(KC),MAC

Field 2, the Communications Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1DDNE000, 1DDNN000, 1DDEE000, 1DDEN000, 1DDDE000, and 1DDDN000.

Header,$E_{MFK.E}$(P),MAC

Field 3, either the ATM Master Key (KM) encrypted under the MFK, or the old Communications Key (KC-1) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The contents of this field depend on the value of Field 6. The following headers are supported: 1A3NE000, 1A3NN000, 1DDNE000, 1DDNN000, 1DDEE000 and 1DDEN000.

Header,$E_{MFK.E}$(KC-1),MAC

Field 4, the old Communications Key (KC-1) encrypted under the MFK. This field contains a 74-byte hexadecimal value, or a volatile table location. The following headers are supported: 1C3NE000, 1DDNE000, 1DDNN000, 1DDEE000 and 1DDEN000.

`Message`

Field 5, bytes five to eight of the IBM 3624 request message, represented as eight hexadecimal characters.

`Reserved`

Field 6, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

Table 3-14 Command 15: Change ATM Communications Key – IBM 3624

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	15
1	ATM identifier (IBM 3624)	1	3
2	Header,$E_{MFK.E}$(KC),MAC*	74	printable ASCII
3	Header,$E_{MFK.E}$(P),MAC*	74	printable ASCII
4	Header,$E_{MFK.E}$(KC-1),MAC*	74	printable ASCII
5	Message	8	0 - 9, A - F
6	Reserved	0, 1, 2	any

* Can be a volatile table location.

Responding Parameters

`25`

Field 0, the response identifier.

`3`

Field 1, the ATM identifier; IBM 3624.

`IBM 3624 Message`

Field 2, the result of the partial double encryption process defined in IBM key management. This is formed using the following steps.

1. First, the Communications Key, KC, is encrypted using the appropriate key, P. The result, Ep(KC), is divided into two parts, L4 and R4.

2. The four variable bytes of Field 5 in the command are concatenated to the left of L4, denoted as follows.

   ```
   (4 Var Bytes) || L4
   ```

   The result is then encrypted using the old Communications Key, KC-1, denoted as follows.

```
EKC-1[(4 Var Bytes) || L4]
```

3. R4 is then concatenated to the right of this encrypted result to obtain KC-1, denoted as follows.

```
EKC-1[(4 Var Bytes) || L4] || R4
```

This result is the 12 byte (24 hexadecimal character) field that is sent to the IBM 3624 ATM.

Table 3-15 Response 25: Change ATM Communications Key – IBM 3624

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	25
1	ATM identifier (IBM 3624)	1	3
2	IBM 3624 message	24	0 - 9, A - F

Usage Notes

- Generate the encryption key to be downloaded.
- Generate both the old and new Communications Keys.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Change the ATM Communications Key

- Clear-text Communications Key: 0123 4567 89AB CDEF
  The Communications Key in AKB format:
  1DDNE000,4509C24B1C13C3798D30B3AEAEB65DE1CEFB3E831C522F7F,1CE66C3D258AF755
- Clear-text ATM Master Key: 3333 3333 3333 3333
  The ATM Master Key in AKB format:
  1A3NE000,BA66B5AECB4A24EDD6AB7832DDB06A7BF0051CEAB91EF83C,E6EF050E065A7D6E
- Clear-text old Communications Key (KC-1): 0123456789ABCDEF
  The old Communications Key (KC-1) in AKB format:
  1DDNE000,4509C24B1C13C3798D30B3AEAEB65DE1CEFB3E831C522F7F,1CE66C3D258AF755

- IBM 3624 request message: 12345678

The command looks like this:

```
<15#3#1DDNE000,4509C24B1C13C3798D30B3AEAEB65DE1CEFB3E831C522F7F,1CE
66C3D258AF755#1A3NE000,BA66B5AECB4A24EDD6AB7832DDB06A7BF0051CEAB91E
F83C,E6EF050E065A7D6E#1DDNE000,4509C24B1C13C3798D30B3AEAEB65DE1CEFB
3E831C522F7F,1CE66C3D258AF755#12345678##>
```

The Network Security Processor returns the following response:

```
<25#3#11581BCF707F368E06463E6C#>
```

Change ATM Communications Key – IBM 4731 (Command 15)

Command 15 – IBM 4731, generates a random communication key (KC) for downloading to an IBM 4731 ATM.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

Command

```
<15#4#Header,EMFK.E(KX),MAC#Message#>
```

Response

```
<25#4#Header,EMFK.E(KC),MAC#EKX(KC)#EKC(message)#KC Check Digits#>
[CRLF]
```

Calling Parameters

`15`

Field 0, the command identifier.

`4`

Field 1, the ATM identifier; IBM 4731.

`Header,EMFK.E(KX),MAC`

Field 2, the Exchange Key encrypted under the MFK. This key is used to encrypt the generated Communications Key. This field contains a 74 byte value. The following headers are supported: 1KDNE000 and 1KDEE000.

`Message`

Field 3, the message type/date in binary form. This field contains an 8 byte binary value.

Table 3-16 Command 15: Change ATM Communications Key – IBM 4731

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	15
1	ATM identifier (IBM 4731)	1	4
2	Header,$E_{MFK.E}$(KX),MAC*	74	printable ASCII
3	Message	8	any (binary)

* Can be a volatile table location.

Responding Parameters

`25`

Field 0, the response identifier.

`4`

Field 1, the ATM identifier; IBM 4731.

Header,$E_{MFK.E}$(KC),MAC

Field 2, the generated Communications Key (KC) encrypted under the MFK. This field contains a 74 byte value.

E_{KX}(KC)

Field 3, the generated Communications Key (KC) encrypted under the Exchange Key. This field contains a 16 byte hexadecimal value.

E_{KC}(message)

Field 4, the message type/date encrypted under the generated Communications Key (KC). This field contains a 16 byte hexadecimal value.

`Communications Key Check Digits`

Field 5, check digits; the first four digits that result from encrypting zeros using the Communications Key. If option 88 is enabled, this field will contain six check digits.

Table 3-17 Response 25: Change ATM Communications Key – IBM 4731

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	25
1	ATM identifier (IBM 4731)	1	4
2	Header,$E_{MFK.E}$(KC),MAC	74	printable ASCII
3	$E_{\text{Exchange Key}}$(Communications Key)	16	0 - 9, A - F
4	$E_{\text{Communications Key}}$(Message)	16	0 - 9, A - F
5	Communications Key Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- Generate the Exchange Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

This command generates a random Communications Key, your test results will be different.

Change the ATM Communications Key

- Clear-text Exchange Key (KX): 0123 4567 89AB CDEF
  The Exchange Key in AKB format:
  1KDNE000,7BC98D21224A2C853A3C48A2A09AF7905CDD901CD903FAE1,FBBDF27401D4CCD8

- Message: 01234567

The command looks like this:

```
<15#4#1KDNE000,7BC98D21224A2C853A3C48A2A09AF7905CDD901CD903FAE1,FBB
DF27401D4CCD8#01234567#>
```

The Network Security Processor returns a response similar to this:

```
<25#4#1c7NE000,CDBDE2F596449C8D0E909218BCBEC47757F5D11508334E2F,985
E755F059CB54E#C9E13778E0AADFB6#ECFC979E884DC6F5#1D8F#>
```

Export a Working Key to non-AKB Format (Command 1A)

Command 1A translates a Working Key from encryption under the MFK, to encryption under a KEK. Use this command to export a Working Key to another node that does not use Atalla Network Security Processors that support the Atalla Key Block. Your node and the remote node must have the same KEK value.

Option E1 must be enabled with the Byte 1, Key Usage value of the Working Key to be exported. For example, if a PIN Encryption Key (header 1PUNE000) is to be exported, option E1 must contain the letter “P”.

This command can export a 1key-3DES (single-length) key only if 6C is enabled.

This command has a high security exposure, it is not enabled in the Network Security Processor’s default security policy. To use this command, you must add it to the Network Security Processor’s security policy.

Command

<1A#Variant#Header,$E_{MFK.E}$(KEK),MAC#Header,$E_{MFK.E}$(WK),MAC#>

Response

<2A#$E_{Key\ Exchange\ Key.V}$(Working Key)#Working Key Check Digits#>[CRLF]

Calling Parameters

`1A`

Field 0, the command identifier.

`Variant`

Field 1, the variant to be applied to the KEK prior to encrypting the Working Key. This field contains a 1 or 2 byte decimal value which can be in the range of 0 to 31. If the Working Key is being exported to an Atalla Network Security Processor that uses variants, specify the appropriate variant in this field. For example, if the Working Key is a KPE, the variant would be 1. If the Working Keys is being exported to a node that does not use the Atalla variant key management method, set this field to 0 (zero).

Header,$E_{MFK.E}$(KEK),MAC

Field 2, the Key Exchange Key (KEK) encrypted under the MFK. This key is used to encrypt the Working Key. This field contains a 74 byte value, or a volatile table location.

The length of the KEK must be equal-to or greater-than the length of the Working Key being exported. For example, if the Working Key is a 2key-3DES key, this field must contain either a 2key-3DES or 3key-3DES key, or a reference to a volatile table location that contains a 2key-3DES or 3key-3DES key.

The following headers are supported: 1KDEE000 and 1KDEN000. If option E2 is enabled, these additional headers are supported: 1KDNE000 and 1KDNN000. If option E3 is enabled, this additional header is supported: 1ADEE000. When both options E2 and E3 are enabled, these additional headers are supported: 1KDNN000, 1KDNE000, 1ADEE000, 1ADNN000and 1ADNE000.

`Header,`$E_{MFK.E}$`(WK),MAC`

Field 3, the Working Key encrypted under the MFK. The Working Key can have any valid header, except that header byte 4 cannot contain “N”. See Working Key Headers for a partial list of supported headers. This field contains a 74 byte value, or a volatile table location.

Table 3-18 Command 1A: Export a Working Key to non-AKB Format

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	1A
1	Variant	1, 2	0 - 31
2	Header,$E_{MFK.E}$(KEK),MAC*	74	printable ASCII
3	Header,$E_{MFK.E}$(WK),MAC*	74	printable ASCII

* Can be a volatile table location.

Responding Parameters

`2A`

Field 0, the response identifier.

$E_{\text{Key Exchange Key.V}}$`(Working Key)`

Field 1, the Working Key encrypted under the variant of the KEK specified in field one of the command. This field contains a 16, 32, or 48 byte hexadecimal value.

`Working Key Check Digits`

Field 2, check digits; that is, the first six digits that result from encrypting zeros using the Working Key. This field contains a six byte hexadecimal value.

Table 3-19 Response 2A: Export a Working Key to non-AKB Format

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	2A
1	$E_{KEK.V}$(Working Key)	16, 32, 48	0 - 9, A - F
2	Working Key Check Digits	6	0 - 9, A - F

Usage Notes

- This command is used for distributing a Working Key to a non-Atalla Network Security Processor or an Atalla Network Security Processor that does not support the Atalla Key Block.
- Generate the Working Key cryptograms.
- Option E1 must specify the key byte usage for the key type to be exported.
- The key to be exported must have an AKB header whose first 6 bytes match one of the allowed AKB headers. Use command <9A#HEADERS#> to obtain a list of allowed AKB headers.
- Use this command to export HMAC-SHA1 or HMAC-SHA256 keys that contain 176-bits or less (those keys that can be stored in a 74 character AKB). HMAC keys that contain more than 176-bits (those keys that are stored in either a 90 or 106 character AKB) cannot be exported with this command. The leftmost 4 hexadecimal characters of the decrypted key field contain a two-byte hexadecimal key length value, which specifies the key length in bytes. For example, a 128-bit key will have a length value of 0x0010. The HMAC key is appended to this length value, and if necessary, random padding characters are appended to the HMAC key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate a PIN Encryption Key for distributing to a non-Atalla node

- Variant: 0
- Clear-text Key Exchange Key (KEK): 0123 4567 89AB CDEF
  The Key Exchange Key in AKB format:
  1KDEE000,00B12295FA3537675054124D8A2FEC708CDCAF6A67104CA8,8A139C212
  92FCB8D
- Clear-text PIN Encryption Key (KPE): 0123 4567 89AB CDEF
  The PIN Encryption Key (KPE) in AKB format:
  1PUNE000,27F0026930C71646D9F4D01EFF1231C7282455FD98438661,F607ECF664
  04617D

The command looks like this:

```
<1A#0#1KDEE000,00B12295FA3537675054124D8A2FEC708CDCAF6A67104CA8,8A1
39C21292FCB8D#1PUNE000,27F0026930C71646D9F4D01EFF1231C7282455FD9843
8661,F607ECF66404617D#>
```

The Network Security Processor returns the following response:

```
<2A#56CC09E7CFDC4CEF#D5D44F#>
```

Generate New Initial Key for PIN Pad Using VISA DUKPT (Command 1E)

Command 1E re-initializes PIN pads that perform VISA Derived Unique Key Per Transaction (DUKPT) key management.

note This command supports an obsolete/proprietary technique for key loading and is not compatible with any current POS terminals. It is not required or recommended in any DUKPT environment.

This command, by default, will generate a 1key-3DES (single-length) session key. Use option A2 to control the length of the generated session key.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<1E#Header,EMFK.E(Derivation Key),MAC#
Current Key Serial Number#New Initial Key Serial Number#
[Header,EMFK.E(New Derivation Key),MAC#][Session Key Length#]>
```

Response

```
<2E#ECurrent Key(New Initial PIN Encryption Key)#
Check Value#>[CRLF]
```

Calling Parameters

`1E`

Field 0, the command identifier.

`Header,EMFK.E(Derivation Key),MAC`

Field 1, the Derivation Key encrypted under the MFK. This key should be a 2key-3DES (double-length) key. It can be a 1key-3DES (single-length) key only if option A2 is set to “S” and option 6C is enabled. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1dDNE000 and 1dDNN000.

`Current Key Serial Number`

Field 2, this value is used with the Derivation Key specified in Field 1 to derive the current PIN pad key. This field contains a 10 to 20 byte hexadecimal value.

`New Key Serial Number`

Field 3, the new key serial number for the PIN pad, left-padded with Fs. If a new Derivation Key is defined in Field 4, this field generates the new initial key serial number, otherwise, the Derivation Key in Field 1 is used. This field contains a 16 byte hexadecimal value.

[Header,$E_{MFK.E}$(New Derivation Key),MAC#]

Field 4, the new Derivation Key encrypted under the MFK. This field is required only if option A2 is set to “B”, for all other cases this field is optional. If it exists, this field contains a 74 byte value, or a volatile table location. This key should be a 2key-3DES (double-length) key. It can be a 1key-3DES (single-length) key only if option A2 is set to “S” and option 6C is enabled. This field is required, but can be empty, if option A2 is set to “B”. The following headers are supported: 1dDNE000 and 1dDNN000.

`[Session Key Length#]`

Field 5, this field is required only if option A2 is set to “B”, for all other cases this field is optional. If it exists, it should contain “S” if a 1key-3DES (single-length) session key is to be generated, or “D” if a 2key-3DES (double-length) session key is to be generated. If option A2 is set to “D”, this field cannot contain the value “S”, and if option A2 is set to “S”, this field cannot contain the value “D”.

Table 3-20 Command 1E: Generate New Initial Key for PIN Pad Using VISA DUKPT

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	1E
1	Header,$E_{MFK.E}$(Derivation Key),MAC*	74	printable ASCII
2	Current Key Serial Number	10 - 20	0 - 9, A- F
3	New Key Serial Number	16	0 - 9, A - F
4**	Header,$E_{MFK.E}$(New Derivation Key),MAC*	74	printable ASCII
5**	[Session Key Length#]	0, 1	S or D

* Can be a volatile table location.
** Optional field; used only when a new Derivation Key is needed.

Responding Parameters

`2E`

Field 0, the response identifier.

$E_{Current\ KSN}$(New Initial Key)

Field 1, the new initial key encrypted under the current PIN pad key. This field contains a 16 byte hexadecimal value.

`Check Value`

Field 2, the new initial key's check value. The length of this field depends on the length of the session key. It will contain 8 hexadecimal digits if the session key is a 1key-3DES (single-length) key. It will contain 16 hexadecimal digits if the session key is a 2key-3DES (double-length) key.

Table 3-21 Response 2E: Generate New Initial Key for PIN Pad Using VISA DUKPT

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	2E
1	$E_{\text{Current KSN}}$(New Initial Key)*	16	0 - 9, A - F
2	Check Value	8, 16	0 - 9, A - F

*E refers to special encryption defined by VISA.

Usage Notes

- This command is used to load a new initial key serial number and a new initial key into a PIN pad without taking the PIN pad out of service.
- Generate the Base Derivation Key and set option A2 appropriately.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Generate a new initial key using a 1key-3DES (single-length) session key

- Option A2 is set to "S" and option 6C is enabled.
- Clear-text Base Derivation Key: 1334 1334 1334 1334
  The Base Derivation Key in AKB format:
  1dDNE000,449DD7F6420AD2AD710102B444C180F37D0E78F5EFC186E7,1817536799D56B5E
- The current Key Serial Number: FFFF 9876 5432 10E0 0001
- The new Initial Key Serial Number: 0123 4567 89

The command looks like this:

```
<1E#1dDNE000,449DD7F6420AD2AD710102B444C180F37D0E78F5EFC186E7,18175
36799D56B5E#9876543210E00001#FFFFFF0123456789#>
```

The Network Security Processor returns the following response:

```
<2E#F90FB12DC2CD138D#1567922B#>
```

This example shows the syntax when the option A2 is set to “B” or “S”, and there is no new Derivation Key.

```
<1E#1dDNE000,449DD7F6420AD2AD710102B444C180F37D0E78F5EFC186E7,18175
36799D56B5E#9876543210E00001#FFFFFF0123456789##S#>
```

The Network Security Processor returns the following response:

```
<2E#F90FB12DC2CD138D#1567922B#>
```

This example shows the syntax when the option A2 is set to “B” or “S”, and a new Derivation Key is included in field 4. The clear text value of the new Derivation Key is 0123456789ABCDEF.

```
<1E#1dDNE000,449DD7F6420AD2AD710102B444C180F37D0E78F5EFC186E7,18175
36799D56B5E#9876543210E00001#FFFFFF0123456789#1dDNE000,791AC3DAFF7D
8502D6AE86F1F594C9AF93E5CCE8C2452D26,103BE672BF1CA47F#S#>
```

The Network Security Processor returns the following response:

```
<2E#1C96A87EDC8672CF#3AE4C948#>
```

Generate Check Digits (Command 7E)

This command generates check digits which are used to confirm that two parties hold the same key value. Each party calculates the check digits from the key using the same algorithm, and then compares results. This command is enabled in the Network Security Processor's default security policy.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

In version 1.30 and above, option 4F controls methods I and R.

Command

```
<7E#Method#Reserved#Header,EMFK.E(WK),MAC#>
```

Response

```
<8E#Check Digit Method#Generated Check Digits#>[CRLF]
```

Calling Parameters

`7E`

Field 0, the command identifier.

`Check Digit Method`

Field 1, the check digit method. This field contains 1 byte. The possible values are:

Method	Description	Value
F	E_{key}(0123456789ABCDEF)	leftmost 4
I*	E_{key}(key)	rightmost 4
R*	(E_{KEY}(KEY)) XOR KEY	rightmost 4
S	E_{key}(0000000000000000)	leftmost 4
V	E_{key}(0000000000000000)	leftmost 6

* Method R, which only supports1key-3DES (single-length) keys, is allowed only when option 4F is enabled. When option 4F is enabled, method I is not allowed.

If method I is selected and the Working Key is either a 2key- or 3key-3DES key, the rightmost key value will be encrypted under the key.

`Reserved`

Field 2, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

`Header,E`$_{MFK.E}$`(WK),MAC`

Field 3, the Working Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location.

Table 3-22 Command 7E: Generate Check Digits

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	2	7E
1	Check Digit method	1	F, I, R, S, V
2	Reserved	0, 1, 2	0 - 31
3	Header,E$_{MFK.E}$(WK),MAC*	74	printable ASCII

* Can be a volatile table location.

Responding Parameters

`8E`

Field 0, the response identifier.

`Check Digit Method`

Field 1, the check digit method supplied in field 1 of the command.

`Generated Check Digits`

Field 2, the check digits. The first four hexadecimal digits that result from encrypting zeros using the Working Key. If option 88 is enabled, this field will contain six check digits. The Visa method always returns six check digits regardless of the status of option 88.

Table 3-23 Response 8E: Generate Check Digits

Field	Contents	Length (bytes)	Legal Characters
0	Response	2	8E
1	Check Digit method	1	F, I, R, S, V
2	Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- Do not use this command to obtain check digits of HMAC-SHA1 or HMAC-SHA256 keys.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text PIN Encryption Key: 0000 0000 5555 6666
  The PIN Verification Key in AKB format:
  1PUNE000,D3266EC69C61820019F4A9640A8F603DA14F78E154C7522D,55720A06F8964B8F

Check digit method: F

The key encrypts 0123456789ABCDEFF. The leftmost 4 digits of the result are the check digits.

The command looks like this:

```
<7E#F##1PUNE000,D3266EC69C61820019F4A9640A8F603DA14F78E154C7522D,55
720A06F8964B8F#>
```

The Network Security Processor returns the following response:

```
<8E#F#E1E3#>
```

Check digit method: IBM method (I)

The key encrypts itself. The rightmost 4 digits of the result are the check digits.

The command looks like this:

```
<7E#I##1PUNE000,D3266EC69C61820019F4A9640A8F603DA14F78E154C7522D,55
720A06F8964B8F#>
```

The Network Security Processor returns the following response:

```
<8E#I#46A5#>
```

Check digit method: R

The key encrypts itself, this cryptogram is exclusive or’d with the key. The rightmost 4 digits of the result are the check digits.

The command looks like this:

```
<7E#R##1PUNE000,D3266EC69C61820019F4A9640A8F603DA14F78E154C7522D,55
720A06F8964B8F#>
```

The Network Security Processor returns the following response:

```
<8E#R#20C3#>
```

Check digit method: Standard Atalla method (S)

The key encrypts zeros. The leftmost 4 digits of the result are the check digits.

The command looks like this:

```
<7E#S##1PUNE000,D3266EC69C61820019F4A9640A8F603DA14F78E154C7522D,55
720A06F8964B8F#>
```

The Network Security Processor returns the following response:

```
<8E#S#3BAF#>
```

Check digit method: VISA method (V)

The key encrypts zeros. The leftmost 6 digits of the result are the check digits.

The command looks like this:

```
<7E#V##1PUNE000,D3266EC69C61820019F4A9640A8F603DA14F78E154C7522D,55
720A06F8964B8F#>
```

The Network Security Processor returns the following response:

```
<8E#V#3BAFC4#>
```

Translate Working Key from Current MFK to Pending MFK (Command 9E)

Command 9E translates a Working Key from encryption under the current MFK to encryption under the pending MFK. This command is enabled in the Network Security Processor's default security policy.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

Command

```
<9E#Reserved#Header,EMFK.E(WK),MAC#>
```

Response

```
<AE#Header,EPMFK.E(WK),MAC#WK Check Digits#>[CRLF]
```

Calling Parameters

`9E`

Field 0, the command identifier.

`Reserved`

Field 1, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

Header, $E_{MFK.E}$(WK),MAC

Field 2, the Working Key encrypted under the MFK. The Working Key can have any valid header, see Working Key Headers for a partial list of supported headers. This field contains a 74 byte value, or a volatile table location.

In version 1.15 and above, when header byte 2, Algorithm, contains the letter "J", this field also supports 58 and 90 byte values. In addition, there is no restriction enforced on the size of the pending MFK.

In version 1.40 and above, when header byte 2, Algorithm, contains the letter "H" or the number 2, this field also supports 90 and 106 byte values.

Table 3-24 Command 9E: Translate Working Key from Current MFK to Pending MFK

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	9E
1	Reserved	0, 1, 2	0 - 31
2	Header,$E_{MFK.E}$(WK),MAC*	58, 74, 90, 106	printable ASCII

* Can be a volatile table location.

Responding Parameters

`AE`

Field 0, the response identifier.

`Header,E`$_{PMFK.E}$`(WK),MAC`

Field 1, the Working Key decrypted under the MFK specified in field one of the command and re-encrypted using the pending MFK. The host application stores this key block on its local database for subsequent use. This field contains a 74 byte value.

`Working Key Check Digits`

Field 2, check digits, the first four hexadecimal digits that result from encrypting zeros using the Working Key. If option 88 is enabled, this field will contain six check digits.

In version 1.15 and above, this field contains 10 hexadecimal digits when the AKB supplied in field 2 of the command has the letter "J" in header byte 2, Algorithm. The CMAC check digit method, used to produce these check digits, generates a MAC of an all zero data block using the key as the CMAC key. These check digits are the leftmost 10 hexadecimal digits of the MAC result.

In version 1.40 and above, this field contains the leftmost 10 digits that result from hashing the key using the SHA-256 algorithm when the AKB supplied in field 2 of the command has the letter "H" or the number 2 in header byte 2, Algorithm.

Table 3-25 Response AE: Translate Working Key from Current MFK to Pending MFK

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	AE
1	Header,E$_{PMFK.E}$(WK),MAC	74	printable ASCII
2	Working Key Check Digits	4, 6, or 10	0 - 9, A - F

Usage Notes

- Confirm that a pending MFK has been loaded into the NSP before using this command.
- If the length of the pending MFK is not equal or greater than the length of the Working Key, the NSP will return an error response that starts with <00#0800..#>.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

The 3key-3DES Pending Master File Key is:
0123 45678 9ABC DEF 1234 1234 5678 5678 AAAA BBBB CCCC DDDD,
check digits = 1F17.

Translate a PIN Encryption Key

- Clear-text PIN Encryption Key is: 0123 4567 89AB CDEF
  The PIN Encryption Key in AKB format:
  1PUNE000,27F0026930C71646D9F4D01EFF1231C7282455FD98438661,F607ECF664 04617D

The command looks like this:

```
<9E##1PUNE000,27F0026930C71646D9F4D01EFF1231C7282455FD98438661,F607
ECF66404617D#>
```

The Network Security Processor returns the following response:

```
<AE#1PUNE000,EE99DEB52C7FB6D232C7B2CFC29998B787EB60A258E44E72,A7934
83C1972BF3C#D5D4#>
```

Translate a key with the letter J in header byte 2

The 2key-3DES Pending Master File Key is:
9810 7654 FED3 CBA2 2ABC 3DEF 4567 0189, check digits = 8792.

The command looks like this:

```
<9E##1DJNE000,DDD5F13F46787B83565536929165DA85,216558F21465C325#>
```

The Network Security Processor returns the following response:

```
<AE#1DJNE000,9A06EEF9E0DFD23A80F1902AC73568C2,55E9561B17351396#75C6
F11844#>
```

Replace the Current MFK with the Pending MFK (Command 9F)

Command 9F replaces the current MFK with the pending MFK. When the pending MFK is promoted to the MFK, the name of the new MFK is incremented. This command is enabled in the Network Security Processor's default security policy.

note Upon successful execution of this command, all keys in the volatile table are erased.

Command

```
<9F#MFK Name#MFK Check Digits#Pending MFK Name#
Pending MFK Check Digits#>
```

Response

```
<AF#OK#>[CRLF]
```

Calling Parameters

`9F`

Field 0, the command identifier.

`MFK Name`

Field 1, the current MFK's name.

`MFK Check Digits`

Field 2, the current MFK's check digits; that is the result of encrypting zeros using the MFK. This field contains a four byte hexadecimal number.

`Pending MFK Name`

Field 3, the pending MFK's name, PMFK1.

`Pending MFK Check Digits`

Field 4, the pending MFK's check digits; that is the result of encrypting zeros using the pending MFK. This field contains a four byte hexadecimal value.

Table 3-26 Command 9F: Replace Current MFK with Pending MFK

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	9F
1	MFK name	0, 4	0 -9, A- Z

Table 3-26 Command 9F: Replace Current MFK with Pending MFK (continued)

Field	Contents	Length (bytes)	Legal Characters
2	MFK Check Digits	4	0 - 9, A - F
3	Pending MFK name	5	PMFK1
4	Pending MFK Check Digits	4	0 - 9, A - F

Responding Parameters

`AF`

Field zero, the response identifier.

`OK`

Field one, an indicator that the current MFK has been replaced and the volatile table has been erased.

Table 3-27 Response AF: Replace Current MFK with Pending MFK

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	AF
1	Verification indicator	2	OK

Usage Notes

- Load the pending MFK and translate all Working Keys using the pending MFK, see Translate Working Key from Current MFK to Pending MFK (Command 9E).
- Command 9F increments the MFK name to the next value in this list: “MFK2”, “MKF3”, “MFK4”, “MFK5”, “MFK6”, “MFK7”, “MFK8”, “MFK9”, “MFKA”, “MFKB”, “MFKC”.... “MFKZ”, “MFK2”, “MFK3” ...
  For Example:
  If the current MFK name is “MFK1”, after command 9F it will be “MFK2”.
  If the current MFK name is “MFK2”, after command 9F it will be “MFK3”.
  If the current MFK name is “MFKZ”, after command 9F it will be “MFK2”.

note The MFK name will not be incremented to “MFK1”. This name is reserved for use with the SCA.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

The 3key-3DES Pending Master File Key is:
0123 45678 9ABC DEF 1234 1234 5678 5678 AAAA BBBB CCCC DDDD, check digits = 1F17.

Replace the current MFK with a pending MFK

- Current MFK’s name: MFK1
- Current MFK’s check digits: B196
- Pending MFK’s name: PMFK1
- Pending MFK’s check digits: 1F17

The command looks like this:

```
<9F#MFK1#B196#PMFK1#1F17#>
```

The Network Security Processor returns the following response:

```
<AF#OK#>
```

Translate an AKB to 3DES-ECB or 3DES-CBC (Command 113)

Command 113 decrypts a Working Key in AKB format that was encrypted under a KEK, and then encrypts it using the same KEK using either the Electronic Code Book (ECB) or the Cipher Block Chaining (CBC) modes of DES. If required, a variant, to be applied to the KEK prior to encrypting the Working Key, can be specified in field 1 of the command. The initialization vector, used in CBC mode, is binary zeros, it is not supplied in the command.

Option E1 must be enabled with the key type of the Working Key to be translated. For example, if a PIN Encryption Key (header 1PUNE000) is to be translated, option E1 must contain the letter “P”.

The KEK and Working Key can be a 1key-3DES (single-length) key if option 6C is enabled in the Network Security Processor’s security policy.

This command is enabled in the Network Security Processor’s default security policy.

Command

<113#Variant V#Header,$E_{MFK.E}$(KEK),MAC#Header,$E_{KEK.E}$(Working Key),MAC#[Mode#]>

Response

<213#$E_{KEK.V}$(Working Key)#Working Key Check Digits#>[CRLF]

Calling Parameters

`113`

Field 0, the command identifier.

`Variant V`

Field 1, the variant applied to the KEK prior to encrypting the Working Key. This field contains a 1 or 2 digit value in the range of 0 through 31. When sending the Working Key to a node that supports Atalla variants, specify the variant to be applied to the KEK, otherwise set the value of this field to zero. For example, if a PIN Encryption Key is being translated and sent to a node that supports Atalla variants, set the value of this field to 1.

Header,$E_{MFK.E}$(KEK),MAC

Field 2, the Key Exchange Key in AKB format encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The length of the KEK must be equal-to or greater-than the length of the Working Key being translated. For example, if the Working Key is a 2key-3DES key, this field must contain either a 2key-3DES or 3key-3DES key, or a reference to a volatile table location that contains a 2key-3DES or 3key-3DES key. The following headers are supported: 1KDEE000. If option E2 is enabled, this additional header is supported: 1KDNE000. If option E3 is enabled, this additional header is supported: 1ADEE000. If both options E2 and E3 are enabled, this additional header is supported: 1ADNE000.

`Header,E`$_{KEK.E}$`(Working Key),MAC`

Field 3, the Working Key in AKB format encrypted under the KEK. The Working Key can have any valid header, see Working Key Headers for a partial list of supported headers. This field contains a 74 byte value, or a volatile table location.

`[Mode#]`

Field 4, the mode of DES used in the translation of the Working Key. This field is optional. If not present, the Working Key will be translated using the CBC mode of DES. If present, this field consists of two characters:

1# - indicates translate the working using the CBC mode of DES.

2# - indicates translate the Working Key using the ECB mode of DES.

Table 3-28 Translate an AKB to 3DES-ECB or 3DES-CBC

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	113
1	Variant V	1, 2	0 - 31
2	Header,E$_{MFK.E}$(KEK), MAC*	74	printable ASCII
3	Header,E$_{KEK.E}$(Working Key),MAC*	74	printable ASCII
4	[Mode#]	empty, 2	1# or 2#

* Can be a volatile table location

Responding Parameters

`213`

Field zero, the response identifier.

`E`$_{KEK.V}$`(Working Key)`

Field 1, the Working Key encrypted, using the mode of DES specified in command field 4, under the variant specified in command field 1, of the Key Exchange Key.

`Working Key Check Digits`

Field 2, the first four digits that result from encrypting zeros using the Working Key. This field contains a four byte hexadecimal value. If Option 88 is enabled, six check digits will be returned.

Table 3-29 Response 213: Translate an AKB to 3DES-ECB or -3DES-CBC

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	213
1	$E_{KEK.V}$(Working Key)	16, 32, 48	0 - 9, A - F
2	Working Key Check Digits	4,6	0 - 9, A - F

Usage Notes

- Generate the KEK in AKB format.
- Obtain the Working Key in AKB format encrypted under the KEK.
- Do not use this command to translate HMAC-SHA1 or HMAC-SHA256 keys.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate a 2key-3DES PIN Encryption Key from AKB to CBC mode of DES

- Clear-text Key Encryption Key (KEK): 0123 4567 89AB CDEF FEDC BA98 7654 3210, check digits = 08D7
  The KEK in AKB format:
  1KDEE000,00B12295FA353767B3E2F92C51F8C4799EE072B479C8C889,815074BF95E27120
- Clear-text PIN Encryption Key (KPE): FEDC BA98 7654 3210 0123 4567 89AB CDEF, check digits = 7B83
  The KPE in AKB format encrypted under the KEK:
  1PUNE000,F19AB270BD66314BC04A30A9C1C81AB0787854EB0A044DEA,5801B818688303F2
- No variant is to be applied to the KEK when encrypting the Working Key for output.

The command looks like this:

```
<113#0#1KDEE000,00B12295FA353767B3E2F92C51F8C4799EE072B479C8C889,81
5074BF95E27120#1PUNE000,F19AB270BD66314BC04A30A9C1C81AB0787854EB0A0
44DEA,5801B818688303F2#>
```

The Network Security Processor returns the following response:

```
<213#1FD1B02B237AF9AE3702240E7080EA4C#7B83#>
```

Translate a 2key-3DES PIN Encryption Key from AKB to EBC mode of DES

- Clear-text Key Encryption Key (KEK): 0123 4567 89AB CDEF FEDC BA98 7654 3210, check digits = 08D7
  The KEK in AKB format:
  1KDEE000,00B12295FA353767B3E2F92C51F8C4799EE072B479C8C889,815074BF95E27120
- Clear-text PIN Encryption Key (KPE): FEDC BA98 7654 3210 0123 4567 89AB CDEF, check digits = 7B83
  The KPE in AKB format encrypted under the KEK:
  1PUNE000,F19AB270BD66314BC04A30A9C1C81AB0787854EB0A044DEA,5801B818688303F2
- Variant one is to be applied to the KEK when encrypting the Working Key for output.

The command looks like this:

```
<113#1#1KDEE000,00B12295FA353767B3E2F92C51F8C4799EE072B479C8C889,81
5074BF95E27120#1PUNE000,F19AB270BD66314BC04A30A9C1C81AB0787854EB0A0
44DEA,5801B818688303F2#2#>
```

The Network Security Processor returns the following response:

```
<213#BA5BC3B6A3EBF89A46919EAFC518618D#7B83#>
```

Import TR-31 Formatted Working Key (Command 117)

Command 117 translates a TR-31 formatted Working Key that is encrypted under a Key Exchange Key to encryption under the Master File Key. The imported TR-31 formatted Working Key is returned in the Atalla Key Block format. The header for the TR-31 key block defines the header to be assigned to the AKB of the Atalla Working Key. See Table 3-33 on page 3-57, Table 3-34 on page 3-58, and Table 3-35 on page 3-58.

Option E0 must be enabled with the Byte 1, Key Usage value of the Working Key to be imported. For example, if a PIN Encryption Key (TR-31 key usage value = P0) is to be imported, option E0 must contain the letter “P”. The AKB key usage value for Base Derivation and EMV Master key types do not match the TR-31 key usage values. When importing these type of keys, option E0 must contain the AKB key usage byte value, not the TR-31 key usage value. See Table 3-32 on page 3-56.

The KEK and Working Key can be a 1key-3DES (single-length) key if option 6C is enabled in the Network Security Processor’s security policy.

This command is not enabled in the Network Security Processor’s default security policy. To use this command, you must add it to the Network Security Processor’s security policy.

Command

```
<117#Reserved#Header,EMFK.E(KEK),MAC#TR-31 Key Block#>
```

Response

```
<217#Status#Header,EMFK.E(Working Key),MAC#
Working Key Check Digits#>[CRLF]
```

Calling Parameters

`117`

Field 0, the command identifier.

`Reserved`

Field 1, this field is reserved for future use, any value in this field will be ignored. This field can be 0, 1, or 2 bytes long.

Header,$E_{MFK.E}$(KEK),MAC

Field 2, the Key Exchange Key in AKB format encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The length of the KEK must be equal-to or greater-than the length of the Working Key being translated. For example, if the Working Key is a 2key-3DES key, this field must contain either a 2key-3DES or 3key-3DES key, or a reference to a volatile tile location that contains a 2key-3DES or 3key-3DES key. The following headers are supported: 1KDDE000 and 1KDDN000. If option E2 is enabled, these additional headers are supported: 1KDNE000 and 1KDNN000.

`TR-31 Key Block`

Field 3, the Working Key in TR-31 format encrypted under the KEK. The minimum length of this field is 56 bytes, the maximum size of this field is 4000. Any data present in a TR-31 Optional Block will not be converted to AKB format. The TR-31 header is used to derive the header of the AKB.

Table 3-30 Command 117: Import TR-31 Formatted Working Key

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	117
1	Reserved	0, 1, 2	0 - 31
2	Header,$E_{MFK.E}$(KEK), MAC*	74	printable ASCII
3	TR-31 Key Block	56 - 4000	printable ASCII

* Can be a volatile table location

Responding Parameters

`217`

Field zero, the response identifier.

`Status`

Field 1, status of the MAC verification on the TR-31 key block. If the MAC verifies, this field will contain the letter “Y”. If the MAC does not verify, this field will contain the letter “N”, and fields 2 and 3 will be empty.

Header,$E_{MFK.E}$(Working Key),MAC

Field 2, the Working Key in AKB format encrypted under the MFK.

`Working Key Check Digits`

Field 3, the first four digits that result from encrypting zeros using the Working Key. This field contains a four byte hexadecimal value. If Option 88 is enabled, six check digits will be returned.

Table 3-31 Response 217: Import TR-31 Formatted Working Key

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	217
1	MAC Verification Indicator	1	Y, N

Table 3-31 Response 217: Import TR-31 Formatted Working Key (continued)

Field	Contents	Length (bytes)	Legal Characters
2	Header,$E_{MFK.E}$(Working Key),MAC	74	printable ASCII
3	Working Key Check Digits	4,6	0 - 9, A - F

Usage Notes

- Any data present in a TR-31 Optional Block will not be converted to AKB format.
- Do not use this command to import HMAC-SHA1 or HMAC-SHA256 keys.

The following table defines the AKB header byte value(s) assigned to each TR-31 key usage value.

Table 3-32 AKB Key Usage value derived from TR-31 Key Usage value

TR-31 Key Usage Value (bytes 5 - 6)	TR-31 Definition	AKB Key Usage Value (byte 1)	Additional AKB header byte affected
B0	BDK Base Derivation Key	d	
C0	CVK Card Verification Key	C	
D0	Data Encryption/Decryption	D	
E0	EMV chip card Master Key: Application Cryptogram	m	AKB Header byte 7 = 'a'
E1	EMV chip card Master Key: Confidentiality	m	AKB Header byte 7 = 'E'
E2	EMV chip card Master Key: Integrity	m	AKB Header byte 7 = 'M'
E3	EMV chip card Master Key: Data Authentication Code	m	AKB Header byte 7 = 'd'
E4	EMV chip card Master Key: Dynamic Numbers	m	AKB Header byte 7 = 'e'
E5	EMV chip card Master Key: Card Personalization	m	AKB Header byte 7 = 'f'
E6	EMV chip card Master Key: Other	m	AKB Header byte 7 = 'g'
I0	Initialization Vector	I	
K0	Key Encryption or Wrapping	K	
M0	ISO 16609 MAC algorithm 1 (using TDEA)	M	AKB Header byte 7 = '1'
M1	ISO 9797-1 MAC algorithm 1	M	AKB Header byte 7 = '1'

Table 3-32 AKB Key Usage value derived from TR-31 Key Usage value (continued)

TR-31 Key Usage Value (bytes 5 - 6)	TR-31 Definition	AKB Key Usage Value (byte 1)	Additional AKB header byte affected
M2*	ISO 9797-1 MAC algorithm 2	M	AKB Header byte 7 = '2'
M3	ISO 9797-1 MAC algorithm 3	M	AKB Header byte 7 = '3'
M4*	ISO 9797-1 MAC algorithm 4	M	AKB Header byte 7 = '4'
M5*	ISO 9797-1 MAC algorithm 5	M	AKB Header byte 7 = '5'
P0	PIN Encryption	P	
V0	PIN Verification, other algorithm	V	AKB Header byte 2 = 'U'
V1	PIN Verification, IBM 3624	V	AKB Header byte 2 = '3'
V2	PIN Verification, Visa	V	AKB Header byte 2 = 'V'

* Atalla Network Security Processor MAC commands do not support this MAC algorithm.

The following table defines the AKB Mode of Use value (byte 3) assigned to each TR-31 Mode of Use value (byte 8).

Table 3-33 AKB Mode of Use value derived from TR-31 Mode of Use value

TR-31 Mode of Use (byte 8)	TR-31 Definition	AKB Mode of Use (byte 3)	Relevant Key Types
B	Both Encrypt and Decrypt	N	KD, KEK, KPE, and EMV Master Key
C	Calculate (Generate or Verify)	N	KPV, KCVV, and KMAC
D	Decrypt Only	D	KD, KEK, KPE, and EMV Master Key
E	Encrypt Only	E	KD, KEK, KPE, and EMV Master Key
G	Generate Only	G	KPV, KCVV, and KMAC
N	No restriction, not applicable	N	all
S	Signature Only	not supported	Signature Only
V	Verify Only	V	KPV, KCVV, and KMAC

This table defines the AKB Algorithm value (byte 2) assigned to each TR-31 Algorithm value (byte 7).

Table 3-34 AKB Algorithm value derived from TR-31 Algorithm value

TR-31 Algorithm (byte 7)	TR-31 Definition	AKB Algorithm (byte 2)
A	AES	not supported
D	DEA	D
E	Elliptic Curve	not supported
H	HMAC-SHA-1	not supported
R	RSA	not supported
S	DSA	not supported
T	Triple DEA (TDEA)	D

The following table defines the AKB Exportability value (byte 4) assigned to each TR-31 Algorithm value (byte 11).

Table 3-35 AKB Exportability value derived from TR-31 Algorithm value

TR-31 Algorithm (byte 11)	TR-31 Definition	AKB Algorithm (byte 4)
E	Exportable under trusted key	E
N	Not exportable	N
S	Sensitive, Exportable under untrusted key	S

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Import TR-31 key block without Optional Block

- Clear-text Key Encryption Key (KEK):
  89E88CF7931444F3 34BD7547FC3F380C (check digits = D1D8)
  The KEK in AKB format:
  1KDDE000,B3800629FFEB4AE8359EAF3D9FB9129A00BBBF8BD9B5A0CE,B0F166D037EEA2DB

- TR-31 Key Block:
  A0072P0TE00E0000F5161ED902807AF26F1D62263644BD24192FDB3193C730301CEE8701

The command looks like this:

```
<117##1KDDE000,B3800629FFEB4AE8359EAF3D9FB9129A00BBBF8BD9B5A0CE,B0F
```

```
166D037EEA2DB#A0072P0TE00E0000F5161ED902807AF26F1D62263644BD24192FD
B3193C730301CEE8701#>
```

The Network Security Processor returns the following response:

```
<217#Y#1PDEE000,C7F6A1BC2FF07A28251946AEEBEB86CEE370157B3B9E5122,F9
8C7A54EBE6C950#CB9D#>
```

Import TR-31 key block with Optional Block

- Clear-text Key Encryption Key (KEK):
  B8ED59E0A279A295 E9F5ED7944FD06B9 (check digits = 2F0E)
  The KEK in AKB format:
  1KDDE000,569AD7D68325400D06CC40B29FB733C45004F248735E1BC9,C28DB8C
  A1345EFA5

- TR-31 Key Block:
  A0112P0TE12E0200KS1400604B120F929280PB047A1BB737854CD7AF58A8A1E450
  6A942277EDA76EBA6BA228AF62ADDA3AD8799E8B2C8CD7

The command looks like this:

```
<117##1KDDE000,569AD7D68325400D06CC40B29FB733C45004F248735E1BC9,C28
DB8CA1345EFA5#A0112P0TE12E0200KS1400604B120F929280PB047A1BB737854CD
7AF58A8A1E4506A942277EDA76EBA6BA228AF62ADDA3AD8799E8B2C8CD7#>
```

The Network Security Processor returns the following response:

```
<217#Y#1PDEE000,C7F6A1BC2FF07A28251946AEEBEB86CEE370157B3B9E5122,F9
8C7A54EBE6C950#CB9D#>
```

Export Working Key in TR-31 Format (Command 118)

Command 118 translates a Working Key that is encrypted under the MFK to encryption under the Key Exchange Key. The exported Working Key is returned in the TR-31 format. The AKB header for the Working Key defines the header to be assigned to the TR-31 formatted Working Key. TR3-1 supports a subset of AKB headers, therefore not all AKB keys can be exported in TR-31 format. See Table 3-39 on page 3-64, Table 3-40 on page 3-65, and Table 3-41 on page 3-66.

Option E1 must be enabled with the Byte 1, Key Usage value of the Working Key to be exported. For example, if a PIN Encryption Key (header 1PDNE000) is to be exported, option E1 must contain the letter “P”. The AKB key usage value for Base Derivation and EMV Master key types do not match the TR-31 key usage values. When exporting these type of keys, option E1 must contain the AKB key usage byte value, not the TR-31 key usage value. See Table 3-38 on page 3-63.

The KEK and Working Key can be a 1key-3DES (single-length) key if option 6C is enabled in the Network Security Processor’s security policy.

This command is not enabled in the Network Security Processor’s default security policy. To use this command, you must add it to the Network Security Processor’s security policy.

Command

```
<118#Reserved#Header,EMFK.E(KEK),MAC#Header,EMFK.E(WK),MAC#
Key Version Number#Number of Optional blocks#
[Optional Block Data 0-99#]>
```

Response

```
<218#TR-31 Key Block#Working Key Check Digits#>[CRLF]
```

Calling Parameters

`118`

Field 0, the command identifier.

`Reserved`

Field 1, this field is reserved for future use, any value in this field will be ignored. This field can be 0, 1, or 2 bytes long.

`Header,EMFK.E(KEK),MAC`

Field 2, the Key Exchange Key in AKB format encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The length of the KEK must be equal-to or greater-than the length of the Working Key being translated. For example, if the Working Key is a 2key-3DES key, this field must contain either a 2key-3DES or 3key-3DES key, or a reference to a volatile tile location that contains a 2key-3DES or

3key-3DES key. The following headers are supported: 1KDDE000 and 1KDDN000. If option E2 is enabled, these additional headers are supported: 1KDNE000 and 1KDNN000.

`Header,E`$_{MFK.E}$`(Working Key),MAC`

Field 3, the Working Key in AKB format encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The AKB of the Working Key to be exported must have a header that adheres to all of these requirements:

- Bytes 0 through 2 must contain any one of these values:

  "1dD", "1CD", "1DD", "1mE", "1ID", "1KD", "1MD", "1PD", "1VU", "1V3", "1VV"

- Byte 3 must contain one of these values:

  "N", "D", "E", "G", or "V"

- Byte 4 must contain the letter "E"

- Bytes 5 through 7 can be any legal value

`Key Version Number`

Field 4, two ASCII characters as defined in Table A.5.4 of the *x9 TR-31 2010 Interoperable Secure Key Exchange KeyBlock Specification for Symmetric Algorithms* document.

`Number of Optional Blocks`

Field 5, a two digit decimal value that defines the number of Optional Blocks included in the TR-31 Key Block. The minimum value is 00 and the maximum is 99, however the number of optional blocks is also limited by the maximum number of 4000 bytes in the entire TR-31 Key Block.

If the total length of all optional blocks is not a multiple of 8, the Network Security Processor will add a padding block consisting of optional ID = 'PB', 2 bytes for length, and pad characters. For example, if the length of all optional fields is 90, the Network Security Processor will add a padding block of 'PB0600' (0x504230363030). Another example, if the length of all optional fields is 94, the Network Security Processor will add a padding block of 'PB0A000000' (10 bytes of padding total, 4 for the ID and length, 6 bytes of ASCII zeros).

`[Optional Block Data 0-99#]`

Field 6, this field is present only if field 5 is not zero. Each optional block data consists of three fields: a two-byte hex-ASCII identifier that indicates the content of the optional block, a two-byte hex-ASCII length that indicates the total number of bytes in all three of the optional block's fields, and the hex-ASCII variable-length data carried in the optional block. See Table A-8, of the *x9 TR-31 2010 Interoperable Secure Key Exchange KeyBlock Specification for Symmetric Algorithms* document, for a list of valid identifiers.

Table 3-36 Command 118: Export Working Key in TR-31 Format

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	118
1	Reserved	0, 1, 2	0 - 31
2	Header,$E_{MFK.E}$(KEK), MAC*	74	printable ASCII
3	Header,$E_{MFK.E}$(Working Key),MAC*	74	printable ASCII
4	Key Version Number	2	printable ASCII
5	Number of Optional Blocks	2	00 - 99
6	[Optional Block Data 0 - 99]	4 - 3912	printable ASCII

* Can be a volatile table location

Responding Parameters

`218`

Field zero, the response identifier.

`TR-31 Key Block`

Field 1, the Working Key encrypted under the KEK and formatted in a TR-31 key block. The minimum length will be 88 bytes assuming no Optional Blocks were provided in field 6 of the command. All keys are padded as follows:

- 3key-3DES (triple-length) keys will have 6 bytes of random padding.
- 2key-3DES (double-length) keys will have 14 bytes of random padding.
- 1key-3DES (single-length) keys will have 22 bytes of random padding.

The maximum length of the TR-31 key block is 4000 bytes, assuming Optional Blocks were defined in field 6 of the command.

`Working Key Check Digits`

Field 2, the first four digits that result from encrypting zeros using the Working Key. If Option 88 is enabled, six check digits will be returned.

Table 3-37 Response 218: Export Working Key in TR-31 Format

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	218

Table 3-37 Response 218: Export Working Key in TR-31 Format (continued)

Field	Contents	Length (bytes)	Legal Characters
1	TR-31 Key Block	88 - 4000	printable ASCII
2	Working Key Check Digits	4, 6	0 - 9, A - F

Usage Notes

- Generate the KEK in AKB format.
- Obtain the Working Key in AKB format encrypted under the MFK.
- Do not use this command to export HMAC-SHA1 or HMAC-SHA256 keys.

The following table defines the AKB header byte value(s) assigned to each TR-31 key usage value.

Table 3-38 TR-31 Key Usage value derived from AKB Key Usage value

AKB Key Usage Value (byte 1)	Additional AKB header byte	TR-31 Key Usage Value (bytes 5 - 6)	TR-31 Definition
d		B0	BDK Base Derivation Key
C		C0	CVK Card Verification Key
D		D0	Data Encryption/Decryption
m	AKB Header byte 7 = ‘a’	E0	EMV chip card Master Key: Application Cryptogram
m	AKB Header byte 7 = ‘E’	E1	EMV chip card Master Key: Confidentiality
m	AKB Header byte 7 = ‘M’	E2	EMV chip card Master Key: Integrity
m	AKB Header byte 7 = ‘d’	E3	EMV chip card Master Key: Data Authentication Code
m	AKB Header byte 7 = ‘e’	E4	EMV chip card Master Key: Dynamic Numbers
m	AKB Header byte 7 = ‘f’	E5	EMV chip card Master Key: Card Personalization
m	AKB Header byte 7 = ‘g’	E6	EMV chip card Master Key: Other
I		I0	Initialization Vector
K		K0	Key Encryption or Wrapping
M	AKB Header byte 7 = ‘1’	M1	ISO 9797-1 MAC algorithm 1
M	AKB Header byte 7 = ‘2’	M2*	ISO 9797-1 MAC algorithm 2

Table 3-38 TR-31 Key Usage value derived from AKB Key Usage value (continued)

AKB Key Usage Value (byte 1)	Additional AKB header byte	TR-31 Key Usage Value (bytes 5 - 6)	TR-31 Definition
M	AKB Header byte 7 = ‘3’	M3	ISO 9797-1 MAC algorithm 3
M	AKB Header byte 7 = ‘4’	M4*	ISO 9797-1 MAC algorithm 4
M	AKB Header byte 7 = ‘5’	M5*	ISO 9797-1 MAC algorithm 5
P		P0	PIN Encryption
V	AKB Header byte 2 = ‘U’	V0	PIN Verification, other algorithm
V	AKB Header byte 2 = ‘3’	V1	PIN Verification, IBM 3624
V	AKB Header byte 2 = ‘V’	V2	PIN Verification, Visa

* Atalla Network Security Processor MAC commands do not support this MAC algorithm.

The following table defines the AKB Mode of Use value (byte 3) assigned to each TR-31 Mode of Use value (byte 8).

Table 3-39 TR-31 Mode of Use value derived from AKB Mode of Use value

AKB Mode of Use (byte 3)	TR-31 Mode of Use (byte 8)	TR-31 Definition	Relevant Key Types
N	B	Both Encrypt and Decrypt	KD, KEK, KPE, and EMV Master Key
N	C	Calculate (Generate or Verify)	KPV, KCVV, and KMAC
D	D	Decrypt Only	KD, KEK, KPE, and EMV Master Key
E	E	Encrypt Only	KD, KEK, KPE, and EMV Master Key
G	G	Generate Only	KPV, KCVV, and KMAC
not supported	S	Signature Only	Signature Only
V	V	Verify Only	KPV, KCVV, and KMAC

The following table defines the AKB Algorithm value (byte 2) assigned to each TR-31 Algorithm value (byte 7).

Table 3-40 TR-31 Algorithm value derived from AKB Algorithm value

AKB Algorithm (byte 2)	**AKB Definition**	**TR-31 Algorithm (byte 7)**
0	SDI Security Dynamic	not supported
3	IBM 3624	T
7	IBM 4731	not supported
B	Atalla Bilevel	not supported
C	Conversion Table	not supported
D	DES - AKB	T
E	EMV Key Derivation	T
F	Key Derivation	not supported
I	Identikey	not supported
N	NCR	not supported
U	unknown or unspecified	T
V	VISA	T
X	American Express	not supported
Z	Key Derivation	not supported
a	Atalla 2x2	not supported
b	Key Derivation	not supported
d	Diebold Number Table	not supported
i	ICC Key Derivation	not supported
r	Routex Derivation	not supported
u	Unisys (Burroughs)	not supported

The following table defines the AKB Exportability value (byte 4) assigned to each TR-31 Algorithm value (byte 11).

Table 3-41 AKB Exportability value derived from TR-31 Algorithm value

AKB Algorithm (byte 4)	**AKB Definition**	**TR-31 Algorithm (byte 11)**
E	Exportable under trusted key	E
N	Not exportable	N
S	Sensitive, Exportable under untrusted key	S

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

The Network Security Processor generates random hexadecimal characters to pad the key before it is encrypted therefore your test results will not match these examples.

Export a 2key-3DES PIN Encryption Key to TR-31 format with No Optional Block

- Clear-text Key Encryption Key (KEK)
  89E88CF7931444F3 34BD7547FC3F380C (check digits = D1D8)
  The KEK in AKB format:
  1KDDE000,B3800629FFEB4AE8359EAF3D9FB9129A00BBBF8BD9B5A0CE,B0F166D037EEA2DB

- Clear-text PIN Encryption Key (KPE)
  F039121BEC83D26B 169BDCD5B22AAF8F (check digits = CB9D)
  The KPE in AKB format:
  1PDEE000,C7F6A1BC2FF07A28251946AEEBEB86CEE370157B3B9E5122,F98C7A54EBE6C950

The command looks like this:

```
<118##1KDDE000,B3800629FFEB4AE8359EAF3D9FB9129A00BBBF8BD9B5A0CE,B0F
166D037EEA2DB#1PDEE000,C7F6A1BC2FF07A28251946AEEBEB86CEE370157B3B9E
5122,F98C7A54EBE6C950#00#00#>
```

The Network Security Processor returns the following response:

```
<218#A0088P0TE00E00007DD4DD9566DC0E2F956DCAC0FDE91531723FD88F18DE07
1A57189B4D3C483341ED79F4E0#CB9D#>
```

Export a 2key-3DES PIN Encryption Key to TR-31 format with Optional Block

- Clear-text Key Encryption Key (KEK)
  B8ED59E0A279A295 E9F5ED7944FD06B9 (check digits = 2F0E)
  The KEK in AKB format:
  1KDDE000,569AD7D68325400D06CC40B29FB733C45004F248735E1BC9,C28DB8C
  A1345EFA5

- Clear-text PIN Encryption Key (KPE)
  F039121BEC83D26B 169BDCD5B22AAF8F (check digits = CB9D)
  The KPE in AKB format:
  1PDEE000,C7F6A1BC2FF07A28251946AEEBEB86CEE370157B3B9E5122,F98C7A54E
  BE6C950

The command looks like this:

```
<118##1KDDE000,569AD7D68325400D06CC40B29FB733C45004F248735E1BC9,C28
DB8CA1345EFA5#1PDEE000,C7F6A1BC2FF07A28251946AEEBEB86CEE370157B3B9E
5122,F98C7A54EBE6C950#12#02#KS1400604B120F929280PB04#>
```

The Network Security Processor returns the following response:

```
<218#A0112P0TE12E0200KS1400604B120F929280PB047A1BB737854CD7AF58A8A1
E4506A942277EDA76EBA6BA228AF62ADDA3AD8799E8B2C8CD7#CB9D#>
```

Import TR-31 Formatted Working Key, versions A, B or C (Command 119)

Command 119 translates a TR-31 formatted Working Key that is encrypted under a Key Block Protection Key (KBPK) to encryption under the Master File Key. The imported TR-31 formatted Working Key is returned in the Atalla Key Block format. The header from the TR-31 key block defines the header to be assigned to the AKB of the Working Key.

This command expands the functionality of command 117, which supports TR-31 version A. TR-31 version B and C are supported, and limited support is provided for version A. It adheres to the *X9 TR-31 2010, Interoperable Secure Key Exchange Key Block Specification for Symmetric Algorithms, December 9, 2010*.

note TR-31 version C is identical to version A, except that the values of some of the key headers have been clarified.

Option E0 must be enabled with the Byte 1, Key Usage value of the Working Key to be imported. For example, if a PIN Encryption Key (TR-31 key usage value = P0) is to be imported, option E0 must contain the letter “P”.

This command will support a 1key-3DES (single-length) Working Key if option 6C is enabled in the Network Security Processor’s security policy. The Key Block Protection Key cannot be a 1key-3DES (single-length) key.

This command is not enabled in the Network Security Processor’s default security policy. To use this command, you must add it to the Network Security Processor’s security policy.

Command

```
<119#AKB Version#[KPE Algorithm]#Header,EMFK.E(KBPK),MAC#
TR-31 Key Block#>
```

Response

```
<219#Header,EMFK.E(Working Key),MAC#Working Key Check Digits#>
[CRLF]
```

Calling Parameters

`119`

Field 0, the command identifier.

`AKB Version`

Field 1, this field must contain the number 1.

`[KPE Algorithm]`

Field 2, this field must contain a value when the Working Key, provided in field 4, is a PIN Encryption Key; the TR-31 header bytes 5 and 6 contain “P0”. This field specifies what value will be used as Byte 2, Algorithm in the AKB header. You can specify either “U” or “D”. This field must be empty when the Working Key is not a PIN Encryption Key.

Header,$E_{MFK.E}$(KBPK),MAC

Field 3, the Key Block Protection Key in AKB format encrypted under the MFK. This key is used to derive the encryption and MAC keys that protect the Working Key. This field contains a 74 byte value, or a volatile table location. The length of the KBPK must be equal-to or greater-than the length of the Working Key being translated. For example, if the Working Key is a 2key-3DES key, this field must contain either a 2key-3DES or 3key-3DES key, or a reference to a volatile table location that contains a 2key-3DES or 3key-3DES key. The following headers are supported: 1kDDE000 and 1kDDN000. If option E2 is enabled, these additional headers are supported: 1kDNE000 and 1kDNN000.

`TR-31 Key Block`

Field 4, the Working Key in TR-31 format, encrypted under the KBPK. The minimum length is 56 bytes, the maximum length is 4000 bytes. Any data present in a TR-31 Optional Block will not be converted to AKB format. The TR-31 header is used to derive the header of the AKB, see AKB header value derived from TR-31 Key header.

Table 3-42 Command 119: Import TR-31 Formatted Working Key, versions A, B or C

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	119
1	AKB Version	1	1
2	[KPE Algorithm]	0, 1	U or D
3	Header,$E_{MFK.E}$(KBPK), MAC*	74	printable ASCII
4	TR-31 Key Block	56 - 4000	printable ASCII

* Can be a volatile table location

Responding Parameters

`219`

Field zero, the response identifier.

Header,$E_{MFK.E}$(Working Key),MAC

Field 1, the Working Key in AKB format encrypted under the MFK.

`Working Key Check Digits`

Field 3, the first four digits that result from encrypting zeros using the Working Key. If Option 88 is enabled, six check digits will be returned.

Table 3-43 Response 219: Import TR-31 Formatted Working Key, versions A, B or C

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	219
1	Header,$E_{MFK.E}$(Working Key),MAC	74	printable ASCII
2	Working Key Check Digits	4, 6	0 - 9, A - F

Usage Notes

- Both the sending and receiving nodes must have the same KBPK value.
- The Master File Key must be a 3key-3DES key if the KBPK is a 3key-3DES key.
- Any data present in a TR-31 Optional Block will not be converted to AKB format.
- Do not use this command to import HMAC-SHA1 or HMAC-SHA256 keys.
- Not all TR-31 headers can be converted to AKB headers. The following table defines the AKB header derived from the TR-31 header bytes 5 through 8 and 11. The TR-31 header byte 11 (Exportability) will be used as byte 4 in the AKB header, this is indicated by the "*" character in the byte 4 of the AKB header in the following table.

Table 3-44 AKB header value derived from TR-31 Key header

TR-31 Header Bytes 5 - 8	Definition	AKB Header
B0DX or B0TX	BDK Base Derivation Key	1dDN*000
C0DC or C0TC	CVK Card Verification Key	1CDN*000
C0DG or C0TG	CVK Card Verification Key - generate only	1CDG*000
C0DV or C0TV	CVK Card Verification Key - verify only	1CDV*000
D0DB or D0TB	Data Encryption using ECB, CBC, CFB, OFB, CCM, or CTR	1DDN*000
D0DE or D0TE	Data Encryption using ECB, CBC, CFB, OFB, CCM, or CTR - encrypt only	1DDE*000
D0DE or D0TE	Data Encryption using ECB, CBC, CFB, OFB, CCM, or CTR - decrypt only	1DDD*000
E0DX or E0TX	EMV chip card Master Key: Application Cryptogram	1mEN*000
E1DX or E1TX	EMV chip card Master Key: Confidentiality	1mEN*00E
E2DX or E2TX	EMV chip card Master Key: Integrity	1mEN*00M

Table 3-44 AKB header value derived from TR-31 Key header (continued)

TR-31 Header Bytes 5 - 8	Definition	AKB Header
I0DN or I0TN	Initialization Vector	1IDN*000
K0DB or K0TB	Key Encryption or Wrapping	1KDN*000
K0DD or K0TD	Key Encryption or Wrapping - decrypt, unwrap only	1KDD*000
K0DE or K0TE	Key Encryption or Wrapping - encrypt, wrap only	1KDE*000
K1DB or K1TB	TR-31 Key Block Protection Key	1kDN*000
K1DD or K1TD	TR-31 Key Block Protection Key - decrypt, unwrap only	1kDD*000
K1DE or K1TE	TR-31 Key Block Protection Key - encrypt, wrap only	1kDE*000
M0DC or M0TC	ISO 16609 MAC algorithm 1 (using TDEA). Equivalent to ISO 9797-1 MAC Algorithm 1.	1MDN*000
M0DG or M0TG	ISO 16609 MAC algorithm 1 (using TDEA) - generate only. Equivalent to ISO 9797-1 MAC Algorithm 1.	1MDG*000
M0DV or M0TV	ISO 16609 MAC algorithm 1 (using TDEA) - verify only. Equivalent to ISO 9797-1 MAC Algorithm 1.	1MDV*000
M1DC or M1TC	ISO 9797-1 MAC algorithm 1	1MDN*000
M1DG or M1TG	ISO 9797-1 MAC algorithm 1 - generate only	1MDG*000
M1DV or M1TV	ISO 9797-1 MAC algorithm 1 - verify only	1MDV*000
M3DC or M3TC	ISO 9797-1 MAC algorithm 3	1MDN*000
M3DG or M3TG	ISO 9797-1 MAC algorithm 3 - generate only	1MDG*000
M3DV or M3TV	ISO 9797-1 MAC algorithm 3 - verify only	1MDV*000
P0DB or P0TB	PIN Encryption	1PDN*000, 1PUN*000
P0DE or P0TE	PIN Encryption - encrypt only	1PDE*000, 1PUE*000
P0DD or P0TD	PIN Encryption - decrypt only	1PDD*000, 1PUD*000
V1DC or V1TC	PIN Verification, IBM 3624	1V3N*000
V1DG or V1TG	PIN Verification, IBM 3624 - generate only	1V3G*000
V1DV or V1TV	PIN Verification, IBM 3624 - verify only	1V3V*000
V2DC or V2TC	PIN Verification, Visa	1VVN*000
V2DG or V2TG	PIN Verification, Visa - generate only	1VVG*000
V2DV or V2TV	PIN Verification, Visa - verify only	1VVV*000

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Import TR-31 key block - no optional block, key variant method

- Clear-text Key Block Protection Key (KBPK):
  89E88CF7931444F3 34BD7547FC3F380C, check digits = D1D8
  The KBPK in AKB format:
  1kDDE000,BBE35384A9AC9E7BB5B33F4B16FE07B726BD45148EBE4867,C160CE8D
  D67EE964
- TR-31 Key Block:
  A0072P0TE00E0000F5161ED902807AF26F1D62263644BD24192FDB3193C730301C
  EE8701
- Clear-text PIN Encryption Key being imported:
  F039121BEC83D26B169BDCD5B22AAF8F, check digits = CB9D

The command looks like this:

```
<119#1#D#1kDDE000,BBE35384A9AC9E7BB5B33F4B16FE07B726BD45148EBE4867,
C160CE8DD67EE964#A0072P0TE00E0000F5161ED902807AF26F1D62263644BD2419
2FDB3193C730301CEE8701#>
```

The Network Security Processor returns the following response:

```
<219#1PDEE000,C7F6A1BC2FF07A28251946AEEBEB86CEE370157B3B9E5122,F98C
7A54EBE6C950#CB9D#>
```

Import TR-31 key block - no optional block, key derivation method

- Clear-text Key Block Protection Key (KBPK):
  DD7515F2BFC17F85 CE48F3CA25CB21F6, check digits = F7BA
  The KBPK in AKB format:
  1kDNE000,33A4BC4FB361DEEABF74BF61927AD9B98C1BCE9C880E94C5,836AE3A7
  8004B182
- TR-31 Key Block:
  B0080P0TE00E000094B420079CC80BA3461F86FE26EFC4A3B8E4FA4C5F5341176
  EED7B727B8A248E
- Clear-text PIN Encryption Key being imported:
  3F419E1CB7079442 AA37474C2EFBF8B8, check digits = 57C4

The command looks like this:

```
<119#1#U#1kDNE000,33A4BC4FB361DEEABF74BF61927AD9B98C1BCE9C880E94C5,
836AE3A78004B182#B0080P0TE00E000094B420079CC80BA3461F86FE26EFC4A3B8
```

```
E4FA4C5F5341176EED7B727B8A248E#>
```

The Network Security Processor returns the following response:

```
<219#1PUEE000,7229E056573E3A5521DF6BBED41A82EC3FEBABA4292C2148,1EAA
31A17A86AA11#57C4#>
```

Import TR-31 key block - with optional block, key variant method

- Clear-text Key Block Protection Key (KBPK):
  EDB380DD340BC262 0247D445F5B8D678 (check digits = F4B0)
  The KBPK in AKB format:
  1kDDE000,3A2C64D2A96ED041EB33D1D85167BE5AE059D0A650D8C6F0,93A4D34
  5D18373A5
- TR-31 Key Block:
  A0096K0TD12S0100KS1800604B120F929280000051722D135D3CE0E0
  E0E7A2BA64F93AC6A5656E5E867A7F180F9A0367
- Clear-text PIN Encryption Key being imported:
  EDB380DD340BC262 0247D445F5B8D678, check digits = F4B0

The command looks like this:

```
<119#1##1kDDE000,3A2C64D2A96ED041EB33D1D85167BE5AE059D0A650D8C6F0,9
3A4D345D18373A5#A0096K0TD12S0100KS1800604B120F929280000051722D135D3
CE0E0E0E7A2BA64F93AC6A5656E5E867A7F180F9A0367#>
```

The Network Security Processor returns the following response:

```
<219#1KDDS000,4E3FC770A99CEB823F687E742304AF58BA682D1D839AEDFE,885A
96677B348379#F4B0#>
```

Export Working Key in TR-31 Format, versions A, B or C (Command 11A)

Command 11A translates a Working Key from encryption under the MFK to a TR-31 key block encrypted under the Key Block Protection Key (KBPK). Use this command to export a Working Key to another node that requires a TR-31 ANSI key block.

This command expands the functionality of command 118, which supports TR-31 version A. TR-31 version B and C are supported, and limited support is provided for version A. It adheres to the *X9 TR-31 2010, Interoperable Secure Key Exchange Key Block Specification for Symmetric Algorithms, December 9, 2010* specification.

note TR-31 version C is identical to version A, except that the values of some of the key headers have been clarified.

Option E1 must be enabled with the Byte 1, Key Usage value of the Working Key to be exported. For example, if a PIN Encryption Key (TR-31 key usage value = P0) is to be exported, option E1 must contain the letter “P”.

This command will support a 1key-3DES (single-length) Working Key if option 6C is enabled in the Network Security Processor’s security policy. The Key Block Protection Key cannot be a 1key-3DES (single-length) key.

This command is not enabled in the Network Security Processor’s default security policy. To use this command, you must add it to the Network Security Processor’s security policy.

Command

```
<11A#TR-31 Version#[TR-3 Key Usage]#[TR-31 Exportability]#
Header,EMFK.E(KBPK),MAC#Header,EMFK.E(Working Key),MAC#
Key Version Number#Number of Optional Blocks#[Optional Block Data]#>
```

Response

```
<21A#TR-31 Key Block#Working Key Check Digits#>[CRLF]
```

Calling Parameters

`11A`

Field 0, the command identifier.

`TR-31 Version`

Field 1, this field defines the TR-31 key block version. It must contain a single letter, “A”, “B”, or “C”.

`[TR-31 Key Usage]`

Field 2, this field must contain a value only when the Working Key, provided in field 5, is a MAC Key; the AKB header byte 1 contains the letter “M”. This field specifies what value will be used for the key usage bytes (5 and 6) of the TR-31 key block. You can specify any one of these three values: “M0”, “M1”, or “M3”. This field must be empty when the Working Key is not a MAC key,

`[TR-31 Exportability]`

Field 3, the value in this field is optional. If present, it specifies the value of the exportability byte (byte 11) in the TR-31 key block. You can specify any one of these three values: “E”, “N”, or “S”.

If this field is empty, the exportability byte defined in byte 4 of the AKB header of the Working Key, supplied in field 5, will be used as byte 11 of the TR-31 key block.

Header,$E_{MFK.E}$(KBPK),MAC

Field 4, the Key Block Protection Key in AKB format encrypted under the MFK. This key is used to derive the encryption and MAC keys that protect the Working Key. This field contains a 74 byte value, or a volatile table location. The length of the KBPK must be equal-to or greater-than the length of the Working Key being translated. For example, if the Working Key is a 2key-3DES key, this field must contain either a 2key-3DES or 3key-3DES key, or a reference to a volatile table location that contains a 2key-3DES or 3key-3DES key. The following headers are supported: 1kDEE000 and 1kDEN000. If option E2 is enabled, these additional headers are supported: 1kDNE000 and 1kDNN000.

Header,$E_{MFK.E}$(Working Key),MAC

Field 5, the Working Key in AKB format encrypted under the MFK. This field must contain a 74 byte value. Byte 4 of the AKB header cannot contain the letter “N”. The AKB header is used to derive the TR-31 header, see TR-31 header value derived from AKB header.

`Key Version Number`

Field 6, the key version number is a two-byte field. You cannot export a key component, therefore the first character in this field cannot contain the lowercase letter “c”.

`Number of Optional Blocks`

Field 7, the number of Optional Blocks to be included in the TR-31 Key Block. The minimum value is 0 and the maximum is 99, but it is also limited by the maximum number of bytes in the entire Key Block.

`[Optional Block Data]`

Field 8, this field must be empty when field 7 contains the number zero. When field 7 contains a value in the range of 1 through 99, this field must contain the optional block data. A maximum of 3912 characters can be provided in this field.

Each optional block consists of three fields:

- a two-byte ASCII hexadecimal identifier that indicates the content of the optional block,
- a two-byte ASCII hexadecimal length that indicates the total number of bytes in all of the optional block fields,
- the ASCII hexadecimal variable-length optional block data.

See Table A-8 in the *X9 TR-31 2010, Interoperable Secure Key Exchange Key Block Specification for Symmetric Algorithms, December 9, 2010* specification for a list of valid identifiers.

This command does not separate the optional block data into different fields. It concatenates all the data blocks, including any necessary padding blocks. The Network Security Processor will not check that the identifiers or contents are correct, it will check that the number of blocks and lengths are well-formed and that the total length is a multiple of the 3DES block size.

Table 3-45 Command 11A: Export Working Key in TR-31 Format, versions A, B or C

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	11A
1	TR-31 Version	1	A, B, or C
2	[TR-31 Key Usage]	0, 2	M0, M1, or M3
3	[TR-31 Exportability]	0, 1	E, N, or S
4	Header,$E_{MFK.E}$(KBPK), MAC*	74	printable ASCII
5	Header,$E_{MFK.E}$(Working Key), MAC	74	printable ASCII
6	Key Version Number	2	printable ASCII
7	Number of Optional Blocks	1-2	0 - 99
8	[Optional Block Data]	0 - 3912	printable ASCII

* Can be a volatile table location

Responding Parameters

`21A`

Field zero, the response identifier.

`TR-31 Key Block`

Field 1, the Working Key in TR-31 format encrypted under the KBPK.

`Working Key Check Digits`

Field 3, the first four digits that result from encrypting zeros using the Working Key. If Option 88 is enabled, six check digits will be returned.

Table 3-46 Response 21A: Export Working Key in TR-31 Format, versions A, B or C

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	21A
1	TR-31 Key Block	88 - 4000	printable ASCII
2	Working Key Check Digits	4, 6	0 - 9, A - F

Usage Notes

- Both the sending and receiving nodes must have the same KBPK value.
- The Master File Key must be a 3key-3DES key if the KBPK is a 3key-3DES key.
- If Byte 3 of the AKB Working Key header defined in field 5 of the command contains E, D, G, or V, the opposite value will be used for byte 8 in the TR-31 header. For example, if an Encrypt-Only KPE header is defined in field 5 with a header value of 1PUEE000, byte 8 of the TR-31 key header will contain D indicating that this key can only be used to decrypt PINs.
- The value supplied in field 1 of the command will be byte 0 of the TR-31 key block.
- The value supplied in field 2 of the command will be bytes 5 and 6 of the TR-31 key block.
- The TR-31 key block, which is field 1 of the response, contains random data, therefore the actual response will be different than shown in the following examples.
- Do not use this command to export HMAC-SHA1 or HMAC-SHA256 keys.
- Not all TR-31 headers can be derived from AKB headers. The following table defines the TR-31 headers which are derived from the AKB headers. The AKB header byte 4 (Exportability) will be used as byte 11 in the TR-31 header, this is indicated by the "*" character in the byte 4 of the AKB header in the following table.

Table 3-47 TR-31 header value derived from AKB header

AKB Header	Definition	TR-31 Header bytes 5 - 8
1dDN*000	BDK Base Derivation Key	B0TX
1iDN*000	DUKPT Initial PIN Encryption Key	B1TX
1CDN*000	CVK Card Verification Key	C0TC

Table 3-47 TR-31 header value derived from AKB header (continued)

AKB Header	Definition	TR-31 Header bytes 5 - 8
1CDG*000	CVK Card Verification Key - generate only	C0TV
1CDV*000	CVK Card Verification Key - verify only	C0TG
1DDN*000	Data Encryption using ECB, CBC, CFB, OFB, CCM, or CTR	D0TB
1DDE*000	Data Encryption using ECB, CBC, CFB, OFB, CCM, or CTR - encrypt only	D0TD
1DDD*000	Data Encryption using ECB, CBC, CFB, OFB, CCM, or CTR - decrypt only	D0TE
1mEN*000	EMV chip card Master Key: Application Cryptogram	E0TX
1mEN*00E	EMV chip card Master Key: Confidentiality	E1TX
1mEN*00M	EMV chip card Master Key: Integrity	E2TX
1IDN*000	Initialization Vector	I0TN
1KDN*000	Key Encryption or Wrapping	K0TB
1KDD*000	Key Encryption or Wrapping - decrypt, unwrap only	K0TE
1KDE*000	Key Encryption or Wrapping - encrypt, wrap only	K0TD
1kDN*000	TR-31 Key Block Protection Key	K1TB
1kDD*000	TR-31 Key Block Protection Key - decrypt, unwrap only	K1TE
1kDE*000	TR-31 Key Block Protection Key - encrypt, wrap only	K1TD
1MDN*000	MAC Key - generate and verify	M0TC, M1TC, or M3TC
1MDG*000	MAC Key - generate only	M0TV, M1TV, or M3TV
1MDV*000	MAC Key - verify only	M0TG, M1TG, or M3TG
1PDN*000 or 1PUN*000	PIN Encryption	P0TB
1PDE*000 or 1PUE*000	PIN Encryption - encrypt only	P0TD
1PDD*000 or 1PUD*000	PIN Encryption - decrypt only	P0TE
1V3N*000	PIN Verification, IBM 3624	V1TC
1V3G*000	PIN Verification, IBM 3624 - generate only	V1TV
1V3V*000	PIN Verification, IBM 3624 - verify only	V1TG
1VVN*000	PIN Verification, Visa	V2TC
1VVG*000	PIN Verification, Visa - generate only	V2TV
1VVV*000	PIN Verification, Visa - verify only	V2TG

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Export TR-31 key block - no optional block, key variant method

- Clear-text Key Block Protection Key (KBPK):
  89E88CF7931444F3 34BD7547FC3F380C, check digits = D1D8
  The KBPK in AKB format:
  1kDEN000,5B7C53C30F58DB9ACABACED0895037A846283A85A3F312F4,151E433A
  FDF3492E

- Clear-text PIN Encryption Key to be exported:
  F039121BEC83D26B 169BDCD5B22AAF8F, check digits = CB9D
  The KPE-Decrypt Only Key in AKB format:
  1PDDE000,D966234DDB60482A2E4B8658872B82166AF9595EA4085E67,8142BA0E
  54806DC2

- Key Version Number: 00

The command looks like this:

```
<11A#A##1kDEN000,5B7C53C30F58DB9ACABACED0895037A846283A85A3F312F4,1
51E433AFDF3492E#1PDDE000,D966234DDB60482A2E4B8658872B82166AF9595EA4
085E67,8142BA0E54806DC2#00#0##>
```

The Network Security Processor returns a response similar to this:

```
<21A#A0088P0TE00E00007DD4DD9566DC0E2F956DCAC0FDE9153159539373E9D82D
3CD4AFD305A7EF1BA67FE03712#CB9D#>
```

Export TR-31 key block - no optional block, key derivation method

- Clear-text Key Block Protection Key (KBPK):
  89E88CF7931444F3 34BD7547FC3F380C, check digits = D1D8
  The KBPK in AKB format:
  1kDEN000,5B7C53C30F58DB9ACABACED0895037A846283A85A3F312F4,151E433A
  FDF3492E

- Clear-text PIN Encryption Key to be exported:
  F039121BEC83D26B 169BDCD5B22AAF8F, check digits = CB9D
  The KPE-Decrypt Only Key in AKB format:
  1PDDE000,D966234DDB60482A2E4B8658872B82166AF9595EA4085E67,8142BA0E
  54806DC2

- Key Version Number: 00

The command looks like this:

```
<11A#B##1kDEN000,5B7C53C30F58DB9ACABACED0895037A846283A85A3F312F4,1
51E433AFDF3492E#1PDDE000,D966234DDB60482A2E4B8658872B82166AF9595EA4
085E67,8142BA0E54806DC2#00#0##>
```

The Network Security Processor returns a response similar to this:

```
<21A#B0096P0TE00E00001376E82DA3FDEB6708799D1EA6D88CFB00000000000000
0000000000000000EC50889560986D9F#CB9D#>
```

Export TR-31 key block - with optional block, key derivation method

- Clear-text Key Block Protection Key (KBPK):
  89E88CF7931444F3 34BD7547FC3F380C, check digits = D1D8
  The KBPK in AKB format:
  1kDEN000,5B7C53C30F58DB9ACABACED0895037A846283A85A3F312F4,151E433AFDF3492E
- Clear-text PIN Encryption Key to be exported:
  F039121BEC83D26B 169BDCD5B22AAF8F, check digits = CB9D
  The KPE-Decrypt Only Key in AKB format:
  1PDDE000,D966234DDB60482A2E4B8658872B82166AF9595EA4085E67,8142BA0E54806DC2
- Key Version Number: 12
- Optional Key Block: KS1400604B120F929280

The command looks like this:

```
<11A#B##1kDEN000,5B7C53C30F58DB9ACABACED0895037A846283A85A3F312F4,1
51E433AFDF3492E#1PDDE000,D966234DDB60482A2E4B8658872B82166AF9595EA4
085E67,8142BA0E54806DC2#12#1#KS1400604B120F929280#>
```

The Network Security Processor returns a response similar to this:

```
<21A#B0120P0TE12E0100KS1400604B120F929280PB04B7E4A203A8C0014C88CC99
ADE0C9F5D2000000000000000000000000000000003BA0FDDA7EC36D8C#CB9D#>
```

Import Non-AKB Formatted Working Keys (Command 11B)

Command 11B imports a Working Key that is encrypted under a Key Exchange Key to encryption under the Master File Key. The imported Working Key is returned in the Atalla Key Block format. The header for the Key Exchange Key defines the header to be assigned to the Working Key.

Option E0 must be enabled with the Byte 1, Key Usage value of the Working Key being imported. This command can import a 1key-3DES (single-length) key only if 6C is enabled.

This command is not enabled in the Network Security Processor’s default security policy. To use this command, you must add it to the Network Security Processor’s security policy.

Command

```
<11B#Variant#E_KEK.v(WK)#Header,E_MFK.E(KEK),MAC#>
```

Response

```
<21B#Header,E_MFK.E(WK),MAC#Working Key Check Digits#>[CRLF]
```

Calling Parameters

`11B`

Field 0, the command identifier.

`Variant`

Field 1, the variant of the Key Exchange Key used to encrypt the Working Key specified in field 2. This field can contain a one or two byte decimal value in the range of 0 through 31. Enter zero in this field if no variant was applied to the KEK prior to encrypting the Working Key.

`E_KEK.v(WK)`

Field 2, the Working Key encrypted under the Key Exchange Key. If a variant was specified in field 1, it will be applied to the Key Exchange Key before decrypting the Working Key.

`Header,E_MFK.E(KEK),MAC`

Field 3, the Key Exchange Key encrypted under the MFK. This key is used to decrypt the Working Key. This field contains a 74 byte value, or a volatile table location.

This header is used to define the header assigned to the Working Key when it is converted to AKB format. The Key Exchange Key header byte 6 must contain “I”. For example, if you are importing a PIN Encryption Key, the Key Exchange Key header would be 1PUNE0I0. See Working Key Headers for a list of supported Working Key headers.

If header byte 6 contains the value K, the AKB returned in field 1 of the response will have header byte 6 set to I.

If option E4 is enabled, and the rightmost three bytes of the Key Exchange Key header are “0IS”, the rightmost three bytes of the imported Working Key header, returned in the response, will be “0dS”. For example, if the Key Exchange Key header is 1PUNN0IS, the header of the imported Working Key will be 1PUNN0dS.

The length of the Key Exchange Key must be equal-to or greater-than the length of the Working Key being imported. For example, if the Working Key is a 2key-3DES key, this field must contain either a 2key-3DES or 3key-3DES key.

Table 3-48 Command 11B: Import Non-AKB Keys

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	11B
1	Variant	1-2	0 - 31
2	$E_{KEK.v}$(WK)	16, 32, 48	0 - 9, A - F
3	Header,$E_{MFK.E}$(KEK),MAC	74	printable ASCII

Responding Parameters

`21B`

Field 0, the response identifier.

`Header,`$E_{MFK.E}$`(WK),MAC`

Field 1, the translated Working Key encrypted under the MFK. This field contains a 74 byte value. Byte 6 of the Working Key header will contain the value “0” (zero). The remaining header bytes will be the same as those specified in field 3 of the command.

`Working Key Check Digits`

Field 2, the check digits of the imported Working Key. This field will contain the first four digits of the result of encrypting zeros using the Working Key. If option 88 is enabled, this field will contain six check digits.

Table 3-49 Response 21B: Import Non-AKB Keys

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	21B
1	Header,$E_{MFK.E}$(WK),MAC	74	printable ASCII
2	Working Key Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- Generate the import KEK. That is make sure the KEK header is correct for the type of key being imported.
- Option E0 must specify the key type to be imported.
- Do not use this command to import HMAC-SHA1 or HMAC-SHA256 keys.
- In version 1.42 and above, the requirement for KEK header byte 4 to contain the letter "N" has been removed.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Import a Data Encryption Key (KD) encrypted under variant 2 of the Key Exchange Key

- Clear-text Data Encryption Key (KD): 0123 4567 89AB CDEF.
  The Data Encryption Key encrypted under variant 2 of the KEK:
  A30F FFDC 12C5 884D
- Clear-text Key Exchange Key (KEK): 0123456789ABCDEF FEDCBA9876543210
  The KEK in AKB format:
  1DDNN0I0,69AFDA80E06E2F4E30D6D1EDACCBB36E7BEFAE872CCC14F7,9D6FCC8892846DC5

The command looks like this:

```
<11B#2#A30FFFDC12C5884D#1DDNN0I0,69AFDA80E06E2F4E30D6D1EDACCBB36E7B
EFAE872CCC14F7,9D6FCC8892846DC5#>
```

The Network Security Processor returns the following response:

```
<21B#1DDNN000,266329B0C6995D0966505F61B46C3704051FAE9C45D341FD,1E39
CD9730F6EB8E#D5D4#>
```

Import a Card Verification Value Key pair encrypted under no variant of the Key Exchange Key

- Clear-text Card Verification Key pair: 0123456789ABCDEF FEDCBA9876543210
  The CVV key pair encrypted under variant 0 of the KEK: 1A4D672DCA6CB335 1FD1B02B237AF9AE

- Clear-text Key Exchange Key (KEK): 0123456789ABCDEF FEDCBA9876543210
  The KEK in AKB format:
  1CDNN0I0,85AF1E727D17562121717572022F030FD29CF88EFAFD066F,9DE20F3E1A74F60C

The command looks like this:

```
<11B#0#1A4D672DCA6CB3351FD1B02B237AF9AE#1CDNN0I0,85AF1E727D17562121
717572022F030FD29CF88EFAFD066F,9DE20F3E1A74F60C#>
```

The Network Security Processor returns the following response:

```
<21B#1CDNN000,64A883D036BBEF32BF146E43A1BC6DF0B1264D674A68E267,88D8
8EA266E7D54F#08D7#>
```

Convert Atalla DES Key to 3DES AKB Format (Command 11C)

Command 11C is used to convert an encrypted 3DES key to encryption, in AKB format, under the current MFK. This command should only be used to convert a 3DES key, encrypted under the MFK (old) in the variant Network Security Processor, to AKB format.

The value of the MFK (old) in the variant Network Security Processor can be provided, in AKB format, as the optional field five. If field five is not present, the value of the MFK (old) in the variant Network Security Processor must be key 1 and key 2 of the current MFK, it is not provided as input to this command.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

This command is a high security exposure it is not enabled in the Network Security Processor's default security policy.

You must specify the number of command executions to be performed in the Network Security Processor. This is accomplished using either the SCA, or commands 108 and 109 and the SCA.

caution Once the count value reaches zero, the Network Security Processor will return an error <00#0300xx#> instead of processing the command. The command <9A#COUNT#> can be used to obtain the current count value for all commands in the command count table.

When the key database has been converted to AKB format, this command should be removed from the Network Security Processor's security policy.

Command

```
<11C#Variant#Header#EMFK.V(Key)[,EMFK.V(Key2)]#
Check Digits[,Check Digits of Key2]#[Header,EMFK.E(MFK),MAC#]>
```

Response

```
<21C#Header,EMFK.E(Key),MAC#>[CRLF]
```

Calling Parameters

`11C`

Field 0, the command identifier.

`Variant`

Field 1, the variant of the MFK used to encrypt the Working Key specified in field 3. This field can contain a one or two byte decimal value in the range of 0 through 31.

`Header`

Field 2, the header to be used when generating the Atalla Key Block. The header specified in this field must be a valid header based on the variant, see Working Key Headers for a list of available headers based on the variant and key type.

Header bytes 6 and 7 can also contain the values K and n, respectively.

Header byte 1 value of “I” is associated with variant 6, and a header byte 1 value of “d” is associated with variant 8.

$E_{MFK.V}$(Key)[,$E_{MFK.V}$(Key2)]

Field 3, the cryptogram of the Working Key encrypted under the variant of the MFK. In the variant version of the Network Security Processor, the KCVV1 and KCVV2 keys which are used to generate and verify CVVs and CVCs, were sent to the Network Security Processor as two single-length keys. Similarly, the KeyLeft and KeyRight keys which are used to generate and verify PVVs, were sent to the Network Security Processor as two single-length keys. Each single-length key had its own set of check digits. Once inside the variant version of the Network Security Processor, these keys were used as a double-length key. In AKB, these keys are double-length and there is only set of check digits. When importing these single-length keys, where field two contains one of the following headers 1CDNE000, 1CDVE000, 1CDGE000, 1VVNE000, 1VVGE000, or 1VVVE000, this field should contain both encrypted single-length key cryptograms separated by a comma.

`Check Digits[,Check Digits of Key2]`

Field 4, the check digits of the Working Key. When importing a single-length keys, where field two contains one of the following headers 1CDNE000, 1CDVE000, 1CDGE000, 1VVNE000, 1VVGE000, or 1VVVE000, this field should contain check digits for both single-length keys separated by a comma. This field must contain the four digits of the result from encrypting zeros using the Working Key(s). If option 88 is enabled, this field must contain the first six digits of the result from encrypting zeros using the Working Key(s).

[Header,$E_{MFK.E}$(MFK),MAC#]

Field 5, this field is optional. If present, the key in this field will be used to decrypt the variant encrypted Working Key(s) in field 3. The header for this field must be 1KDDN0M0.

Table 3-50 Command 11C: Convert Atalla DES Key to 3DES AKB Format

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	11C
1	Variant	1, or 2	0-31
2	Header	8	Printable characters
3	$E_{MFK.V}$(Key)[,$E_{MFK.V}$(Key2)]	16, 32 or 33	0 - 9, A - F

Table 3-50 Command 11C: Convert Atalla DES Key to 3DES AKB Format (continued)

Field	Contents	Length (bytes)	Legal Characters
4	Check Digits[,Check Digits of Key2]	4 or 6, 9 or 13	0 - 9, A - F
5	[Header, $E_{MFK.E}$(MFK), MAC#]	0, 74	Printable characters

Responding Parameters

```
21C
```

Field 0, the response identifier.

Header,$E_{MFK.E}$(Key),MAC

Field 1, the Working Key in AKB format.

Table 3-51 Response 21C: Convert Atalla DES Key to 3DES AKB Format

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	21B
1	Header,$E_{MFK.E}$(Key),MAC	74	printable ASCII

Usage Notes

- If the current MFK does not match key blocks 1 and 2 of the MFK (old) in the variant Network Security Processor, include field 5 in the command. An example of this situation would be if the MFK (old) in the variant Network Security Processor were a single-length, or replicated single-length, key and the current MFK were a 2key- or 3key-3DES key. If key blocks 1 and 2 are the same value as the MFK (old) in the variant Network Security Processor, do not include field 5 in the command.
- Do not use this command to convert HMAC-SHA1 or HMAC-SHA256 keys.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Key blocks 1 and 2 of current MFK are the same value as key blocks 1 and 2 of the MFK in the variant Network Security Processor

In this scenario, field 5 is not included in the command.

Convert a Data Encryption Key

- Clear-text Data Encryption Key (KD): 0123 4567 89AB CDEF, check digits = D5D4. The Data Encryption Key encrypted under variant 2 of the 2key MFK: 80BC DEAC 5703 BC84

The command looks like this:

```
<11C#2#1DDNE000#80BCDEAC5703BC84#D5D4#>
```

The Network Security Processor returns the following response:

```
<21C#1DDNE000,4509C24B1C13C3798D30B3AEAEB65DE1CEFB3E831C522F7F,1CE6
6C3D258AF755#>
```

Convert a Visa KPV Key pair

- Clear-text KeyLeft (KPVL): 0123 4567 89AB CDEF, check digits = D5D4
  The KeyLeft encrypted under variant 4 of the 2key MFK: 66FC 7000 6073 B1BF
- Clear-text KeyRight (KPVR): FEDC BA98 7654 3210 A68C, check digits = A68C. The KeyRight encrypted under variant 4 of the 2key MFK: E474 700E DE15 5096

The command looks like this:

```
<11C#4#1VVNE000#66FC70006073B1BF,E474700EDE155096#D5D4,A68C#>
```

The Network Security Processor returns the following response:

```
<21C#1VVNE000,F78F26AB6F5763645E41E3FC8EC48F0C68F828FB78FB5AF9,3FA7
A4230E019EA0#>
```

Key blocks 1 and 2 of current MFK are NOT the same value as key blocks 1 and 2 of the MFK in the variant Network Security Processor

In this scenario, field 5 is included in the command.

Convert a KPV when the variant Network Security Processor's MFK is a Single-Length Key

- Clear-text single-length variant Network Security Processor's MFK:
  0123 4567 89AB CDEF, check digits = D5D4
- The AKB of the single-length variant Network Security Processor's MFK:
  1KDDN0M0,C5BDA2BA68D68BDBF97CC1877FA63E2F3869C49DA55C2EE3,99FCB3C
  B7F048281
- Clear-text KPV: 0123 4567 89AB CDEF, check digits = D5D4
  The KPV encrypted under variant 4 of the single-length MFK: 2BA1 112B AB3A 3FB6.

The command looks like this:

```
<11C#4#1V3NE000#2BA1112BAB3A3FB6#D5D4#1KDDN0M0,C5BDA2BA68D68BDBF97C
C1877FA63E2F3869C49DA55C2EE3,99FCB3CB7F048281#>
```

The Network Security Processor returns the following response:

```
<21C#1V3NE000,1C33AD320C016E5D580A3E4FE68A82EBB2B64FB342F9B811,70FD
84D875F0BA1C#>
```

Derive DUKPT Initial PIN Encryption Key (Command 38C)

Command 38C derives the Initial PIN Encryption Key (IPEK) from the supplied Base Derivation Key and Key Serial Number. The derived 2key-3DES IPEK is returned in AKB format.

You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<38C#Header,EMFK.E(BDK),MAC#KSN#>
```

Response

```
<48C#Header,EMFK.E(BDK),MAC#BDK Check Digits#IPEK Check Digits#>
[CRLF]
```

Calling Parameters

`38C`

Field 0, the command identifier.

`Header,EMFK.E(BDK),MAC`

Field 1, the Base Derivation Key encrypted under the MFK. This field contains a 74 byte value. This key must be a 2key-3DES key. The following headers are supported: 1dDNE000 and 1dDNN000.

`KSN`

Field 2, the Key Serial Number. This field must contain 10 through 20 hexadecimal characters, where the rightmost 21 bits are equal to zero. If this field contains less than 20 hexadecimal characters, the Network Security Processor will left-pad this value with Fs.

Table 3-52 Command 38C: Derive DUKPT Initial PIN Encryption Key

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	38C
1	Header,$E_{MFK.E}$(BDK),MAC	74	printable ASCII
2	KSN	10 - 20	0 - 9, A - F

Responding Parameters

`48C`

Field 0, the response identifier.

Header,$E_{MFK.E}$(IPEK),MAC

Field 1, the Initial PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value. This key is a 2key-3DES key. The AKB header will be 1iDNE000.

`BDK Check Digits`

Field 2, the first four hexadecimal characters of the result from encrypting zeros using the Base Derivation Key. If option 88 is enabled, this field will contain six check digits.

`IPEK Check Digits`

Field 3, the first four hexadecimal characters of the result from encrypting zeros using the Initial PIN Encryption Key. If option 88 is enabled, this field will contain six check digits.

Table 3-53 Response 48C: Derive DUKPT Initial PIN Encryption Key

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	3	48C
1	Header,$E_{MFK.E}$(IPEK),MAC	74	Printable ASCII
2	BDK Check Digits	4 or 6	0 - 9, A - F
3	IPEK Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- The rightmost 21 bits of the KSN supplied in field 2 must be zeros.
- This command generates a 2key-3DES Initial PIN Encryption Key, option A2 does not apply to this command.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text Base Derivation Key: 0123456789ABCDEF FEDCBA9876543210, check digits = 08D7B4
  The Base Derivation Key in AKB format:
  1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,342946FE884AA8B2
- KSN = 9876543210E00000

The command looks like this:

```
<38C#1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,3429
46FE884AA8B2#9876543210E00000#>
```

The Network Security Processor returns the following response:

```
<48C#1iDNE000,936AB7B89DA6FA2FA538B4190414CB6C94DE64CEB88C5B0F,E9D4
1BC1A0981CE3#08D7#AF8C#>
```

Generate AES or HMAC Symmetric Key (Command 39A)

Command 39A generates either an AES or HMAC symmetric key.

In version 1.50 and above, this command has been modified to generate AES keys.

AES keys can be 128, 192 or 256 bits in length.

HMAC keys can be variable length. Valid key sizes are a multiple of 8 bits, the minimum length is 80 bits and the maximum length is 256 bits.

You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<39A#Key Type#Header#Key Length#[AES Key Check Digit Method#]>
```

Response

<49A#Header,$E_{MFK.E}$(Key),MAC#Key Check Digits#>[CRLF]

Calling Parameters

`39A`

Field 0, the command identifier.

`Key Type`

Field 1, the type of key to be generated. To generate an AES key, this field must contain the uppercase letter J. To generate an HMAC key, this field must contain the uppercase letter V.

`Header`

Field 2, the header that will be assigned to the generated key.

For AES keys, the allowed headers are: 1DJNE000, 1DJNN000, 1DJEE000, 1DJEN000, 1DJDE000, 1DJDN000, 1JJNE000, and 1JJNN000.

For HMAC-SHA1 keys, the allowed headers are: 1MHNE000, 1MHNN000, 1MHGE000, 1MHGN000, 1MHVE000, and 1MHVN000.

For HMAC-SHA256 keys, the allowed headers are: 1M2NE000, 1M2NN000, 1M2GE000, 1M2GN000, 1M2VE000, and 1M2VN000.

`Key Length`

Field 3, the length, in bits, of the key to be generated. When field 1 contains the letter J, this field must be one of these three values: 128, 192, or 256. When field 1 contains the letter V, this field must contain a decimal value in the range of 80 through 256, and be a multiple of 8.

`[AES Key Check Digit Method#]`

Field 4, the AES key check digit method. This field is optional. It must be present and contain the letter 'H' when field 1 contains the letter J and the AES key check digits should be generated using check digit method H. If this field is not present, check digit method C will be used to generate the AES key check digits.

note HMAC key check digits are always generated using check digit method H.

Table 3-54 Command 39A: Generate AES or HMAC Symmetric Key

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	39A
1	Key Type	1	V
2	Header	8	printable ASCII
3	Key Length	2-3	80 - 256
4	[AES Key Check Digit Method#]	0, 1	H

Responding Parameters

`49A`

Field 0, the response identifier.

Header,$E_{MFK.E}$(Key),MAC

Field 1, the generated key in AKB format. The length of this field is based on field 3 of the command.

Key Length (in bits)	AES AKB Length	HMAC AKB Length
80 - 176	58 (128-bit key)	74
184 - 240	74 (192-bit key)	90
248 - 256	106 (256-bit key)	106

`Key Check Digits`

Field 2, check digits of the generated key. Check digit method H will SHA-256 hash the clear key without any padding. For AES keys, the check digit will be the entire hash, 64 hexadecimal digits. For HMAC keys, the check digits will be the leftmost 14 digits of the hash. Check digit method C will calculate the MAC of an all zero block using the CMAC algorithm. The check digit will be the leftmost 10 digits of the result.

Table 3-55 Response 49A: Generate AES or HMAC Symmetric Key

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	49A
1	Header,$E_{MFK.E}$(Key),MAC	58, 74, 90, 106	printable ASCII
2	Key Check Digits	10, 14, 64	0 - 9, A - F

Usage Notes

- If the HMAC key to be generated is greater than 256 bytes, call this command twice, and then supply both of the resulting AKBs in the commands 39B or 39C. For example, if the required key size is 384 bits, specify 256 bits in the first command and specify 128 bits in the second command. Use the AKB returned in the response to the first command in field 1 and use the AKB returned in the response to the second command in field 2 of the commands 39B or 39C.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

AES 256-bit key - check digit method H

- Header: 1DJNE000
- The key length: 256-bits

The command looks like this:

```
<39A#J#1DJNE000#256#H#>
```

The Network Security Processor returns a response similar to this:

```
<49A#1DJNE000,28F5E07D53295DF3261BCB48A68FF4A9A3E75D81A8372B00BD583
436266FFBD8,B4295FEB34CD3C7F#29A6DC7889F40961D741E9CCED10B6E1378AA0
853C1D5814212BC11DF83E5F47#>
```

HMAC 256-bit key

- Header: 1MHNE000
- The key length: 256-bits

The command looks like this:

```
<39A#V#1MHNE000#256#>
```

The Network Security Processor returns a response similar to this:

```
<49A#1MHNE000,F05883DE1887118B2888AB6686B62D386333B63DBEC25CBD69B36
09A660703AB051AF43F9DF29584,58643450A27C2CCF#79267CA84AF6B3#>
```

Generate Check Digits for an AES Key (Command 392)

Command 392 supports CMAC and SHA-256 algorithms to generate check digits for an AES key.

This command is not enabled in the Network Security Processor’s default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor’s security policy.

Command

```
<392#Method#Reserved#Header,EMFK.E(KD),MAC#>
```

Response

```
<492#Method#Check Digits#>[CRLF]
```

Calling Parameters

`392`

Field 0, the command identifier.

`Method`

Field 1, the method used to calculate the check digits of the Data Key. The following table indicates the method that the Network Security Processor supports and the value to enter in this field for each method:

Value	Method
C	CMAC_{KD}(Block of zeros)
H	SHA-256 (KD)

`Reserved`

Field 2, this field must be empty.

`Header,EMFK.E(KD),MAC`

Field 3, the Data Key (KD) encrypted under the MFK. This field contains a 58, 74 byte value which is required to support a 128-, 192- or 256-bit key. Only Data Keys with the following headers are supported: 1DJNE000, 1DJNN000, 1DJEE000, 1DJEN000, 1DJDE000, and 1DJDN000.

Table 3-56 Command 392: Generate Check Digits for an AES Key

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	392
1	Check Digit Method	1	C or H
2	Reserved	0	none
3	Header,$E_{MFK.E}$(KD),MAC	58, 74, 90	printable ASCII

Responding Parameters

`492`

Field 0, the response identifier.

`Method`

Field 1, the check digit method. This field will contain the same value as field 1 of the command.

`Check Digits`

Field 2, check digits of the Data Key. The check digits are generated using the algorithm specified in field 8 of the command. If the check digit algorithm is C (CMAC), this field will contain 10 hexadecimal characters. If the check digit algorithm is H (SHA-256), this field will contain 64 hexadecimal characters.

Table 3-57 Response 492: Generate Check Digits for an AES Key

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	492
1	Method	1	C, H
2	KD Check Digits	10 or 64	0 - 9, A - F

Usage Notes

- Before using this command, generate the AKB for the Data Key.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

CMAC check digits

- Method: C
- Data Key (128 bits): 0123456789ABCDEFFEDCBA9876543210
  The Data Key in AKB format:
  1DJNE000,454EB64571576C185EA9D338840CD76F,D669BF4E9D039D94

The command looks like this:

```
<392#C##1DJNE000,454EB64571576C185EA9D338840CD76F,D669BF4E9D039D94#>
```

The Network Security Processor returns the following response:

```
<492#C#2090A67375#>
```

SHA-256 check digits

- Method: H
- Data Key (128 bits): 0123456789ABCDEFFEDCBA9876543210
  The Data Key in AKB format:
  1DJNE000,454EB64571576C185EA9D338840CD76F,D669BF4E9D039D94

The command looks like this:

```
<392#H##1DJNE000,454EB64571576C185EA9D338840CD76F,D669BF4E9D039D94#>
```

The Network Security Processor returns the following response:

```
<492#H#411D3F1D2390FF3F482AC8DF4E730780BB081A192F283D2F373138FD101D
C8FE#>
```

4

Processing Personal Identification Numbers

This section outlines the tasks involved in processing PINs and describes the PIN processing commands supported by the Network Security Processor.

To skip this introduction, go to Table 4-10 on page 4-18 for a list of commands.

About PIN Processing

The personal identification number – or PIN – is the secret, unique number that identifies a consumer who is transacting business on an automated teller machine (ATM) or point of sale (POS) network.

The following list outlines the processes that a PIN typically undergoes, starting with its entry into an ATM or PIN pad and ending with its verification by the issuing host.

1. The PIN is entered into an ATM or PIN pad.
2. The ATM or PIN pad formats the PIN into a PIN block.
3. The ATM or PIN pad encrypts the PIN block and sends it to the host.
4. The host determines whether the PIN corresponds to an account that belongs to its own financial institution or to another institution.
    a. If the PIN corresponds to an account at this financial institution (making it an “on-us” transaction), the host verifies the PIN and confirms whether sufficient funds are available for the requested transaction.
    b. If the PIN does not correspond to an account at this financial institution (making it a “not-on-us” transaction), the host translates the PIN and sends it to the switch encrypted under the acquirer’s working key. The switch determines the issuing financial institution, translates the PIN block, and then sends it to another switch or to the issuing financial institution encrypted under the issuer working key. When the PIN block arrives at the issuer, the host verifies it and confirms whether sufficient funds are available for the requested transaction.

The following section explains the programming tasks that you must accomplish to facilitate PIN processing.

PIN Processing Tasks

Processing PINs typically involves the following tasks.

- Encrypting PINs or PIN blocks
- Translating PIN blocks
- Verifying incoming PIN blocks and authorizing or denying transaction requests

Encrypting PINs

This subsection explains how PINs are encrypted in ATM networks and VISA DUKPT POS networks.

PIN Encryption in ATM Networks

In ATM networks, PINs can be encrypted at two different places:

- The point of capture (an ATM) or
- The host’s Network Security Processor.

Encrypting PINs at an ATM involves two steps:

1. The ATM formats the PIN into a **PIN block**. PIN blocks are packages of data that contain the PIN, pad characters, and sometimes other information like the length of the PIN. The Network Security Processor supports a variety of PIN block types. They are described in PIN Block Types later in this section.
2. Once the PIN has been formatted into a PIN block, the ATM encrypts it using a PIN Encryption Key that is common to both the ATM and its host. To encrypt PINs at the host’s Network Security Processor, the clear-text PIN must travel from the ATM to the host. If the host is unable to verify the PIN, the PIN is formatted into a PIN block, and then encrypted using a PIN Encryption Key. Formatting and encrypting the PIN enables it to be transmitted to a node that can verify it.

The point to remember is that PINs never pass to the switch in clear-text format when they have passed first through an intercepting processor.

PIN Encryption In VISA DUKPT Networks

In networks that use VISA DUKPT key management, PIN pads are always responsible for encrypting PINs. The difference between PIN encryption on VISA DUKPT networks and PIN encryption at the ATM is that on VISA DUKPT networks, the PIN Encryption Key is unique for every transaction.

Translating PIN Blocks

Once an ATM or PIN pad receives a PIN, the objective is to verify that it corresponds to a valid account. If this verification is not done at the ATM or PIN pad, the PIN block must travel to the host or switch to be verified. If the PIN is verified at the switch or issuer host, the PIN block must be **translated** each time it stops at an intermediary, or intercept, processor. Translation refers simply to the process of changing the PIN block's type or the PIN Encryption Key in use so that the PIN block can travel from one processor to the next. Typically, the first intercept processor receives the PIN block encrypted in the type supported by the sending ATM or PIN pad, and then translates it into an ANSI PIN block. Most networks require ANSI PIN blocks.

Verifying Incoming PIN Blocks

PINs are not verified directly. The PIN is known only to the card holder; no one else – not even the issuing financial institution – knows the PIN's clear-text value. PIN verification is facilitated by means of the **PIN verification number (PVN)**. The PIN verification number is derived from an algorithm that takes as its input the PIN and the Primary Account Number (PAN). The result is operated on by the PIN Verification Key; the result is a calculated PIN verification number. The PIN verification number calculated at the verifying node is compared to the PVN that is encoded on consumer's credit or debit card, or stored on a host database. If the two values match, the PIN has been verified.

The Network Security Processor supports the following standard methods of PIN verification:

- Identikey
- IBM 3624
- Visa
- Atalla DES Bilevel
- Diebold
- NCR
- Burroughs
- Atalla 2x2

PIN Sanity Error

When an encrypted PIN is translated or verified, it is decrypted with the incoming PIN Encryption Key. The Network Security Processor examines the format of the decrypted PIN block. Option 4B specifies the type of PIN sanity test to be performed. If the Network Security Processor determines that the decrypted PIN block is not valid, it returns a PIN Sanity error in the response. The usual causes of PIN sanity errors are:

- The wrong key was specified as the incoming PIN Encryption Key.
- The PIN length is incorrect. Option A0 can be used to configure the Network Security Processor for a specific minimum PIN length, the default is 4 digits. The maximum PIN length is fixed at 12 digits. If the decrypted PIN does not fall within minimum and maximum range, a sanity error will be returned. Option A1 can be used to configure the Network Security Processor to return an “L” if the decrypted PIN is less than the minimum PIN length. The Network Security Processor does not allow a PIN greater than 12 digits. When the Network Security Processor decrypts a PIN that is greater than 12 digits, it will return “N” even if option A1 is set to “L”.
- Wrong data in the data block. For example, the ANSI PIN block requires the rightmost 12 digits of the account number, excluding the check digit. If the origin and destination do not use the exact same 12 digits in the data block, a sanity error will be returned.

PIN Block Types

The Network Security Processor supports a variety of PIN block types; not all PIN block types are supported in all commands. Each PIN block type requires a specific set of data. This data is provided as separate fields at the end of the command. Each of these extra fields is delimited with a “#”, just like any other field in the command.

PIN Block Type	Value	PIN Block Data fields added to the end of the command
ANSI PIN Block	1	1, labeled Field A
IBM 3624 PIN Block	2	3, labeled Fields A, B, and C
PIN/Pad PIN Block	3	2, labeled Fields A and B
Docutel PIN Block	3	2, labeled Fields A and B
IBM Encrypting PIN Pad PIN Block	4	1, labeled Field A
Burroughs PIN Block	5	2, labeled Fields A and B
VISA Derived Unique Key Per Transaction PIN Block	7	3, labeled Fields, A, B, and C
ISO-3 PIN Block	8	1, labeled Field A

PIN Block Type	Value	PIN Block Data fields added to the end of the command
IBM 4731 PIN Block	9	4, labeled Fields A, B, C, and D
ISO-2 PIN Block	C	0

The following sections define the contents of the PIN Block Data for each supported PIN block type.

ANSI PIN Block

The ANSI PIN block is also referred to as an ISO-0 PIN block.

PIN Block Data

The ANSI PIN Block requires one PIN Block Data field; the last field of the command.

Table 4-1 ANSI - PIN Block Data

Field	Contents	Length (bytes)	Legal Characters
A	Twelve rightmost PAN digits (excluding check digits)	12	0 - 9

Constructing an ANSI PIN Block

The ANSI PIN block is the result of performing an exclusive-OR on two data blocks, the PIN block and the account number block.

```
C  N  P  P  P  P/F  P/F  P/F  P/F  P/F  P/F  P/F  P/F  F  F
```

Figure 4-1 PIN Block

C

The control field. A four bit value; hexadecimal 0.

N

The length of the PIN. A four bit hexadecimal value 4 to 9, A, B, or C. A ten digit PIN is represented as A, an 11 digit PIN is represented as B, and a 12 digit PIN is represented as C.

P

PIN digit. A four bit hexadecimal value in the range of 0 through 9.

F

Pad character. A four bit value; hexadecimal F.

`P/F`

A PIN digit or a pad character, depending on the length of the PIN.

0	0	0	0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12

Figure 4-2 ANSI Account Number Block

`0`

Pad character. A four bit value; hexadecimal 0.

`A1 to A12`

The 12 rightmost digits of the Primary Account Number (PAN), **excluding** the check digit. A1 is the most significant digit; A12 is the digit that immediately precedes the Primary Account Number's check digit.

Example

```
PIN = 1234

Primary Account Number = 5999997890123457

PIN Block              = 041234FFFFFFFFFF

Account Number Block   = 0000999789012345

exclusive-OR
the PIN and
Account Number Blocks

ANSI PIN Block         = 0412AD6876FEDCBA
```

IBM 3624 PIN Block

This encrypted PIN block is 18 hexadecimal characters. When a command contains an encrypted IBM 3624 PIN block, the last field of the Network Security Processor's response will be the two digit sequence number.

PIN Block Data

The IBM 3624 PIN block requires three PIN Block Data fields; the last three fields of the command.

Table 4-2 IBM 3624 - PIN Block Data

Field	Contents	Length (bytes)	Legal Characters
A	Pad character*	1	0 - 9, A - F, X, W

Table 4-2 IBM 3624 - PIN Block Data (continued)

Field	Contents	Length (bytes)	Legal Characters
B	Twelve digit; required but only used in command 39.	12	0 - 9
C	Header,$E_{MFK.E}$(KC),MAC**	74	0 - 9, A - F

* Legal pad characters are a hexadecimal value, X and W. The value X indicates that the pad character is unspecified but can be any hexadecimal character. The value W indicates that the sanity check will not be performed.
** Can be a volatile table location.

PIN Block

The IBM 3624 PIN block is produced in two steps:

1. Encrypt the eight rightmost bytes (16 hexadecimal characters) using the PIN Encryption Key (KPE).
2. Encrypt the eight leftmost bytes (16 hexadecimal characters) using the Communications Key.

The resulting cryptogram is written as $E_{KC}(E_{KPE}$(PIN Block)).

The PIN block is illustrated in Figure 4-3.

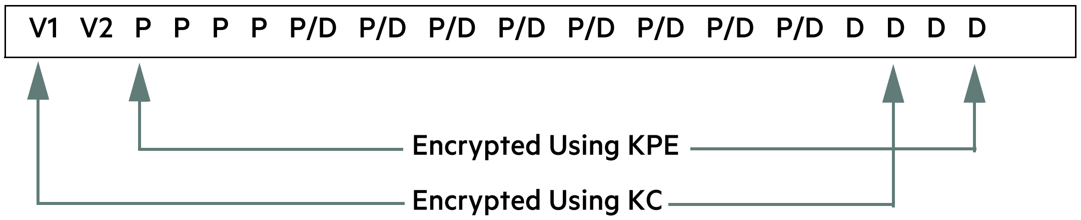


Figure 4-3 IBM 3624 PIN Block

`V1 V2`

Sequence number; two hexadecimal characters.

`P`

PIN digit. A four bit hexadecimal value in the range of 0 through 9.

`D`

Pad character. A four bit hexadecimal value.

`P/D`

A PIN digit or a pad character, depending on the PIN's length.

Example

The following example illustrates how the IBM 3624 PIN block is produced.

KPE = 1111 1111 1111 1111

KC = 2222 2222 2222 2222

PIN = 1234

Pad = B

Sequence # = FF

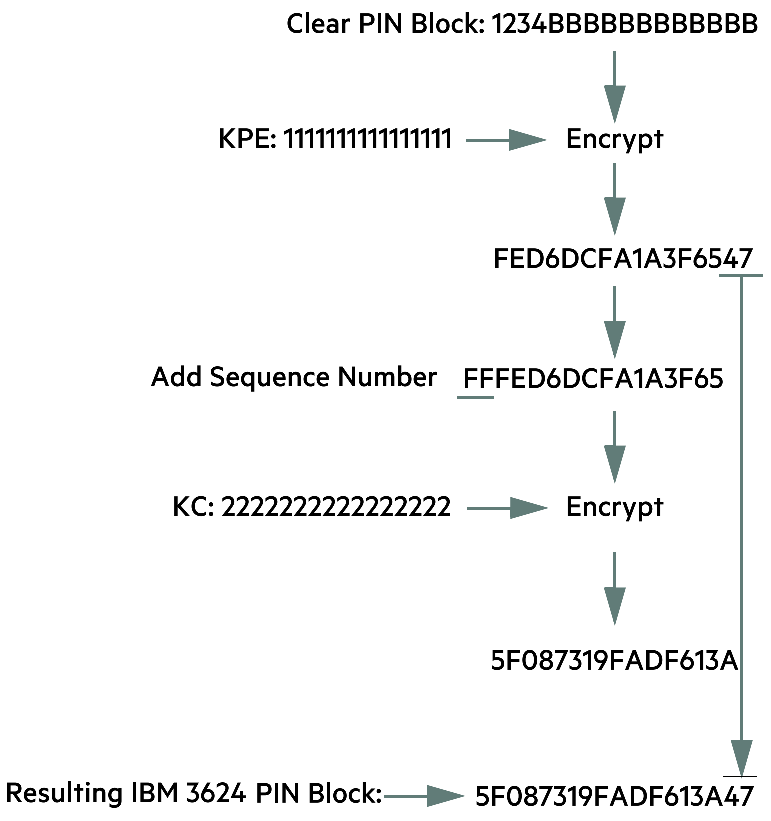


Figure 4-4 Producing the IBM 3624 PIN Block

PIN/Pad PIN Block

This PIN block is used by Diebold and some other ATM and PIN pad vendors.

PIN Block Data

The PIN/pad character PIN block requires two PIN Block Data fields; the last two fields of the command

.

Table 4-3 PIN/Pad - PIN Block Data

Field	Contents	Length (bytes)	Legal Characters
A	Pad character*	1	0 - 9, A - F, X, W
B	Twelve digit; required but only used in command 39.	12	0 - 9

* Legal pad characters are a hexadecimal value, X and W. The value X indicates that the pad character is unspecified but can be any hexadecimal character. The value W indicates that the sanity check will not be performed.

PIN Block

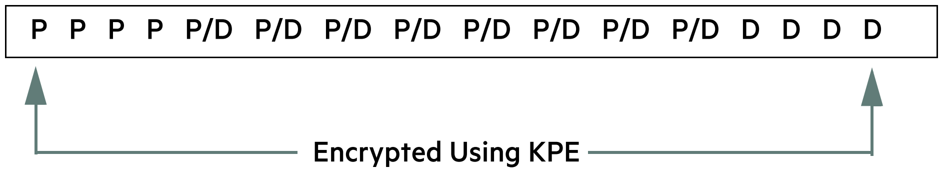


Figure 4-5 PIN/Pad Character PIN Block

P

PIN digit. A four bit hexadecimal value in the range of 0 through 9.

D

Pad character. A four bit hexadecimal value. All pad characters must be the same value.

P/D

A PIN digit or a pad character, depending on the PIN’s length.

Example

```
PIN = 1234

Pad = F

PIN Pad PIN block = 1234FFFFFFFFFFFF
```

Docutel PIN Block

The PIN digits are followed by a single character F and numeric pad characters.

PIN Block Data

The Docutel PIN block requires two PIN Block Data fields; the last two fields of the command.

Table 4-4 Docutel - PIN Block Data

Field	Contents	Length (bytes)	Legal Characters
A	Pad character*	1	0 - 9, X, W
B	Twelve digit; required but only used in command 39.	12	0 - 9

* Legal pad characters are a hexadecimal value, X and W. The value X indicates that the pad character is unspecified but can be any hexadecimal character. The value W indicates that the sanity check will not be performed.

PIN Block

P	P	P	P	P/F	P/F/D	P/F/D	P/F/D	P/F/D	P/F/D	P/F/D	P/F/D	F/D	D	D	D

Figure 4-6 Docutel PIN Block

P

PIN digit. A four bit hexadecimal value in the range of 0 through 9.

F

The four bit hexadecimal character F. This PIN block can contain only one F; it delimits the PIN.

D

Pad character. A four bit hexadecimal value in the range of 0 through 9.

Example

```
PIN = 1234

Pad = 10897645231

Docutel PIN block = 1234F10897645231
```

IBM Encrypting PIN Pad PIN Block

PIN Block Data

The IBM encrypting PIN pad PIN block requires one PIN Block Data data field; the last field of the command.

Table 4-5 IBM Encrypting PIN Pad - PIN Block Data

Field	Contents	Length (bytes)	Legal Characters
A	Twelve digit; required but only used in command 39.	12	0 - 9

PIN Block

C	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	S	S

Figure 4-7 IBM Encrypting PIN Pad

`C`

The length of the PIN. A four bit hexadecimal value 4 to 9, A, B, or C. A ten digit PIN is represented as A, an 11 digit PIN is represented as B, and a 12 digit PIN is represented as C.

`P`

PIN digit. A four bit hexadecimal value in the range of 0 through 9.

`F`

Pad character. A four bit value; hexadecimal F.

`P/F`

A PIN digit or a pad character, depending on the PIN's length.

`S`

The sequence number. Two 4 bit hexadecimal characters.

Example

```
PIN = 1234

Sequence Number = 07

Pin Block = 41234FFFFFFFFF07
```

Burroughs PIN Block

This PIN block is similar to the PIN/pad character PIN block, except that the PIN digits are ASCII hexadecimal characters instead of four bit hexadecimal values. A Burroughs PIN Block supports a maximum of eight PIN digits.

PIN Block Data

The Burroughs PIN block requires two PIN Block Data fields; the last two fields of the command.

Table 4-6 Burroughs - PIN Block Data

Field	Contents	Length (bytes)	Legal Characters
A	Pad character*	1	0 - 9, A - F, X, W
B	Twelve digit; required but only used in command 39.	12	0 - 9

* Legal pad characters are a hexadecimal value, X and W. The value X indicates that the pad character is unspecified but can be any hexadecimal character. The value W indicates that the sanity check will not be performed.

PIN Block

P	P	P	P	P/D	P/D	P/D	P/D	P/D	P/D	P/D	P/D	P/D	P/D	P/D	P/D

Figure 4-8 Burroughs PIN Block Type

`P`

PIN digit. Each PIN digit is converted to an ASCII hexadecimal value, 30 through 39 represents the values 0 through 9.

`D`

Pad character. A four bit hexadecimal value.

`P/D`

A PIN digit or a pad character, depending on the PIN's length.

Example

```
PIN = 1234

Pad = F

Burroughs PIN block = 31323334FFFFFFFF
```

ISO-2 PIN Block

The ISO-2 PIN block is the result of concatenating the PIN and filler data. The filler data consists of the letter F.

PIN Block

C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

Figure 4-9 ISO-2 PIN Block

`C`

The control field. A four bit value; hexadecimal 2.

`N`

PIN length. A four bit hexadecimal value in the range of 4 through 9, A, B, or C.

`P`

PIN digit. A four bit hexadecimal value in the range of 0 through 9.

`F`

The character F.

Example

```
PIN = 1234

ISO-2 PIN Block = 241234FFFFFFFFFF
```

ISO-3 PIN Block

The ISO-3 PIN block is the result of performing an exclusive-OR on two data blocks, the PIN block and the account number block.

PIN Block Data

The ISO-3 PIN block requires one PIN Block Data field; the last field of the command.

Table 4-7 ISO-3 - PIN Block Data

Field	Contents	Length (bytes)	Legal Characters
A	Twelve rightmost PAN digits (excluding check digits)	12	0 - 9

PIN Block

C	N	P	P	P	P	P/R	P/R	P/R	P/R	P/R	P/R	P/R	P/R	R	R

Figure 4-10 ISO-3 PIN Block

`C`

The control field. A four bit value; hexadecimal 3.

`N`

PIN length. A four bit hexadecimal value in the range of 4 through 9, A, B, or C.

`P`

PIN digit. A four bit hexadecimal value in the range of 0 through 9.

`R`

Random pad character. A four bit hexadecimal value in the range of A through F.

0	0	0	0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12

Figure 4-11 ISO-3 Account Number Block

`0`

Pad character. A four bit hexadecimal value 0.

`A1 to A12`

The 12 rightmost digits of the Primary Account Number (PAN), **excluding** the check digit. A1 is the most significant digit; A12 is the digit that immediately precedes the Primary Account Number's check digit.

Example

```
PIN = 1234

Primary Account Number = 5999997890123457

Random Pad             = DBFFAEBACE

PIN Block              = 341234DBFFAEBACE

Account Number Block   = 0000999789012345

exclusive-OR
the PIN and
Account Number Blocks

ISO-3 PIN Block        = 3412AD4C76AF998B
```

IBM 4731 PIN Block

The ATM Master Key encrypts the PIN, it is not provided in the command.

PIN Block Data

The IBM 4731 PIN block requires four PIN Block Data fields; the last four fields of the command. The following headers are supported in this PIN block format: 1c7NE000, 1c7NN000, 1c7DE000, and 1c7DN000.

Table 4-8 IBM 4731 - PIN Block Data

Field	Contents	Length (bytes)	Legal Characters
A	Pad Character*	1	0 - 9, A - F, X, W
B	PAN	12	0 - 9
C	Header,$E_{MFK.E}$(KC), MAC***	74	0 - 9, A - F
D	ICV	16	0 - 9, A - F

* Legal pad characters are a hexadecimal value, X and W. The value X indicates that the pad character is unspecified but can be any hexadecimal character. The value W indicates that the sanity check will not be performed.
*** Can be a volatile table location.

PIN Block

The IBM 4731 PIN pad creates a PIN block as illustrated in Figure 4-14 on page 4-16.

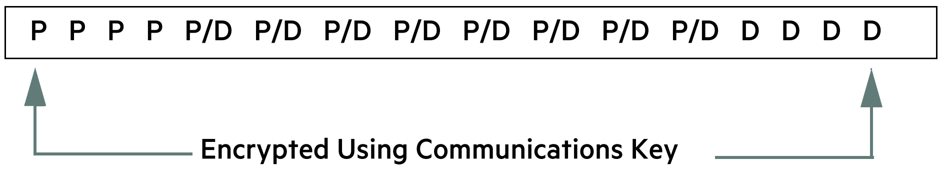

P P P P P/D P/D P/D P/D P/D P/D P/D P/D D D D D
Encrypted Using Communications Key

Figure 4-12 IBM 4731 PIN Block

`P`

PIN digit. A four bit hexadecimal value in the range of 0 through 9.

`D`

Pad character. A four bit hexadecimal value.

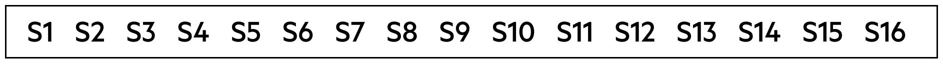


Figure 4-13 IBM 4731 ICV

`S1 to S16`

A 16 hexadecimal character value.

Example

Master Key = C8B3 047C F7A4 2A70
Communication Key = 68D5 9437 1067 794F
ICV = 0000 1560 0065 0039
PIN = 6731
Pad = F

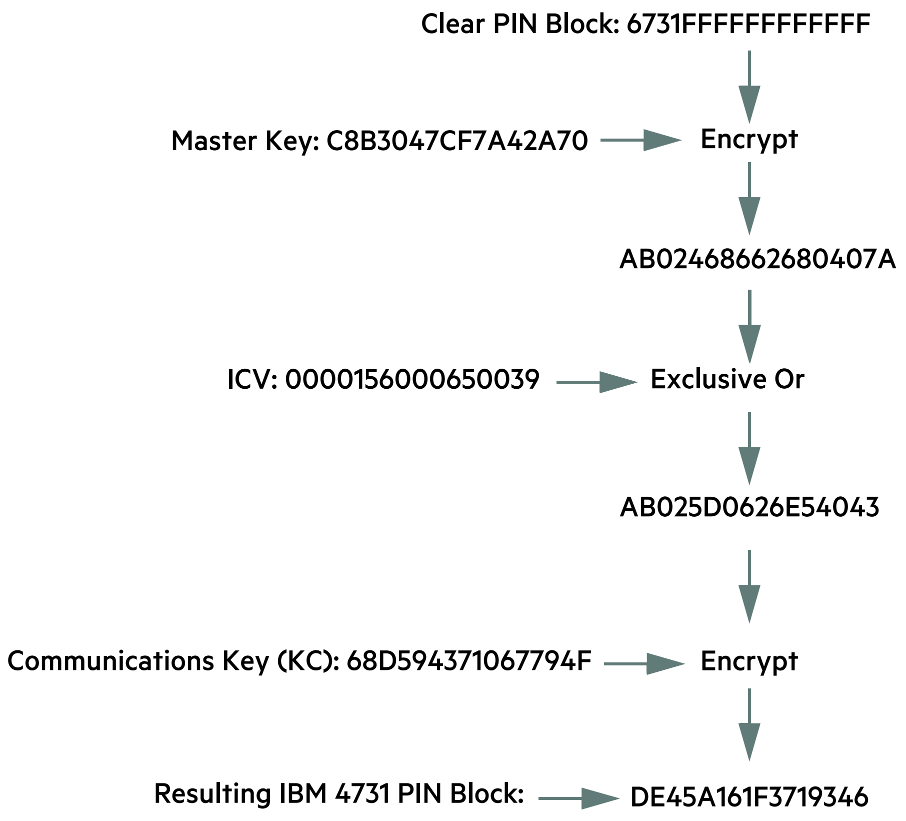

Clear PIN Block: 6731FFFFFFFFFFFF
Master Key: C8B3047CF7A42A70
Encrypt
AB02468662680407A
ICV: 0000156000650039
Exclusive Or
AB025D0626E54043
Communications Key (KC): 68D594371067794F
Encrypt
Resulting IBM 4731 PIN Block:
DE45A161F3719346

Figure 4-14 Producing the IBM 4731 PIN Block

VISA Derived Unique Key Per Transaction PIN Block

This subsection explains the VISA Derived Unique Key Per Transaction (DUKPT) PIN Block.

PIN Block Data

The following table delineates the VISA unique key per transaction PIN block data. These three fields are inserted as the last three fields of the command. It is referred to in the command syntax as PIN Block Data.

Table 4-9 VISA DUKPT - PIN Block Data

Field	Contents	Length (bytes)	Legal Characters
A	PAN digits for ANSI PIN block	12	0 - 9
B	Key serial number to generate current PIN Encryption Key (leading Fs are suppressed)	10 - 20	0 - 9, A - F
C	PIN encryption key derivation algorithm. When option A2 set to "B", the Network Security Processor generates a 1key-3DES (single-length) session key when this field contains either a "1" or "S". It generates a 2key-3DES (double-length) session key when this field contains the letter "D". When option A2 is set to "S", this field must contain the number "1" or the letter "S". When option A2 is set to "D", this field must contain the number "1" or the letter "D". The length of the Base Derivation Key must be greater than or equal to the length of the session key.	1	1, S, D

PIN Block

It is important to understand specific terms that are unique to the VISA DUKPT methodology.

Definition/Terms:

Derivation Key (DK)

A key used to encrypt the Initial Key Serial Number (IKSN) to obtain the Initial PIN Encryption Key (IPEK).

Key Serial Number (KSN)

A 20 character value that is transmitted from the EFT/POS terminal to the host. It allows the host to determine the key used to encrypt the PIN. The KSN consists of the Initial Key Serial Number (59 bits) + the Encryption Counter (21 bits).

Initial Key Serial Number (IKSN)

The leftmost 64 bits of the Key Serial Number.

Initial PIN Encryption Key (IPEK)

The result of encrypting the IKSN with the DK. (This value is not used to encrypt a PIN); see Current PIN Encryption Key.

Current Key

The result of encryption of the KSN with the IPEK.

Current PIN Encryption Key

Exclusive-OR the last byte of current key with FF.

Current MAC Key (VISA)

Exclusive-OR the last two bytes of current key with FFFF.

For information on 3DES-DUKPT see *ANSI x9-24-2004 Annex A*.

PIN Processing Commands

The remainder of this section contains the command and response syntax for the Network Security Processor PIN processing commands.

Quick Reference

Table 4-10 identifies each command by number, name, and purpose. It organizes the PIN processing commands by category, the commands themselves are presented in numerical order.

Table 4-10 PIN Processing Commands

Command	Name	Purpose
PIN encryption and decryption commands		
30	Encrypt PIN	Formats a clear-text PIN in the ANSI PIN Block, and encrypts it under a KPE.
90	Decrypt PIN	Decrypts an incoming PIN block and returns the clear-text PIN.
PIN translation commands		
31	Translate PIN	This command supports a variety of PIN block types. It outputs the PIN in an ANSI PIN block. It also translates the PIN from one key to encryption under another key. Support for Visa DUKPT PIN block requires option 62 to be enabled.
33	Translate PIN	Same as command 31, except the outgoing PIN Block is not limited to an ANSI PIN Block.
35	Translate PIN	Same as command 31, except the incoming and outgoing PIN block may be double encrypted.

Table 4-10 PIN Processing Commands (continued)

Command	Name	Purpose
39	Translate PIN and Generate MAC	Translates the PIN and generates a MAC. The outgoing PIN Block type is ANSI.
BA	PIN Translate ANSI to PLUS and Generate MAC	Translates the PIN and generates a MAC. The outgoing PIN Block type is PLUS.
BB	PIN Translate ANSI to PLUS and Verify MAC	Translates the PIN and verifies a MAC. The outgoing PIN Block type is PLUS.
BD	Translate PIN and Generate MAC	Translates the PIN and generates a MAC. The outgoing PIN Block type is ANSI and can be included in the MAC generation process.
335	PIN Translate	Supports multiple incoming and outgoing PIN block types.
346	PIN Translate - DUKPT to 3DES and Verify MAC	This command translates a DUKPT encrypted PIN to a 3DES encrypted PIN and verifies a MAC.
347	PIN Translate - DUKPT to 3DES and Generate MAC	This command translates a DUKPT encrypted PIN to a 3DES encrypted PIN and generates a MAC.
PIN Verify and PIN Change commands		
32	Verify PIN	Decrypts an incoming PIN and verifies it using a variety of PIN algorithms. Support for the Visa DUKPT requires option 63 to be enabled.
36	Verify Double-Encrypted PIN	Decrypts an incoming double-encrypted PIN and verifies it according to the specified PIN algorithm.
37	PIN Change	Decrypts an incoming PIN and verifies the old PIN using a variety of PIN algorithms, and calculates new PVN using the new PIN.
3A	Card and PIN Verify - IBM3624	This combination command verifies a CVV/CVC/CSC and then verifies a PIN.
3A	Card and PIN Verify - Visa	This combination command verifies a CVV/CVC/CSC and then verifies a PIN.
3A	Card and PIN Verify - Atalla Bilevel	This combination command verifies a CVV/CVC/CSC and then verifies a PIN.
3A	Card and PIN Verify - NCR	This combination command verifies a CVV/CVC/CSC and then verifies a PIN.
D0	Verify Clear PIN	Verifies a clear PIN using either the Identikey, IBM 3624, or Visa PIN algorithms.
32C	Verify ePIN	Verifies the entered ePIN using the ePIN Object.

Table 4-10 PIN Processing Commands (continued)

Command	Name	Purpose
Offset/PIN Generation commands		
3D	Generate PVN and Offset	Generates an Identikey PVN and IBM 3624 Offset for a PIN and account number.
11E	Generate Atalla 2x2 PVN	Generates an Atalla 2x2 PVN based on clear-text input.
30A	Calculate PIN Offset	Generates a new PIN Offset based on the PIN.
37B	Generate ePIN Offset	Generates an ePIN offset based on the ePIN and PAN.

Encrypt PIN - ANSI Format 0 (Command 30)

Command 30 encrypts a clear-text PIN. The ANSI PIN block is used to format the PIN prior to encryption.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

$<30\#Header,E_{MFK.E}(KPE),MAC\#PIN\#PAN\#>$

Response

$<40\#E_{KPE}(ANSI\ PIN\ Block)\#>[CRLF]$

Calling Parameters

`30`

Field 0, the command identifier.

$Header,E_{MFK.E}(KPE),MAC$

Field 1, the PIN Encryption Key (KPE). This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000 and 1PUEN000.

`PIN`

Field 2, the clear-text PIN. This field contains a numeric value. Option A0 defines the minimum PIN length. The maximum PIN length is 12 digits.

`PAN`

Field 3, the Primary Account Number (PAN) digits used to form the ANSI PIN block; the 12 rightmost digits, excluding the check digit. This field contains a 12 digit numeric value.

Table 4-11 Command 30: Encrypt PIN

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	30
1	Header,$E_{MFK.E}$(KPE),MAC*	74	printable ASCII
2	PIN	4 - 12	0 - 9
3	PAN	12	0 - 9

* Can be a volatile table location.

Responding Parameters

`40`

Field 0, the response identifier.

E_{KPE}(ANSI PIN Block)

Field 1, the encrypted ANSI PIN block. This field contains 16 hexadecimal characters.

Table 4-12 Response 40: Encrypt PIN

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	40
1	E_{KPE}(ANSI PIN Block)	16	0- 9, A - F

Usage Notes

- Generate the PIN Encryption Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Encrypting a PIN

- Clear-text PIN Encryption Key (KPE): 0123456789ABCDEF FEDCBA9876543210
  The PIN Encryption Key (KPE) in AKB format is:
  1PUNE000,27F0026930C71646A3ADED3356C36BB0CCC2D6874DE6AB43,83237B4A
  EA731054
- PIN: 12345678901
- Twelve rightmost digits of the Primary Account Number excluding the check digit:
  0002 3456 7890

The command looks like this:

```
<30#1PUNE000,27F0026930C71646A3ADED3356C36BB0CCC2D6874DE6AB43,
83237B4AEA731054#12345678901#000234567890#>
```

The Network Security Processor returns the following response:

```
<40#787F14641D51304D#>
```

Translate PIN (Command 31)

Command 31 translates an encrypted PIN block from encryption under an incoming PIN Encryption Key to an outgoing PIN Encryption Key. The translated PIN block will be in ANSI PIN Block format. The incoming PIN Encryption key is designated as KPE_I, and the outgoing PIN Encryption Key is designated as KPE_O. This command is enabled in the Network Security Processor's default security policy.

Command

```
<31#PIN Block Format#Header,EMFK.E(KPEI),MAC# Header,EMFK.E(KPEO),MAC#
EKPEI(PIN Block)#PIN Block Data#>
```

Response

```
<41#EKPEO(ANSI PIN Block)#Sanity Check Indicator#
[IBM 3624 Sequence Number#]>[CRLF]
```

Calling Parameters

`31`

Field 0, the command identifier.

`PIN Block Type`

Field 1, the incoming PIN block type. This field is 1 byte, it can contain the numbers 1, 2, 3, 4, 5 or 9. When option 46 is enabled, this field can only contain the value 1 (ANSI).

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN/pad character
3	Docutel
4	IBM encrypting PIN pad
5	Burroughs
9	IBM 4731
See Translate PIN – VISA DUKPT (Command 31)	VISA DUKPT

`Header,EMFK.E(KPEI),MAC`

Field 2, the Incoming PIN Encryption Key encrypted under the MFK. This field can be either a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported:

1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

`Header,`$E_{MFK.E}$`(KPE`$_O$`),MAC`

Field 3, the Outgoing PIN Encryption Key under the MFK. This field can be either a 74 byte value, or a volatile table location. The following headers are supported in this command: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, an error response is returned if the length of the (KPEo) is not equal to or greater than the length of the (KPEi).

E_{KPEI}`(PIN Block)`

Field 4, the incoming PIN block encrypted under the Incoming PIN Encryption Key. This field contains a 16 or 18 byte hexadecimal value.

`PIN Block Data`

Field 5, PIN block data. Its contents depend on the PIN block type. See PIN Block Types for information on PIN block data.

Table 4-13 Command 31: Translate PIN

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	31
1	PIN block type	1	1 - 5, 9
2	Header,$E_{MFK.E}$(KPE$_I$),MAC*	74	printable ASCII
3	Header,$E_{MFK.E}$(KPE$_O$),MAC*	74	printable ASCII
4	E_{KPEI}(PIN Block)	16, 18	0 - 9, A - F
5	PIN block data**		

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

`41`

Field 0, the response identifier.

E_{KPEO}`(ANSI PIN Block)`

Field 1, the PIN in ANSI PIN block format, encrypted under the Outgoing PIN Encryption Key. This field contains 16 hexadecimal characters. When a PIN sanity error is detected, the value in this field may not be correct. When a PIN sanity error is detected, and option 4B is enabled, this field will contain 16 zeros.

`Sanity Check Indicator`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

`[IBM 3624 Sequence Number#]`

Field 3, the IBM 3624 sequence number. This field is returned only if the PIN block type is IBM 3624. When present, this field contains 2 hexadecimal characters.

Table 4-14 Response 41: Translate PIN

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	41
1	E_{KPEO}(ANSI PIN Block)	16	0 - 9, A - F
2	Sanity check indicator	1	Y, N, L
3	IBM 3624 sequence number*	2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

- Generate the incoming and outgoing PIN Encryption Keys.
- Generate the ATM Communications Key if the incoming PIN block is IBM 3624.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate a PIN formatted in an ANSI PIN Block

- Clear-text PIN Encryption Key (KPE): 0123456789ABCDEF FEDCBA9876543210
  The PIN Encryption Key (KPE) in AKB format is:
  1PUNE000,27F0026930C71646A3ADED3356C36BB0CCC2D6874DE6AB43,83237B4AEA731054

- Clear-text Outgoing PIN Encryption Key:
  FEDCBA9876543210 0123456789ABCDEF
  The Outgoing PIN Encryption Key in AKB format:
  1PUNE000,8B672E58F4435F901BCC617C95C16388F06F9853B09A2301,DAD1F04BA C6D34BD

- Clear-text ANSI PIN block: 0B12 3454 4CC6 676F. The PIN is 12345678901
  Twelve rightmost digits of the Primary Account Number excluding the check digit: 0002 3456 7890. The encrypted incoming PIN block: 787F14641D51304D

The command looks like this:

```
<31#1#1PUNE000,27F0026930C71646A3ADED3356C36BB0CCC2D6874DE6AB43,
83237B4AEA731054#1PUNE000,8B672E58F4435F901BCC617C95C16388F06F985
3B09A2301,DAD1F04BAC6D34BD#787F14641D51304D#000234567890#>
```

The Network Security Processor returns the following response:

```
<41#50DD506F53C3828A#Y#>
```

Translate PIN – VISA DUKPT (Command 31)

Command 31 – VISA DUKPT translates an ANSI PIN block that is encrypted using a VISA DUKPT session key to an ANSI PIN block encrypted under an outgoing PIN Encryption Key.

This command, by default, will generate a 1key-3DES (single-length) session key. Use option A2 and field 7 - Algorithm, to control the length of the generated session key.

This command is enabled in the Network Security Processor's default security policy.

Command

<31#7#Header,$E_{MFK.E}$(Derivation Key),MAC# Header,$E_{MFK.E}$(KPE_O),MAC# E_{KPEn}(PIN Block)#PAN Digits#Key Serial Number#Algorithm#>

Response

<41#E_{KPEO}(ANSI PIN Block)#Sanity Check Indicator#>[CRLF]

Calling Parameters

`31`

Field 0, the command identifier.

`7`

Field 1, the type of the incoming PIN block; in this command, ANSI with DUKPT. This field contains a 1 byte decimal value of 7.

Header,$E_{MFK.E}$(Derivation Key),MAC

Field 2, the Derivation Key encrypted under the MFK This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1dDNE000 and 1dDNN000.

Header,$E_{MFK.E}$(KPE_O),MAC

Field 3, the Outgoing PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, an error response is returned if the length of the outgoing PIN encryption key is not equal to or greater than the length of the session key used to encrypt the incoming PIN.

E_{KPEn}(PIN Block)

Field 4, the incoming PIN, encrypted using the VISA DUKPT session key. This field contains 16 hexadecimal characters.

`PAN Digits`

Field 5, the 12 PAN digits used to form the ANSI PIN block. This field contains a 12 byte decimal value.

`Key Serial Number`

Field 6, the 10 to 20 digit Key Serial Number (KSN) from the PIN pad. This field contains a 10 to 20 byte hexadecimal value.

`Algorithm`

Field 7, this field is used to determine the length of the session key only when option A2 is set to “B”. With option A2 set to “B”, the Network Security Processor will generate a 1key-3DES (single-length) session key when this field contains either a “1” or “S”. It will generate a 2key-3DES (double-length) session key when this field contains the letter “D”.

The Network Security Processor will generate a 1key-3DES (single-length) session key when option A2 is set to “S”, therefore this field must contain the number “1” or the letter “S”.

The Network Security Processor will generate a 2key-3DES (double-length) session key when option A2 is set to “D”, therefore this field must contain the number “1” or the letter “D”.

Table 4-15 Command 31: Translate PIN – VISA DUKPT

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	2	31
1	PIN block type	1	7
2	Header,$E_{MFK.E}$(Derivation Key),MAC*	74	printable ASCII
3	Header,$E_{MFK.E}$(KPE_O),MAC*	74	printable ASCII
4	E_{KPEn}(PIN block)	16	0 - 9, A - F
5	PAN digits for ANSI PIN block	12	0 - 9
6	Key serial number to generate current PIN Encryption Key (leading Fs are suppressed)	10 - 20	0 - 9, A - F
7	Algorithm	1	1, S, D

* Can be a volatile table location.

Responding Parameters

`41`

Field 0, the response identifier.

`E`$_{KPEO}$`(ANSI PIN Block)`

Field 1, the PIN formatted in an ANSI PIN block and encrypted under the outgoing PIN Encryption Key. This field contains 16 hexadecimal characters. When a PIN sanity error is detected, the value in this field may not be correct. When a PIN sanity error is detected, and option 4B is enabled, this field will contain 16 zeros.

`Sanity Check Indicator`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	**Description**
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

Table 4-16 Response 41: Translate PIN – VISA DUKPT

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Response indicator	2	41
1	E$_{KPEO}$(ANSI PIN Block)	16	0 - 9, A - F
2	Sanity check indicator	1	Y, N, L

Usage Notes

- Generate the outgoing PIN Encryption Key and the derivation key.
- To use this command, option 62 must be enabled in the Network Security Processor's security policy.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate VISA DUKPT PIN block using a 1key-3DES (single-length) session key

- Option A2 is set to "S" or "B" and option 6C is enabled in the Network Security Processor security policy

- Clear-text Base Derivation Key: 0123456789ABCDEF FEDCBA9876543210
  The Base Derivation Key in AKB format:
  1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,342946FE884AA8B2

- Clear-text Outgoing PIN Encryption Key:
  FEDCBA9876543210 0123456789ABCDEF
  The Outgoing PIN Encryption Key in AKB format:
  1PUNE000,8B672E58F4435F901BCC617C95C16388F06F9853B09A2301,DAD1F04BAC6D34BD

- Clear-text ANSI PIN block: 0B12 3454 4CC6 676F. The PIN is 12345678901
  Twelve rightmost digits of the Primary Account Number excluding the check digit: 0002 3456 7890. The encrypted incoming PIN block: BC14A8602228A412

- PIN block data:
  - Twelve rightmost Primary Account Number digits: 0002 3456 7890
  - Key serial number: 9876 5432 10E0 0008
  - PIN encryption key derivation algorithm number: 1

The command looks like this:

```
<31#7#1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,
342946FE884AA8B2#1PUNE000,8B672E58F4435F901BCC617C95C16388F06F985
3B09A2301,DAD1F04BAC6D34BD#BC14A8602228A412#000234567890#98765432
10E00008#1#>
```

The Network Security Processor returns the following response:

```
<41#50DD506F53C3828A#Y#>
```

This example shows the syntax when the option A2 is set to "B" or "S" and field 7 is set to "S".

```
<31#7#1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,3
42946FE884AA8B2#1PUNE000,8B672E58F4435F901BCC617C95C16388F06F9853
B09A2301,DAD1F04BAC6D34BD#BC14A8602228A412#000234567890#987654321
0E00008#S#>
```

The Network Security Processor returns the following response:

```
<41#50DD506F53C3828A#Y#>
```

Translate VISA DUKPT PIN block using a 2key-3DES (double-length) session key

- Option A2 is set to "B" and option 6C is enabled in the Network Security Processor security policy

- Clear-text Base Derivation Key: 0123456789ABCDEF FEDCBA9876543210
  The Base Derivation Key in AKB format:
  1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,342946FE884AA8B2

- Clear-text Outgoing PIN Encryption Key:
  FEDCBA9876543210 0123456789ABCDEF
  The Outgoing PIN Encryption Key in AKB format:
  1PUNE000,8B672E58F4435F901BCC617C95C16388F06F9853B09A2301,DAD1F04BAC6D34BD

- Clear-text ANSI PIN block: 041274EDCBA9876F. The PIN is 1270.
  Twelve rightmost digits of the Primary Account Number excluding the check digit: 0412 3456 7890. The encrypted incoming PIN block: 7A21BD10F36DC41D

- PIN block data:
  - Twelve rightmost Primary Account Number digits: 041234567890
  - Key serial number: 9876 543210E00012
  - PIN encryption key derivation algorithm number: D

The command looks like this:

```
<31#7#1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,3
42946FE884AA8B2#1PUNE000,8B672E58F4435F901BCC617C95C16388F06F9853
B09A2301,DAD1F04BAC6D34BD#7A21BD10F36DC41D#041234567890#987654321
0E00012#D#>
```

The Network Security Processor returns the following response:

```
<41#7820FE6CFD54CE3A#Y#>
```

This example shows the syntax when the option A2 is set to “D” and field 7 is set to “1”.

```
<31#7#1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,3
42946FE884AA8B2#1PUNE000,8B672E58F4435F901BCC617C95C16388F06F9853
B09A2301,DAD1F04BAC6D34BD#7A21BD10F36DC41D#041234567890#987654321
0E00012#1#>
```

The Network Security Processor returns the following response:

```
<41#7820FE6CFD54CE3A#Y#>
```

Verify PIN – Identikey (Command 32)

Command 32 – Identikey decrypts an incoming encrypted PIN block and verifies it using the Atalla Identikey PIN verification method. This command is enabled in the Network Security Processor's default security policy.

Command

```
<32#1#PIN Block Format#E_KPE(PIN Block)#
Header,E_MFK.E(KPE),MAC#Bank ID#PVN#Comparison Indicator# Partial
PAN#PIN Block Data#>
```

Response

```
<42#Sanity Check Indicator/Verification Flag#
[IBM 3624 Sequence Number#]>[CRLF]
```

Calling Parameters

`32`

Field 0, the command identifier.

`1`

Field 1, the PIN verification method; Identikey.

`PIN Block Type`

Field 2, incoming PIN block type. This field is 1byte, and can contain the numbers 1, 2, 3, 4, 5, 7 or 9.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN/pad character / Docutel
4	IBM encrypting PIN pad
5	Burroughs
7	VISA DUKPT
9	IBM 4731

E_{KPE}(PIN Block)

Field 3, the encrypted PIN. This field contains a 16 or 18 byte hexadecimal value.

`Header,`$E_{MFK.E}$`(KPE),MAC`

Field 4, the Incoming PIN Encryption Key encrypted under the MFK. This field must be a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain
a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

When the PIN block type is VISA DUKPT (field 2 = 7), this field will contain the Derivation Key encrypted under the MFK. This field contains a 74 byte value, or a key table location. The following headers are supported: 1dDNE000 and 1dDNN000.

`Bank ID`

Field 5, the bank ID; clear-text or encrypted. The clear-text Bank ID can be a two, six, or eight digit number.

Bank ID	**Allowable Size (bytes)**
Backward index (algorithm number less than 65)	2
ISO number	6
Route and transfer number	8

The encrypted bank ID is a 74 byte AKB comprised of the following four data fields: ll, bbbbbbbb, p, and cc. The following headers are supported: 1VINE000, 1VINN000, 1VIVE000, and 1VIVN000.

ll - a two-digit number; the length of the Bank ID:

- 02 – The Bank ID in backward index format; the algorithm number must be less than 65.
- 06 – The Bank ID is a six digit ISO number.
- 08 – The Bank ID is an eight digit route-and-transfer number.

bbbbbbbb - The bank ID number (digits 0 - 9); must be the same length as ll.

p - The pad character F, right pads the combined length of the bank ID length (ll) and the bank ID value (bb - bbbbbbbb) resulting in 14 hexadecimal characters. Four pad characters are required when the bank ID is an eight digit value. Six pad characters are required when the bank ID is an six digit value. Ten pad characters are required when the bank ID is a two digit value.

cc - The two hexadecimal character comparison indicator. This field specifies the group (left, middle, or right) of four digits of the six-digit Identikey PIN Verification Number that will be used for the comparison.

- 4C – Compare the leftmost four digits.

- 4D – Compare the middle four digits.
- 52 – Compare the rightmost four digits.

`PVN`

Field 6, the PIN Verification Number. The PVN can be four, six, or eight digits in length, containing the numbers 0 to 7.

`Comparison Indicator`

Field 7, a comparison indicator that specifies which four digits (left, middle, or right) of the six-digit PVN will be compared. This field is 1 byte, and can contain the character L, M, or R. When the PVN is six or eight digits in length or field 5 contains an encrypted bank ID, the value of this field is not evaluated by the Network Security Processor.

`Partial PAN`

Field 8, the portion of the Primary Account Number to be used for verification. This field contains a 4 to 19 byte decimal value.

`PIN Block Data`

Field 9, PIN block data. The content and number of fields depend on the PIN block type. See PIN Block Types.

Table 4-17 Command 32: Verify PIN – Identikey

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	32
1	Identikey	1	1
2	PIN block type	1	1 - 5, 7, 9
3	E_{KPE}(PIN block)	16, 18	0 - 9, A - F
4	Header,$E_{MFK.E}$(KPE),MAC* or Header,$E_{MFK.E}$(DK),MAC*	74	printable ASCII
5	Bank ID	2, 6, 8 or 74	0 - 9 or printable ASCII
6	PIN verification number	4, 6, 8	0 - 7
7	Comparison indicator	1	L, M, R
8	Partial PAN	4 - 19	0 - 9
9	PIN block data**		

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

`42`

Field 0, the response identifier.

`Sanity Check Indicator/Verification Flag`

Field 1, the sanity check indicator and verification flag. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. If the PIN block passes the sanity check, the verification check is conducted. This field can contain one of the following values:

Value	Description
Y	PIN verification was successful.
N	PIN verification failed.
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

`[IBM 3624 Sequence Number#]`

Field 2, the IBM 3624 sequence number. This field is returned only if the PIN block type is IBM 3624. When present, this field contains 2 hexadecimal characters.

Table 4-18 Response 42: Verify PIN – Identikey

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	42
1	Sanity check indicator/verification flag	1	Y, N, S, L
2	IBM 3624 sequence number*	2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

- Generate the incoming PIN Encryption Key.
- Generate the ATM Communications Key when the incoming PIN block is IBM 3624.
- Generate the Derivation Key when the incoming PIN block is VISA DUKPT.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Identikey PIN Verification - clear-text Bank ID

- Verification method: Identikey (1)
- PIN block type: ANSI (1)
- Clear-text PIN block: 0B12 3454 4CC6 676F
  The Encrypted PIN block: 48E8 8008 12B0 C9EF
- Clear-text PIN Encryption Key: 0000 1111 2222 3333
  The PIN Encryption Key in AKB format:
  1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D60C5A5CA83A4D1258,0DBB2D4988A072F2
- Bank ID: 9876 5432
- PIN verification number: 7532 75
- Comparison indicator: L (not used)
- Partial PAN: 2345 6789 0
- PIN block data; in this case, the 12 rightmost digits of the Primary Account Number excluding the check digit: 0002 3456 7890

The command looks like this:

```
<32#1#1#48E8800812B0C9EF#1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D
60C5A5CA83A4D1258,0DBB2D4988A072F2#98765432#753275#L#234567890#00
0234567890#>
```

The Network Security Processor returns the following response:

```
<42#Y#>
```

Identikey PIN Verification - encrypted Bank ID

This example uses the same data values as shown in the previous example.

- Encrypted Bank ID:
  1VINE000,D8773B0C62848A84DF794716CDAF56293B4D0176F799DD04,9C17570A4E28B15F

The command looks like this:

```
<32#1#1#48E8800812B0C9EF#1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D
60C5A5CA83A4D1258,0DBB2D4988A072F2#1VINE000,D8773B0C62848A84DF794
716CDAF56293B4D0176F799DD04,9C17570A4E28B15F#
753275#L#234567890#000234567890#>
```

The Network Security Processor returns the following response:

```
<42#Y#>
```

Identikey PIN Verification - DUKPT encrypted PIN block

- Verification method: Identikey (1)
- PIN block type: VISA DUKPT (7)
- Clear-text PIN block: 0B12 3454 4CC6 676F
  The encrypted PIN Block: C623 0BAC D0AC 414B
- Clear-text Base Derivation Key: 1334 1334 1334 1334
  The Base Derivation Key in AKB format:
  1dDNE000,449DD7F6420AD2AD710102B444C180F37D0E78F5EFC186E7,181753679
  9D56B5E
- Identikey data:
  - Bank ID: 9876 5432
  - PIN verification number: 7532 75
  - Comparison indicator: L
  - Partial PAN: 2345 6789 0
- Twelve rightmost digits of the Primary Account Number: 0002 3456 7890
- Key serial number: 9876 5432 10E0 0001
- Algorithm: 1

The command looks like this:

```
<32#1#7#C6230BACD0AC414B#1dDNE000,449DD7F6420AD2AD710102B444C180F
37D0E78F5EFC186E7,1817536799D56B5E#98765432#753275#L#234567890#00
0234567890#9876543210E00001#1#>
```

The Network Security Processor returns the following response:

```
<42#Y#>
```

Verify PIN – IBM 3624 (Command 32)

Command 32 – IBM 3624 decrypts an incoming encrypted PIN block and verifies it using the IBM 3624 PIN Verification method. This command is enabled in the Network Security Processor's default security policy.

Command

```
<32#2#PIN Block Format#E_KPE(PIN Block)#Header,E_MFK.E(KPE),MAC#
Conversion Table#Offset#Validation Data#Pad#Check-Length Parameter#
Header,E_MFK.E(KPV),MAC#PIN Block Data#>
```

Response

```
<42#Sanity Check Indicator/Verification Flag#
[IBM 3624 Sequence Number#]>[CRLF]
```

Calling Parameters

`32`

Field 0, the command identifier.

`2`

Field 1, the PIN verification method; IBM 3624.

`PIN Block Type`

Field 2, the incoming PIN block type. This field is 1byte, it can contain the numbers 1, 2, 3, 4, 5, 7 or 9.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN/pad character / Docutel
4	IBM encrypting PIN pad
5	Burroughs
7	VISA DUKPT
9	IBM 4731

E_{KPE}`(PIN Block)`

Field 3, the encrypted PIN. This field contains a 16 or 18 byte hexadecimal value.

`Header,`$E_{MFK.E}$`(KPE),MAC`

Field 4, the Incoming PIN Encryption Key encrypted under the MFK. This field must be a 74 byte value, or a volatile table location. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

When the PIN block type is VISA DUKPT (field 2 = 7), this field will contain the Derivation Key encrypted under the MFK. This field contains a 74 byte value, or a key table location. The following headers are supported: 1dDNE000 and 1dDNN000.

When option 6C is enabled, this field can contain a 1key-3DES (single-length) key.

`Conversion Table`

Field 5, a table that maps hexadecimal digits (0 through 9, A through F) to decimal digits (0 through 9). This field contains a 16 byte decimal value containing the clear-text Conversion Table or a volatile table location. When option 48 is enabled, this field contains the conversion table in AKB format (the header must be 1nCNE000) or a volatile table location. When option 4E is enabled, all three forms of the conversion table (clear-text, decrypted, or value stored in volatile table location) must adhere to these rules:

- The conversion table must have at least eight unique digits.
- No single digit can occur more than four times.

`Offset`

Field 6, an offset value applied to the algorithm-generated PIN before comparing it with the customer-entered PIN. This field contains a 4 to 16 byte decimal value.

`Validation Data`

Field 7, validation data. This value is unique for each card holder and is typically the account number. This field contains a 4 to 16 byte hexadecimal value. When the PIN block type is ANSI (field 1 = 1) and option 4C is enabled, the value supplied in this field must be 12 digits in length and equal to the PIN Block Data value supplied in field 11.

`Pad`

Field 8, the pad character which right-pads the validation data. This field contains a one byte hexadecimal value. The pad character is only applied when the validation data is less than 16 digits.

`Check-Length`

Field 9, the check-length. This value, specified by the issuing financial institution, is typically the PIN length and determines the number of PIN digits to be compared. This field contains one hexadecimal character in the range of 4 through C.

`Header,`$E_{MFK.E}$`(KPV),MAC`

Field 10, the PIN Verification Key (KPV) encrypted under the MFK. This field contains a

74 byte value, or a volatile table location. When option 4A is enabled, this key can be a 1key-3DES (single-length) key. The following headers are supported: 1V3NE000, 1V3NN000, 1V3VE000, and 1V3VN000.

`PIN Block Data`

Field 11, PIN block data. The content and number of fields depend on the PIN block type. See PIN Block Types.

Table 4-19 Command 32: Verify PIN – IBM 3624

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	32
1	BM 3624	1	2
2	PIN block type	1	1 - 5, 7, 9
3	E_{KPE}(PIN Block)	16, 18	0 - 9, A - F
4	Header,$E_{MFK.E}$(KPE),MAC* or Header,$E_{MFK.E}$(DK),MAC*	74	printable ASCII
5	Conversion table*	16	0 - 9
6	Offset	4 - 16	0 - 9
7	Validation data	4 - 16	0 - 9, A - F
8	Pad	1	0 - 9, A - F
9	Check-length	1	4 - 9, A - C
10	Header,$E_{MFK.E}$(KPV),MAC*	74	printable ASCII
11	PIN block data**		

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

`42`

Field 0, the response identifier.

`Sanity Check Indicator/Verification Flag`

Field 1, the sanity check indicator and verification flag. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. If the PIN block passes the sanity check, the verification check is conducted. This field can contain one of the following values:

Value	Description
Y	PIN verification was successful.
N	PIN verification failed.
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

`[IBM 3624 Sequence Number#]`

Field 2, the IBM 3624 sequence number. This field is returned only if the PIN block type is IBM 3624. When present, this field contains 2 hexadecimal characters.

Table 4-20 Response 42: Verify PIN – IBM 3624

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	42
1	Sanity check indicator/verification flag	1	Y, N, S, L
2	IBM 3624 sequence number*	2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

- Generate the incoming PIN Encryption Key.
- Generate the ATM Communications Key when the incoming PIN block is IBM 3624.
- Generate the Derivation Key when the incoming PIN block is VISA DUKPT.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify an encrypted ANSI PIN block using the IBM 3624 verification method

- PIN block type: ANSI (1)
- Clear-text ANSI PIN block: 0936 1436 270E EEEE
  The encrypted ANSI PIN block: 0558 007D 2156 C394

- Clear-text PIN Encryption Key (KPE): 0000 1111 2222 3333
  The PIN Encryption Key (KPE) in AKB format:
  1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D60C5A5CA83A4D1258,0DBB2D4 988A072F2
- Conversion table: 8351 2964 7746 1538
- Offset: 6694 537
- Validation data: 3333 3333
- Pad character: D
- Check-length: 7
- Clear-text PIN Verification Key (KPV): 89B0 7B35 A1B3 F47E
  The PIN Verification Key in AKB format:
  1V3NE000,85F6B20112A3BCCCECBEB2BB9F050B788524A06CD21CFCC8,2442B5799 37AF5BB
- PIN block data; in this case, the 12 rightmost digits of the Primary Account Number: 0000 3331 1111

The command looks like this:

```
<32#2#1#0558007D2156C394#1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D60
C5A5CA83A4D1258,0DBB2D4988A072F2#8351296477461538#6694537#33333333#
D#7#1V3NE000,85F6B20112A3BCCCECBEB2BB9F050B788524A06CD21CFCC8,2442B
579937AF5BB#000033311111#>
```

The Network Security Processor returns the following response:

```
<42#Y#>
```

Verify PIN – VISA (Command 32)

Command 32 decrypts an incoming encrypted PIN block and verifies it using the VISA Verification Method. This command is enabled in the Network Security Processor's default security policy.

Command

```
<32#3#PIN Block Format#E_KPE(PIN Block)#Header,E_MFK.E(KPE),MAC#
Header,E_MFK.E(KPV),MAC#Reserved#PVV#PVKI#PAN#PIN Block Data#>
```

Response

```
<42#Sanity Check Indicator/Verification Flag#
[IBM 3624 Sequence Number#]>[CRLF]
```

Calling Parameters

`32`

Field 0, the command identifier.

`3`

Field 1, the verification method; VISA.

`PIN Block Format`

Field 2, the incoming PIN block type. This field is 1byte, it can contain the numbers 1, 2, 3, 4, 5, 7 or 9.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN pad character / Docutel
4	IBM encrypting PIN pad
5	Burroughs
7	VISA DUKPT
9	IBM 4731

E_{KPE}(PIN Block)

Field 3, the incoming PIN. This field contains a 16 or 18 byte hexadecimal value.

`Header,`$E_{MFK.E}$`(KPE),MAC`

Field 4, the Incoming PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

When the PIN block type is VISA DUKPT (field 2 = 7), this field will contain the Derivation Key encrypted under the MFK. This field contains a 74 byte value, or a key table location. The following headers are supported: 1dDNE000 and 1dDNN000.

When option 6C is enabled, this field can contain a 1key-3DES (single-length) key.

`Header,`$E_{MFK.E}$`(KPV),MAC`

Field 5, the PIN Verification Key pair (KPV) encrypted under the MFK. Both keys of the Visa key pair must be in the same Atalla Key Block. This field contains a 74 byte value, or a volatile table location. This key must be either a 2key- or 3key-3DES key. If option 8C is enabled, this key can be a 1key-3DES (single-length) key.The following headers are supported: 1VVNE000, 1VVNN000, 1VVVE000, and 1VVVN000.

`Reserved`

Field 6, this field must be empty.

`PVV`

Field 7, the PIN Verification Value (PVV) used to compare to the calculated value. This field contains a 4 byte decimal value.

`PVKI`

Field 8, the PIN Verification Key Indicator (PVKI) used in the algorithm that calculates the PIN verification value. This field contains a 1 byte decimal value in the range of 0 to 9.

`PAN`

Field 9, the rightmost eleven Primary Account Number digits, excluding the check digit. When the PIN block type is ANSI (field 1 = 1) and option 4C is enabled, this value must be present in the PIN Block Data value supplied in field 10.

`PIN Block Data`

Field 10, PIN block data. The content and number of fields depend on the PIN block type. See PIN Block Types.

Table 4-21 Command 32: Verify PIN – VISA

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	2	32
1	VISA	1	3

Table 4-21 Command 32: Verify PIN – VISA (continued)

Field	Contents	Length (bytes)	Legal Characters
2	PIN block type	1	1 - 5, 7, 9
3	E_{KPE}(PIN Block)	16, 18	0 - 9, A - F
4	Header,$E_{MFK.E}$(KPE),MAC* or Header,$E_{MFK.E}$(DK),MAC*	74	printable ASCII
5	Header,$E_{MFK.E}$(KPV),MAC*	74	printable ASCII
6	Reserved	0	none
7	PVV	4	0 - 9
8	PVKI	1	0 - 9
9	PAN	11	0 - 9
10	PIN block data**		

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

`42`

Field 0, the response identifier.

`Sanity Check Indicator/Verification Flag`

Field 1, the sanity check indicator and verification flag. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. If the PIN block passes the sanity check, the verification check is conducted. This field can contain one of the following values:

Value	Description
Y	PIN verification was successful.
N	PIN verification failed.
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

`[IBM 3624 Sequence Number#]`

Field 2, the IBM 3624 sequence number. This field is returned only if the PIN block type is

IBM 3624. When present, this field contains 2 hexadecimal characters.

Table 4-22 Response 42: Verify PIN – VISA

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	42
1	Sanity check indicator/verification flag	1	Y, N, S, L
2	IBM 3624 sequence number*	2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

- Generate the incoming PIN Encryption Key.
- Generate the ATM Communications Key if the incoming PIN block is IBM 3624.
- Generate the PIN Verification Key pair.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify an encrypted ANSI PIN block using the VISA verification method

- PIN block type: ANSI (1)
- Clear-text ANSI PIN block: 0638 23A0 BA98 76FE
  The ANSI PIN block encrypted under the PIN Encryption Key: 2BBF 4AF7 5D2F 8D62
- Clear-text PIN Encryption Key (KPE): 0000 1111 2222 3333
  The PIN Encryption Key in AKB format:
  1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D60C5A5CA83A4D1258,0DBB2D4
  988A072F2
- Clear-text key pair: 4CA2161637D0133E 5E151AEA45DA2A16
  The key pair in AKB format:
  1VVNE000,C56554CDE94948D004EB47FF82A81D8C24F89AFF4C47776C,9DE7167F
  56F172D7
- PIN verification value: 3691
- PIN Verification Key indicator: 3

- The 11 rightmost digits of the Primary Account Number excluding the check digit: 1234 5678 901
- PIN block data; in this case, the 12 rightmost digits of the Primary Account Number excluding the check digit: 5123 4567 8901

The command looks like this:

```
<32#3#1#2BBF4AF75D2F8D62#1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D
60C5A5CA83A4D1258,0DBB2D4988A072F2#1VVNE000,C56554CDE94948D004EB4
7FF82A81D8C24F89AFF4C47776C,9DE7167F56F172D7##3691#3#12345678901#
512345678901#>
```

The Network Security Processor returns the following response:

```
<42#Y#>
```

Verify PIN – Atalla DES BiLevel (Command 32)

Command 32 – Atalla DES Bilevel decrypts an incoming PIN and verifies it using the Atalla DES Bilevel method. This command is enabled in the Network Security Processor's default security policy.

Command

```
<32#4#PIN Block Format#E_KPE(PIN Block)#Header,E_MFK.E(KPE),MAC#
Bank ID#Partial PAN#Header,E_MFK.E(KPV),MAC#PVN-2#PVN-2 Type#
PVN-1 Flag#PVN-2 Start-Compare Flag#PIN Block Data#>
```

Response

```
<42#Sanity Check Indicator/Verification Flag#
[IBM Sequence Number#]>[CRLF]
```

Calling Parameters

`32`

Field 0, the command identifier.

`4`

Field 1, the PIN verification method; Atalla DES Bilevel.

`PIN Block Type`

Field 2, the incoming PIN block type. This field is 1byte, it can contain the numbers 1, 2, 3, 4, 5, 7 or 9.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN pad character / Docutel
4	IBM encrypting PIN pad
5	Burroughs
7	VISA DUKPT
9	IBM 4731

E_{KPE}`(PIN Block)`

Field 3, the encrypted PIN. This field contains a 16 or 18 byte hexadecimal value.

`Header,E`$_{MFK.E}$`(KPE),MAC`

Field 4, the Incoming PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

When the PIN block type is VISA DUKPT (field 2 = 7), this field will contain the Derivation Key encrypted under the MFK. This field contains a 74 byte value, or a key table location. The following headers are supported: 1dDNE000 and 1dDNN000.

When option 6C is enabled, this field can contain a 1key-3DES (single-length) key.

`Bank ID`

Field 5, the bank ID field for the Identikey card issuer. The ID is specified by the issuer and can be a two-, six-, or eight byte decimal value.

Bank ID	**Allowable Size (bytes)**
Backward index (algorithm number less than 65)	2
ISO number	6
Route and transfer number	8

`Partial PAN`

Field 6, validation data. This value is unique for each card holder, and in the case of this command, is the partial Primary Account Number (PAN). This field contains a 4 to 19 byte decimal value.

`Header,E`$_{MFK.E}$`(KPV),MAC`

Field 7, the PIN Verification Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1VBNE000, 1VBNN000, 1VBVE000, and 1VBVN000. When option 4A is enabled, this key can be a 1key -3DES (single-length) key.

`PVN-2`

Field 8, the PIN Verification Number to be verified. This field contains a 4 to 16 byte hexadecimal value.

`PVN-2 Type`

Field 9, the PVN-2 type. This field indicates whether the PVN-2 should be converted to a decimal value. This field is 1 byte, it contains the numbers 0 or 1. The following table identifies the Value for each type of PVN-2.

Value	Action
0	Convert PVN-2 to a decimal value
1	Do not convert PVN-2; leave it as a hexadecimal value

`PVN-1 Flag`

Field 10, a flag that indicates that 8 digits of the PVN-1 value should be used to compute the PVN-2. This field is 1 byte, it contains the number 8.

`PVN-2 Start-Compare Flag`

Field 11, a flag that specifies the starting position within the generated PVN-2 for the comparison. This field is 1 byte, it contains the number 1.

`PIN Block Data`

Field 12, PIN block data. The content and number of fields depend on the PIN block type. See PIN Block Types.

Table 4-23 Command 32: Verify PIN – Atalla DES Bilevel

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	32
1	Atalla DES bilevel	1	4
2	PIN block type	1	1 - 5, 7, 9
3	E_{KPE}(PIN block)	16, 18	0 - 9, A - F
4	Header,$E_{MFK.E}$(KPE),MAC*,Header,$E_{MFK.E}$(DK),MAC*	74	printable ASCII
5	Bank ID	2, 6, 8	0 - 9
6	Partial PAN	4 - 19	0 - 9
7	Header,$E_{MFK.E}$(KPV),MAC*	74	printable ASCII
8	PVN-2	4 - 16	0 - 9, A - F
9	PVN-2 type	1	0, 1
10	PVN-1 flag	1	8
11	PVN-2 start-compare flag	1	1
12	PIN block data**		

* Can be a volatile table location. ** See PIN Block Types for PIN block data information.

Responding Parameters

```
42
```

Field 0, the response identifier.

```
Sanity Check Indicator/Verification Flag
```

Field 1, the sanity check indicator and verification flag. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. If the PIN block passes the sanity check, the verification check is conducted. This field can contain one of the following values:

Value	Description
Y	PIN verification was successful.
N	PIN verification failed.
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

```
[IBM 3624 Sequence Number#]
```

Field 2, the IBM 3624 sequence number. This field is returned only if the PIN block type is IBM 3624. When present, this field contains 2 hexadecimal characters.

Table 4-24 Response 42: Verify PIN – Atalla DES Bilevel

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	42
1	Sanity check indicator/verification flag	1	Y, N, S, L
2	IBM 3624 sequence number*	2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

- Generate the incoming PIN Verification Key.
- Generate the ATM Communications Key if the incoming PIN block is IBM 3624.
- Generate the PIN Verification Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify an encrypted ANSI PIN block using Atalla DES Bilevel

- PIN block type: ANSI (1)
- Clear-text ANSI PIN block: 0512 345F FFFF FFFF
  The ANSI PIN block encrypted under the PIN Encryption Key (KPE): D492 0F0B 1BF0 39F2
- Clear-text PIN Encryption Key (KPE): 0000 1111 2222 3333
  The PIN Encryption Key in AKB format:
  1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D60C5A5CA83A4D1258,0DBB2D4988A072F2
- Bank ID: 591210
- Validation data: 5678901
- Clear-text PIN Verification Key (KPV): ABCD EF01 2345 6789
  The PIN Verification Key (KPV) in AKB format:
  1VBNE000,44DF0C0126B29C120B5FF8DC45E4CEF3983E3F17A91A3A95,E52824C52AA5A291
- PVN-2: 6341 6081 3974 3500
- PVN-2 type: 0
- PVN-1 flag: 8
- PVN-2 start-compare flag: 1
- PIN block data; in this case, the 12 rightmost digits of the Primary Account Number: 0000 0000 0000

The command looks like this:

```
<32#4#1#D4920F0B1BF039F2#1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D
60C5A5CA83A4D1258,0DBB2D4988A072F2#591210#5678901#1VBNE000,44DF0C
0126B29C120B5FF8DC45E4CEF3983E3F17A91A3A95,E52824C52AA5A291#63416
08139743500#0#8#1#000000000000#>
```

The Network Security Processor returns the following response:

```
<42#Y#>
```

Verify PIN – Diebold (Command 32)

Command 32 – Diebold decrypts an incoming encrypted PIN block and verifies it using the Diebold PIN Verification method. This command is enabled in the Network Security Processor's default security policy.

Command

```
<32#5#PIN Block Format#E_KPE(PIN Block)#Header,E_MFK.E(KPE),MAC#
Validation Data#Offset#Algorithm Number#
Diebold Number Table Location#PIN Block Data#>
```

Response

```
<42#Sanity Check Indicator/Verification Flag#
[IBM 3624 Sequence Number#]>[CRLF]
```

Calling Parameters

`32`

Field 0, the command identifier.

`5`

Field 1, the PIN verification method; Diebold.

`PIN Block Type`

Field 2, the incoming PIN block type. This field is 1byte, it can contain the numbers 1, 2, 3, 4, 5, 7 or 9.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN/pad character / Docutel
4	IBM encrypting PIN pad
5	Burroughs
7	VISA DUKPT
9	IBM 4731

E_{KPE}`(PIN Block)`

Field 3, the encrypted PIN. This field contains a 16 or 18 byte hexadecimal value.

`Header,E`$_{MFK.E}$`(KPE),MAC`

Field 4, the Incoming PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

When the PIN block type is VISA DUKPT (field 2 = 7), this field will contain the Derivation Key encrypted under the MFK. This field contains a 74 byte value, or a key table location. The following headers are supported: 1dDNE000 and 1dDNN000.

When option 6C is enabled, this field can contain a 1key-3DES (single-length) key.

`Validation Data`

Field 5, validation data. This value is unique for each card holder, and in the case of this command, is the Primary Account Number (PAN). This field contains a 4 to 19 byte decimal value.

`Offset`

Field 6, an offset value applied to the algorithm-generated PIN before comparing it with the customer-entered PIN. This field contains a 4 byte decimal value.

`Algorithm Number`

Field 7, the Diebold algorithm number. This field is 2 byte decimal value.

`Diebold Number Table Location`

Field 8, the index to the first volatile table location where the Diebold Number Table is stored. Thirty-two contiguous table locations hold the Diebold number table. This field contains a 1 to 4 byte decimal value.

`PIN Block Data`

Field 9, PIN block data. The content and number of fields depend on the PIN block type. See PIN Block Types.

Table 4-25 Command 32: Verify PIN – Diebold

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	32
1	Diebold	1	5
2	PIN block type	1	1 - 5, 7, 9
3	E_{KPE}(PIN block)	16, 18	0 - 9, A - F
4	Header,$E_{MFK.E}$(KPE),MAC* or Header,$E_{MFK.E}$(DK),MAC*	74	printable ASCII

Table 4-25 Command 32: Verify PIN – Diebold (continued)

Field	Contents	Length (bytes)	Legal Characters
5	Validation data	4 - 19	0 - 9
6	Offset	4	0 - 9
7	Algorithm number	2	0 - 9
8	Diebold Number Table location	1 - 4	0 - 9
9	PIN block data.**		

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

`42`

Field 0, the response identifier.

`Sanity Check Indicator/Verification Flag`

Field 1, the sanity check indicator and verification flag. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. If the PIN block passes the sanity check, the verification check is conducted. This field can contain one of the following values:

Value	Description
Y	PIN verification was successful.
N	PIN verification failed.
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.
INVALID NUMBER TABLE	The Diebold number table is invalid (it is empty or contains data other than the Diebold number table). This error usually indicates that the Diebold Number Table was not properly loaded into the volatile table.

`[IBM 3624 Sequence Number#]`

Field 2, the IBM 3624 sequence number. This field is returned only if the PIN block type is IBM 3624. When present, this field contains 2 hexadecimal characters.

Table 4-26 Response 42: Verify PIN – Diebold

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	42
1	Sanity check indicator/verification flag	1, 20	Y, N, S, L, INVALID NUMBER TABLE
2	IBM 3624 sequence number*	2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

- Use Load Diebold Number Table (Command 74) to load the Diebold number.
- Generate the ATM Communications Key if the incoming PIN block is IBM 3624.
- By default, this command processes the leftmost four PIN digits. Enable option 027 to process the rightmost four PIN digits.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

This example uses a specific Diebold Number Table your test results will be different.

Verify an encrypted ANSI PIN block using the Diebold verification method

- PIN block type: ANSI (1)
- Clear-text ANSI PIN block: 0464 56ED CB49 876F
  The ANSI PIN block encrypted under the PIN Encryption Key: AFC5 C290 4C92 2280
- Clear-text PIN Encryption Key (KPE): 0123 4567 89AB CDEF
  The PIN Encryption Key in AKB format:
  1PUNE000,27F0026930C71646D9F4D01EFF1231C7282455FD98438661,F607ECF664
  04617D
- Validation data: 1234 5678 90
- Offset: 0000
- Algorithm number: 82
- Diebold Number Table location: 250

- PIN block data; in this case, the 12 rightmost digits of the Primary Account Number: 0012 3456 7890

The command looks like this:

```
<32#5#1#AFC5C2904C922280#1PUNE000,27F0026930C71646D9F4D01EFF1231C
7282455FD98438661,F607ECF66404617D#1234567890#0000#82#250#0012345
67890#>
```

The Network Security Processor returns the following response:

```
<42#Y#>
```

Verify PIN – NCR (Command 32)

Command 32 – NCR decrypts an incoming encrypted PIN block and verifies it using the NCR method of verification. This command is enabled in the Network Security Processor's default security policy.

Command

```
<32#6#PIN Block Format#EKPE(PIN Block)#Header,EMFK.E(KPE),MAC#
Conversion Table#Offset#Validation Data#Pad#PLEN#
Header,EMFK.E(KPV),MAC#Padding Flag#Counting Flag#
Start-Count Position#Select-PLEN Position#PIN Block Data#>
```

Response

```
<42#Sanity Check Indicator/Verification Flag#
[IBM 3624 Sequence Number#]>[CRLF]
```

Calling Parameters

`32`

Field 0, the command identifier.

`6`

Field 1, the PIN verification method; NCR.

`PIN Block Type`

Field 2, the incoming PIN block type. This field is 1byte, it can contain the numbers 1, 2, 3, 4, 5, 7 or 9.

PIN Block Type	Value
ANSI	1
IBM 3624	2
PIN pad character / Docutel	3
IBM encrypting PIN pad	4
Burroughs	5
VISA DUKPT	7
IBM 4731	9

E_{KPE}(PIN Block)

Field 3, the encrypted PIN. This field contains a 16 or 18 byte hexadecimal value.

`Header,`$E_{MFK.E}$`(KPE),MAC`

Field 4, the Incoming PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

When the PIN block type is VISA DUKPT (field 2 = 7), this field will contain the Derivation Key encrypted under the MFK. This field contains a 74 byte value, or a key table location. The following headers are supported: 1dDNE000 and 1dDNN000.

When option 6C is enabled, this field can contain a 1key-3DES (single-length) key.

`Conversion Table`

Field 5, a table that maps hexadecimal digits (0 through 9, A through F) to decimal digits (0 through 9). This field contains a 16 byte decimal value containing the clear-text Conversion Table or a volatile table location. When option 48 is enabled, this field contains the conversion table in AKB format (the header must be 1nCNE000) or a volatile table location. When option 4E is enabled, all three forms of the conversion table (clear-text, decrypted, or value stored in volatile table location) must adhere to these rules:

- The conversion table must have at least eight unique digits.
- No single digit can occur more than four times.

`Offset`

Field 6, an offset value applied to the algorithm-generated PIN before comparing it with the customer-entered PIN. This field contains a 4 to 16 byte decimal value.

`Validation Data`

Field 7, validation data. This value is unique for each card holder and is typically the account number. This field contains a 4 to 16 byte hexadecimal value. When the PIN block type is ANSI (field 1 = 1) and option 4C is enabled, the value supplied in this field must be 12 digits in length and equal to the PIN Block Data value supplied in field 15.

`Pad`

Field 8, a pad character used to fill out the partial PAN. This field contains a one byte hexadecimal value.

`PLEN`

Field 9, the number of contiguous PIN digits selected for verification; the PIN length, or PLEN. This field is 1 byte, it contains a hexadecimal value in the range of 4 through C.

`Header,`$E_{MFK.E}$`(KPV),MAC`

Field 10, the PIN Verification Key (KPV) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1VNNE000, 1VNNN000, 1VNVE000, and 1VNVN000. When option 4A is enabled, this key can be a 1key -3DES (single-length) key.

`Padding Flag`

Field 11, a flag that indicates whether the validation data (Field 7) is to be padded on the left or on the right. This field is 1 byte, it contains the character L or R.

`Counting Flag`

Field 12, a flag that indicates whether the counting scheme for selecting the PIN digit for verification is left or right. This field is 1 byte, it contains the character L or R.

`Start-Count Position`

Field 13, the field that indicates the starting position for the counting scheme measured from either the left or right of the entered PIN depending on field 12. This field is 1 byte, it can contain a number in the range of 1 - 9.

`Select-PLEN Position`

Field 14, the field that indicates the beginning position (from the direction of the counting flag, starting with 0) for selecting PLEN characters from the output of the decimalization step. This field contains a one byte hexadecimal value. If the counting flag is “L”, the leftmost digit of the decimalized result is position zero. If the counting flag is “R,” the rightmost digit of the decimalized result is position zero.

`PIN Block Data`

Field 15, PIN block data. The content and number of fields depend on the PIN block type. See PIN Block Types.

Table 4-27 Command 32: Verify PIN – NCR

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	2	32
1	NCR	1	6
2	PIN block type	1	1 - 5, 7, 9
3	E_{KPE}(PIN block)	16, 18	0 - 9, A - F
4	Header,$E_{MFK.E}$(KPE),MAC* or Header,$E_{MFK.E}$(DK),MAC*	74	printable ASCII
5	Conversion table*	16, 74	0 - 9,
6	Offset	4 - 16	0 - 9
7	Validation data	4 - 16	0 - 9, A - F

Table 4-27 Command 32: Verify PIN – NCR (continued)

Field	Contents	Length (bytes)	Legal Characters
8	Pad	1	0 - 9, A - F
9	PLEN	1	4 - 9, A - C
10	Header,$E_{MFK.E}$(KPV),MAC*	74	printable ASCII
11	Padding flag	1	L, R
12	Counting flag	1	L, R
13	Start-count position	1	1 - 9
14	Select-PLEN position	1	0 - 9, A - C
15	PIN block data**		

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

`42`

Field 0, the response identifier.

`Sanity Check Indicator/Verification Flag`

Field 1, the sanity check indicator and verification flag. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. If the PIN block passes the sanity check, the verification check is conducted. This field can contain one of the following values:

Value	Description
Y	PIN verification was successful.
N	PIN verification failed.
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

`[IBM 3624 Sequence Number#]`

Field 2, the IBM 3624 sequence number. This field is returned only if the PIN block type is IBM 3624. When present, this field contains 2 hexadecimal characters.

Table 4-28 Response 42: Verify PIN – NCR

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	42
1	Sanity check indicator/verification flag	1	Y, N, S, L
2	IBM 3624 sequence number*	2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

- Generate the incoming PIN Encryption Key.
- Generate the PIN Verification Key.
- Generate the ATM Communications Key if the incoming PIN block is IBM 3624.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify an encrypted ANSI PIN block using the NCR verification method

- PIN block type: ANSI (1)
- Clear-text ANSI PIN block: 0413 25FF FFFF FFFF
  The ANSI PIN block encrypted under the PIN Encryption Key: 9A9C 37BF 6B38 8736.
- Clear-text PIN Encryption Key (KPE): 0000 1111 2222 3333
  The PIN Encryption Key in AKB format:
  1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D60C5A5CA83A4D1258,0DBB2D4
  988A072F2
- Conversion table: 0123 4567 8901 2345
- Offset: 0000
- Validation data: 2700 4552 4000 0121
- Pad character: F
- PLEN: 4

- Clear-text PIN Verification Key (KPV): 68BA 0794 F140 641C
  The PIN Verification Key in AKB format:
  1VNNE000,71013C7998BA9F5D6C4134C2B782851138021646F1C0065B,61FCCC1CA36B188E
- Padding flag: R
- Counting flag: L
- Start-count position: 1
- Select-PLEN position: 6
- PIN block data; in this case, the 12 rightmost digits of the Primary Account Number: 0455 2400 0012

The command looks like this:

```
<32#6#1#9A9C37BF6B388736#1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D
60C5A5CA83A4D1258,0DBB2D4988A072F2#0123456789012345#0000#27004552
40000121#F#4#1VNNE000,71013C7998BA9F5D6C4134C2B782851138021646F1C
0065B,61FCCC1CA36B188E#R#L#1#6#045524000012#>
```

The Network Security Processor returns the following response:

```
<42#Y#>
```

Verify PIN – Clear-PIN Comparison (Command 32)

Command 32 – Clear-PIN comparison decrypts an incoming encrypted PIN block and verifies it against the expected Clear-PIN value.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. You must purchase option 60 in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<32#7#PIN Block Format#E_KPE(PIN Block)#Header,E_MFK.E(KPE),MAC#
Clear-Text PIN#PIN Block Data#>
```

Response

```
<42#Sanity Check Indicator/Verification Flag#
[IBM 3624 Sequence Number#]>[CRLF]
```

Calling Parameters

`32`

Field 0, the command identifier.

`7`

Field 1, the PIN verification method; Clear-PIN comparison.

`PIN Block Type`

Field 2, the incoming PIN block type. This field is 1 byte, it can contain the numbers 1, 2, 3, 4, 5 or 9.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN pad character / Docutel
4	IBM encrypting PIN pad
5	Burroughs
9	IBM 4731

E_{KPE}(PIN Block)

Field 3, the encrypted PIN. This field contains a 16 or 18 byte hexadecimal value.

`Header,E`$_{MFK.E}$`(KPE),MAC`

Field 4, the PIN Encryption Key (KPE) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

`Clear-Text PIN`

Field 5, the clear-text PIN. This value will be compared to the PIN in the incoming PIN block. This field contains a 0 to 12 byte decimal value.

`PIN Block Data`

Field 6, PIN block data. Its contents depend on the PIN block type used. See PIN Block Types for information on PIN block data.

Table 4-29 Command 32: Verify PIN – Clear-PIN Comparison

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	32
1	PIN verification method	1	7
2	PIN block type	1	1 - 5, 9
3	E_{KPE}(PIN block)	16, 18	0 - 9, A - F
4	Header,$E_{MFK.E}$(KPE),MAC*	74	printable ASCII
5	Clear-text PIN	0 - 12	0 - 9
6	PIN block data**		

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

`42`

Field 0, the response identifier.

`Sanity Check Indicator/Verification Flag`

Field 1, the sanity check indicator and verification flag. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. If the PIN block passes the sanity check, the verification check is conducted. This field can contain one of the following values:

Value	Description
Y	PIN verification was successful.
N	PIN verification failed.
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

```
[IBM 3624 Sequence Number#]
```

Field 2, the IBM 3624 sequence number. This field is returned only if the PIN block type is IBM 3624. When present, this field contains 2 hexadecimal characters.

Table 4-30 Response 42: Verify PIN – Clear-PIN Comparison

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	42
1	Sanity check indicator/verification flag	1	Y, N, S, L
2	IBM 3624 sequence number*	2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

- This command supports PINs of length zero to twelve digits.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify an encrypted ANSI PIN block using the Clear-PIN comparison verification method

- Verification method: Clear-PIN comparison (7)
- PIN block type: ANSI (1)
- Clear-text ANSI PIN block: 0B12 3454 4CC6 676F
  The ANSI PIN block encrypted under the PIN Encryption Key: 48E8 8008 12B0 C9EF

- Clear-text PIN Encryption Key (KPE): 0000 1111 2222 3333
  The PIN Encryption Key in AKB format:
  1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D60C5A5CA83A4D1258,0DBB2D4988A072F2
- Clear-text PIN: 1234 5678 901
- PIN block data; in this case, the 12 rightmost digits of the Primary Account Number: 0002 3456 7890

The command looks like this:

```
<32#7#1#48E8800812B0C9EF#1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D
60C5A5CA83A4D1258,0DBB2D4988A072F2#12345678901#000234567890#>
```

The Network Security Processor returns the following response:

```
<42#Y#>
```

Verify PIN – PIN-Block Comparison (Command 32)

Command 32 – PIN-block comparison decrypts two incoming encrypted PIN blocks and compares the clear-text PIN blocks.

This command has a high security exposure. It is not enabled in the Network Security Processor's default security policy. You must purchase option 61 in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<32#8#E_KPE1(PIN Block 1)#Header,E_MFK.E(KPE_1),MAC#E_KPE2(PIN Block 2)#
Header,E_MFK.E(KPE_2),MAC#>
```

Response

```
<42#Verification Flag#>[CRLF]
```

Calling Parameters

`32`

Field 0, the command identifier.

`8`

Field 1, the PIN verification method; PIN-block comparison.

`E`$_{KPE1}$`(PIN Block1)`

Field 2, the first incoming PIN block encrypted under the first PIN Encryption Key (KPE1). This field contains 16 hexadecimal characters.

`Header,E`$_{MFK.E}$`(KPE`$_1$`),MAC`

Field 3, the first PIN Encryption Key encrypted under the MFK. This key is used to encrypt the first incoming PIN block. This field must be a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

`E`$_{KPE2}$`(PIN Block2)`

Field 4, the second incoming encrypted PIN block. This field contains 16 hexadecimal characters.

`Header,E`$_{MFK.E}$`(KPE`$_2$`),MAC`

Field 5, the second PIN Encryption Key (KPE_2) encrypted under the MFK. This key is used to encrypt the second incoming PIN block. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

Table 4-31 Command 32: Verify PIN – PIN-Block Comparison

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	32
1	Verification method	1	8
2	E_{KPE1}(PIN Block1)	16	0 - 9, A - F
3	Header,$E_{MFK.E}(KPE_1)$,MAC*	74	printable ASCII
4	E_{KPE2}(PIN Block2)	16	0 - 9, A - F
5	Header,$E_{MFK.E}(KPE_2)$,MAC*	74	printable ASCII

* Can be a volatile table location.

Responding Parameters

`42`

Field 0, the response identifier.

`Verification Flag`

Field 1, the verification flag. Starting with the leftmost position, the Network Security Processor scans the decrypted PIN blocks. The scan stops when a non-numeric character is encountered. The numeric digit of both PIN blocks are compared. Based on the comparison result, this field will contain one of the following values:

Value	Description
Y	Both PIN blocks are the same.
N	Both PIN blocks are not the same, or the first character in either PIN block is not a digit.
S	There are more than 12 digits in one or both of the PIN blocks.

Table 4-32 Response 42: Verify PIN – PIN-Block Comparison

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	42
1	Verification flag	1	Y, N, S

Usage Notes

- Generate the PIN Encryption Keys.
- This command does not check the entire PIN to be sure its length is legal. It compares two 16 character strings.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify different PIN Blocks which contain the same PIN

- Clear-text PIN Block1: 9999 9999 F103 3465
  The PIN Block encrypted under the KPE-1: 2D67 26EC DBCD EC3B
- Clear-text PIN Encryption Key-1: 634A 00F7 8F96 3784
  The PIN Encryption Key-1 in AKB format:
  1PUNE000,5CA851A9ADABFB947B0C52845BEC9BEC68424D38D5E78E71,94DBAE534A2A192C
- Clear-text PIN Block-2: 9999 9999 AAAA AAAA
  The PIN Block-2 encrypted under the KPE-2: 3F84 347A 3857 1B13
- Clear-text PIN Encryption Key-2: FEDC BA98 7654 3210
  The PIN Encryption Key-2 in AKB format:
  1PUNE000,8B672E58F4435F90A4E5F02158E165606ADADCDB58B3197C,6166E7F61B4920EF

The command looks like this:

```
<32#8#2D6726ECDBCDEC3B#1PUNE000,5CA851A9ADABFB947B0C52845BEC9BEC6
8424D38D5E78E71,94DBAE534A2A192C#3F84347A38571B13#1PUNE000,8B672E
58F4435F90A4E5F02158E165606ADADCDB58B3197C,6166E7F61B4920EF#>
```

The Network Security Processor returns the following response:

```
<42#Y#>
```

Verify PIN – Burroughs (Command 32)

Command 32 – Burroughs decrypts an incoming encrypted PIN block and verifies it using the Burroughs method of verification. This command is enabled in the Network Security Processor's default security policy.

Command

```
<32#F#PIN Block Type#$E_{KPE}$(PIN Block)#Header,$E_{MFK.E}$(KPE),MAC#
PAN#SECPD#SECTYPE#Offset#Header,$E_{MFK.E}$(Table1 Line0),MAC#
Header,$E_{MFK.E}$(Table1 Line1),MAC#Header,$E_{MFK.E}$(Table2 Line0),MAC#
Header,$E_{MFK.E}$(Table2 Line1),MAC#PIN Block Data#>
```

Response

```
<42#Sanity Check Indicator/Verification Flag#
[IBM 3624 Sequence Number#]>[CRLF]
```

Calling Parameters

`32`

Field 0, the command identifier.

`F`

Field 1, the PIN verification method; Burroughs.

`PIN Block Type`

Field 2, the incoming PIN block type. This field is 1 byte, it contains the numbers 1, 2 or 3.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN pad character / Docutel

E_{KPE}(PIN Block)

Field 3, the encrypted PIN. This field contains a 16 or 18 byte hexadecimal value.

Header,$E_{MFK.E}$(KPE),MAC

Field 4, the Incoming PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location.When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

`PAN`

Field 5, the Primary Account Number (PAN) to be used for verification. This field is 16 to 19 digits long.

`SECPD`

Field 6, Security Period. This field contains a 1 byte decimal value.

`SECTYPE`

Field 7, Security Method Character. This field contains a 1 byte decimal value.

`Offset`

Field 8, an offset value applied to the algorithm-generated PIN before comparing it with the customer-entered PIN. This field contains a 4 byte decimal value.

Header,$E_{MFK.E}$(Table1 Line0),MAC

Field 9, the first row of the first lookup table encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1nuNE000 and 1nuNN000.

Header,$E_{MFK.E}$(Table1 Line1),MAC

Field 10, the second row of the first lookup table encrypted under the MFK. This field contains a 16 byte hexadecimal value, or a volatile table location. The following headers are supported: 1nuNE000 and 1nuNN000.

Header,$E_{MFK.E}$(Table2 Line0),MAC

Field 11, the first row of the second lookup table encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1nuNE000 and 1nuNN000.

Header,$E_{MFK.E}$(Table2 Line1),MAC

Field 12, the second row of the second lookup table encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1nuNE000 and 1nuNN000.

`PIN Block Data`

Field 13, PIN block data. The content and number of fields depend on the PIN block type. See PIN Block Types.

Table 4-33 Command 32: Verify PIN – Burroughs

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	32

Table 4-33 Command 32: Verify PIN – Burroughs (continued)

Field	Contents	Length (bytes)	Legal Characters
1	Burroughs	1	F
2	PIN block type	1	1, 2, or 3
3	E_{KPE}(PIN block)	16, 18	0 - 9, A - F
4	Header,$E_{MFK.E}$(KPE),MAC*	74	printable ASCII
5	PAN	16-19	0 - 9,
6	SECPD	1	0 - 9
7	SECTYPE	1	0 - 9
8	Offset	4	0 - 9
9	Header,$E_{MFK.E}$(Table1 Line0),MAC*	16	0 - 9, A - F
10	Header,$E_{MFK.E}$(Table1 Line1),MAC*	16	0 - 9, A - F
11	Header,$E_{MFK.E}$(Table2 Line0),MAC*	16	0 - 9, A - F
12	Header,$E_{MFK.E}$(Table2 Line1),MAC*	16	0 - 9, A - F
13	PIN block data**		

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

```
42
```

Field 0, the response identifier.

```
Sanity Check Indicator/Verification Flag
```

Field 1, the sanity check indicator and verification flag. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. If the PIN block passes the sanity check, the verification check is conducted. This field can contain one of the following values:

Value	Description
Y	PIN verification was successful.
N	PIN verification failed.

Value	Description
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

```
[IBM 3624 Sequence Number#]
```

Field 2, the IBM 3624 sequence number. This field is returned only if the PIN block type is IBM 3624. When present, this field contains 2 hexadecimal characters.

Table 4-34 Response 42: Verify PIN – Burroughs

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	42
1	Sanity check indicator/verification flag	1	Y, N, S, L
2	IBM 3624 sequence number*	2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

- Generate the incoming PIN Encryption Key.
- Generate the table cryptograms.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify an encrypted ANSI PIN block using the Burroughs verification method

- PIN block type: ANSI (1)
- Clear-text ANSI PIN block: 0445 62FD 79FF FFFF
  The ANSI PIN block encrypted under the PIN Encryption Key (KPE): 47EB F9B3 877D B5C8
- Clear-text PIN Encryption Key (KPE): 0000 1111 2222 3333
  The PIN Encryption Key (KPE) in AKB format:
  1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D60C5A5CA83A4D1258,0DBB2D4988A072F2
- PAN: 0010 0006 0286 0000 00

- SECPD: 0
- SECTYPE: 0
- Offset: 0000
- Clear-text Table 1, Line 0: D7A9 E2FB 6834 05C1
  Table 1, Line 0 in AKB format:
  1nuNE000,255B63F3696C28A8451EF6715D8BEF88AFDA7D0C50CAC27F,E0D22CAB09684D98
- Clear-text Table 1, Line 1: 0000 0000 0000 0000
  Table 1, Line 1 in AKB format:
  1nuNE000,4ECAFCB2A1D801D5E5A03FC5BB6C3676CD8502434884E9C5,C7A554CA30C33BD3
- Clear-text Table 2, Line 0: C8B9 D1F2 A06E 5734
  Table 2, Line 0 in AKB format:
  1nuNE000,D87F79E05869E8D4D037C541928B5A58C6631F3968B1890D,5575800849EF1061
- Clear-text Table 1, Line 1: 0000 0000 0000 0000
  Table 2, Line 1 in AKB format:
  1nuNE000,4ECAFCB2A1D801D5E5A03FC5BB6C3676CD8502434884E9C5,C7A554CA30C33BD3
- PIN block data; in this case, the 12 rightmost digits of the Primary Account Number (PAN): 0602 8600 0000

The command looks like this:

```
<32#F#1#47EBF9B3877DB5C8#1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D6
0C5A5CA83A4D1258,0DBB2D4988A072F2#0010000602860000000#0#0#0000#1nuN
E000,255B63F3696C28A8451EF6715D8BEF88AFDA7D0C50CAC27F,E0D22CAB0968
4D98#1nuNE000,4ECAFCB2A1D801D5E5A03FC5BB6C3676CD8502434884E9C5,C7A
554CA30C33BD3#1nuNE000,D87F79E05869E8D4D037C541928B5A58C6631F3968B
1890D,5575800849EF1061#1nuNE000,4ECAFCB2A1D801D5E5A03FC5BB6C3676CD
8502434884E9C5,C7A554CA30C33BD3#060286000000#>
```

The Network Security Processor returns the following response:

```
<42#Y#>
```

Verify PIN – Atalla 2x2 (Command 32)

Command 32 – Atalla 2x2 verifies encrypted ANSI PIN blocks using the Atalla 2x2 algorithm. This command is enabled in the Network Security Processor's default security policy.

Command

```
<32#I#PIN Block Type#EKPE(PIN Block)#Header,EMFK.E(KPE),MAC#
Header,EMFK.E(KPV),MAC#Reserved#PVN Format#PVN#PAN Digits#>
```

Response

```
<42#Sanity Check Indicator/Verification Flag#>[CRLF]
```

Calling Parameters

`32`

Field 0, the command identifier.

`I`

Field 1, the verification method; Atalla 2x2.

`PIN Block Type`

Field 2, incoming PIN block is ANSI. This field contains the value 1.

E_{KPE}(PIN Block)

Field 3, the ANSI PIN block encrypted under the PIN Encryption Key (KPE). This field contains 16 hexadecimal characters.

Header,$E_{MFK.E}$(KPE),MAC

Field 4, the Incoming PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain
a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

Header,$E_{MFK.E}$(KPV),MAC

Field 5, the PIN Verification Key encrypted under the MFK. Both keys of the key pair must be in the same Atalla Key Block. This field contains a 74 byte value, or a volatile table location. This key must be either a 2key- or 3key-3DES key. The following headers are supported: 1VaNE000, 1VaNN000, 1VaVE000, and 1VaVN000.

`Reserved`

Field 6, this field is reserved for future use, any value in this field will be ignored by the

Network Security Processor. This field can be 0, 1, or 2 bytes long.

`PVN Format`

Field 7, specifies the format of the PVN. The choices are hexadecimal or decimal. This field should contain the letter H for hexadecimal format. For decimal format, this field should contain the letter D, followed by the 16 byte decimalization table. If you use the default decimalization table of 0123 4567 8901 2345, this field will contain only the letter D.

`PVN`

Field 8, the PIN Verification Number to be compared against the computed result. This field contains a 6 to 16 byte hexadecimal value.

`PAN Digits`

Field 9, the Primary Account Number digits used in the algorithm to generate the PVN. This field contains a 12 byte decimal value.

Table 4-35 Command 32: Verify PIN –Atalla 2x2

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	32
1	Atalla 2x2	1	I
2	PIN block type	1	1
3	E_{KPE}(PIN Block)	16	0 - 9, A - F
4	Header,$E_{MFK.E}$(KPE),MAC*	74	printable ASCII
5	Header,$E_{MFK.E}$(KPV),MAC*	74	printable ASCII
6	Reserved	0, 1, 2	0 - 31
7	PVN Format	1, 17	H, or D and 0 - 9
8	PVN	6-16	0 - 9, A - F
9	PAN Digits	12	0 - 9

* Can be a volatile table location.

Responding Parameters

`42`

Field 0, the response identifier.

`Sanity Check Indicator/Verification Flag`

Field 1, the sanity check indicator and verification flag. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. If the PIN block passes the sanity check, the verification check is conducted. This field can contain one of the following values:

Value	Description
Y	PIN verification was successful.
N	PIN verification failed.
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

Table 4-36 Response 42: Verify PIN – Atalla 2x2

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	42
1	Sanity check indicator/verification flag	1	Y, N, S, L

Usage Notes

- Generate the incoming PIN Encryption Key
- Generate the PIN Verification Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify an encrypted ANSI PIN block using the Atalla 2x2 method

- Verification method: Atalla 2x2 (I)
- PIN block type: ANSI (1)
- Clear-text ANSI PIN block: 0655 476B EDCB EDCB. The PIN is 555555
  The encrypted PIN Block: 661A B611 2C5E B5A0

- Clear-text PIN Encryption Key (KPE): 0000 1111 2222 3333
  The PIN Encryption Key in AKB format:
  1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D60C5A5CA83A4D1258,0DBB2D4988A072F2

- Clear-text PIN Verification Key pair: 5555 6666 7777 8888 9999 AAAA BBBB CCCC.
  The PIN Verification Key pair in AKB format:
  1VaNE000,A82F67C4AEF8EDB3A8851D83181497E82137C8836060C72F,1C6D2D968A0A6F7D

- PVN: 3436 593F 00F3 C754

- Twelve Primary Account Number digits: 1234 1234 1234

The command looks like this:

```
<32#I#1#661AB6112C5EB5A0#1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D
60C5A5CA83A4D1258,0DBB2D4988A072F2#1VaNE000,A82F67C4AEF8EDB3A8851
D83181497E82137C8836060C72F,1C6D2D968A0A6F7D##H#3436593F00F3C754#
123412341234#>
```

The Network Security Processor returns the following response:

```
<42#Y#>
```

Translate PIN – ANSI to PLUS and PLUS to ANSI (Command 33)

Command 33 – ANSI to PLUS and PLUS to ANSI. This command translates an encrypted PIN block from incoming encryption in an ANSI PIN block to outgoing encryption in the PLUS PIN block, or from incoming encryption in the PLUS PIN block to outgoing encryption in an ANSI PIN block. Both ANSI and PLUS use the same PIN block format. The PLUS PIN block requires the leftmost 12 account number digits, whereas the ANSI PIN block requires the rightmost 12 account number digits excluding the check digit. The incoming PIN Encryption key is designated as KPE_I and the outgoing PIN Encryption Key is designated as KPE_O.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

Command

```
<33#11#Header,E_MFK.E(KPE_I),MAC#Header,E_MFK.E(KPE_O),MAC#
E_KPEI(PIN Block)#Incoming PAN Digits#Outgoing PAN Digits#>
```

Response

```
<43#E_KPE_O(PIN Block)#Sanity Check Indicator#>[CRLF]
```

Calling Parameters

`33`

Field 0, the command identifier.

`11`

Field 1, the PIN translation method; in this command, both the input and output PIN blocks have the same format, only the account number digits may be different.

Header, $E_{MFK.E}(KPE_I)$, MAC

Field 2, the incoming PIN Encryption Key (KPE_I) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

Header, $E_{MFK.E}(KPE_O)$, MAC

Field 3, the outgoing PIN Encryption Key (KPE_O) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, the length of the KPEo must be equal to or greater than the length of the KPEi (field 2). The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000.

E_{KPE_I}(PIN Block)

Field 4, the PIN block encrypted under the incoming PIN Encryption Key. This field contains 16 hexadecimal characters.

`Incoming PAN Digits`

Field 5, the Primary Account Number (PAN) digits used in the incoming PIN block; that is, the 12 leftmost digits for PLUS or the 12 rightmost digits, excluding the check digit, for ANSI. This field contains a 12 byte decimal value. When either option 46 or 47 is enabled, the value of this field and field 6 must be identical.

`Outgoing PAN Digits`

Field 6, the Primary Account Number (PAN) digits used in the outgoing PIN block; the 12 leftmost digits for PLUS or the 12 rightmost digits, excluding the check digit, for ANSI. This field contains a 12 byte decimal value.

Table 4-37 Command 33: Translate PIN – ANSI to PLUS, PLUS to ANSI

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	33
1	PIN block Type	2	11
2	Header,$E_{MFK.E}$(KPE$_I$),MAC*	74	printable ASCII
3	Header,$E_{MFK.E}$(KPE$_O$),MAC*	74	printable ASCII
4	E_{KPE_I}(PIN block)	16	0 - 9, A - F
5	Incoming PAN digits	12	0 - 9
6	Outgoing PAN digits	12	0 - 9

* Can be a volatile table location.

Responding Parameters

`43`

Field 0, the response identifier.

E_{KPE_O}(PIN Block)

Field 1, the PIN in ANSI PIN block format, encrypted under the Outgoing PIN Encryption Key. This field contains 16 hexadecimal characters. When a PIN sanity error is detected, the value in this field may not be correct. When a PIN sanity error is detected, and option 4B is enabled, this field will contain 16 zeros.

`Sanity Check Indicator`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

Table 4-38 Response 43: Translate PIN – ANSI to PLUS, PLUS to ANSI

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	43
1	E_{KPE_O}(PIN block)	16	0 - 9, A - F
2	Sanity check indicator	1	Y, N, L

Usage Notes

- Generate the incoming and outgoing PIN Encryption Keys.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values..

Translate a PIN formatted in an ANSI PIN block to PLUS PIN block

- Clear-text incoming PIN Encryption Key (KPE_I): 07CE A74F 4607 5D8F
  The incoming PIN Encryption Key (KPE_I) in AKB format:
  1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,C47517937
  1FDBE43

- Clear-text outgoing PIN Encryption Key (KPE_O): D029 23D9 AD4F E90B
  The outgoing PIN Encryption Key (KPE_O) in AKB format:
  1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53AF657A4508,7ED6957
  0CDE54F30

- Clear-text PIN block: 0453 55F8 BEF7 EBBA
  The PIN block encrypted under the incoming PIN Encryption Key: 5196 681F 910C 408C.

- Incoming PAN digits: 1207 4108 1445
- Outgoing PAN digits: 2074 1081 4457

The command looks like this:

```
<33#11#1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,
C475179371FDBE43#1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53A
F657A4508,7ED69570CDE54F30#5196681F910C408C#120741081445#20741081
4457#>
```

The Network Security Processor returns the following response:

```
<43#7BB41A6FAA3BF848#Y#>
```

Translate PIN – ANSI to PIN/Pad (Command 33)

Command 33 – ANSI to PIN/pad. This command translates an encrypted ANSI PIN block to an encrypted PIN/pad character PIN block. The incoming PIN Encryption Key is designated as KPE_I, and the outgoing PIN Encryption Key is designated as KPE_O.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

This command will return an error if either option 46 or 47 is enabled.

Command

<33#13#Header,$E_{MFK.E}$(KPE$_I$),MAC#Header,$E_{MFK.E}$(KPE$_O$),MAC#
E_{KPE_I}(PIN Block)#Pad#PAN Digits#>

Response

<43#E_{KPE_O}(PIN Block)#Sanity Check Indicator#>[CRLF]

Calling Parameters

`33`

Field 0, the command identifier.

`13`

Field 1, the PIN translation method; in this command, ANSI to PIN pad.

Header,$E_{MFK.E}$(KPE$_I$),MAC

Field 2, the incoming PIN Encryption Key (KPE_I) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

Header,$E_{MFK.E}$(KPE$_O$),MAC

Field 3, the outgoing PIN Encryption Key (KPE_O) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, the length of the KPEo must be equal to or greater than the length of the KPEi (field 2). The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000.

E_{KPE_I}(PIN Block)

Field 4, the incoming PIN block encrypted under the incoming PIN Encryption Key. This field contains 16 hexadecimal characters.

`Pad`

Field 5, the pad character in the PIN pad block. This field is 1 byte, it can contain a hexadecimal value, X, or W. When this field contains the value X or W, the character F will be used as the pad character.

`PAN Digits`

Field 6, the Primary Account Number (PAN) digits used in the incoming ANSI PIN block. This field contains a 12 byte decimal value.

Table 4-39 Command 33: Translate PIN – ANSI to PIN/Pad

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	2	33
1	PIN translation method (ANSI to PIN pad)	2	13
2	Header,$E_{MFK.E}(KPE_I)$,MAC*	74	printable ASCII
3	Header,$E_{MFK.E}(KPE_O)$,MAC*	74	printable ASCII
4	E_{KPE_I}(PIN block)	16	0 - 9, A - F
5	Pad	1	0 - 9, A - F, X, W
6	PAN digits	12	0 - 9

* Can be a volatile table location.

Responding Parameters

`43`

Field 0, the response identifier.

E_{KPE_O}`(PIN Block)`

Field 1, the outgoing, encrypted PIN. This field contains 16 hexadecimal characters. When a PIN sanity error is detected, the value in this field may not be correct. When a PIN sanity error is detected, and option 4B is enabled, this field will contain 16 zeros.

`Sanity Check Indicator`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

Table 4-40 Response 43: Translate PIN – ANSI to PIN/Pad

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	43
1	E_{KPE_O}(PIN block)	16	0 - 9, A - F
2	Sanity check indicator	1	Y, N, L

Usage Notes

- Generate the incoming and outgoing PIN Encryption Keys.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate a PIN formatted in an ANSI PIN block to PIN/pad character PIN block

- Clear-text incoming PIN Encryption Key (KPE_I): 07CE A74F 4607 5D8F
  The incoming PIN Encryption Key (KPE_I) in AKB format:
  1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,C47517937
  1FDBE43

- Clear-text outgoing PIN Encryption Key (KPE_O): D029 23D9 AD4F E90B
  The outgoing PIN Encryption Key (KPE_O) in AKB format:
  1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53AF657A4508,7ED6957
  0CDE54F30

- Clear-text incoming PIN block: 045355F8BEF7EBBA
  The incoming PIN block encrypted under the incoming PIN Encryption Key (KPE_I):
  5196 681F 910C 408C

- Outgoing Pad character: D

- Incoming PAN: 1207 4108 1445

  The command looks like this:

```
<33#13#1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,
C475179371FDBE43#1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53A
F657A4508,7ED69570CDE54F30#5196681F910C408C#D#120741081445#>
```

The Network Security Processor returns the following response:

```
<43#F9081E2639080784#Y#>
```

Translate PIN – ANSI to IBM 4731 (Command 33)

Command 33 – ANSI to IBM 4731. This command translates an encrypted ANSI PIN block to an encrypted IBM 4731 PIN block. The incoming PIN Encryption Key is designated as KPE_I, and the outgoing PIN Encryption Key is designated as KPE_O.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

This command will return an error if either option 46 or 47 is enabled.

Command

```
<33#19#Header,EMFK.E(KPEI),MAC#Header,EMFK.E(KPEO),MAC#
EKPEI(PIN Block)#Incoming PAN#Outgoing Pad#Outgoing ICV#
Header,EMFK.E(KC),MAC#>
```

Response

```
<43#EKPEO(PIN Block)#Sanity Check Indicator#>[CRLF]
```

Calling Parameters

`33`

Field 0, the command identifier.

`19`

Field 1, the PIN translation method; in this command, ANSI to IBM 4731.

Header,$E_{MFK.E}(KPE_I)$,MAC

Field 2, the incoming PIN Encryption Key (KPE_I) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

Header,$E_{MFK.E}(KPE_O)$,MAC

Field 3, the outgoing PIN Encryption Key (KPE_O) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, the length of the KPEo must be equal to or greater than the length of the KPEi (field 2). The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000.

E_{KPE_I}(PIN Block)

Field 4, the incoming PIN block encrypted under the incoming PIN Encryption Key. This

field contains 16 hexadecimal characters.

`Incoming PAN`

Field 5, the Primary Account Number (PAN) used in the incoming PIN block; the 12 rightmost digits, excluding the check digit. This field contains a 12 byte decimal value.

`Outgoing Pad`

Field 6, the pad character for the outgoing PIN block. This field is 1 byte, it can contain a hexadecimal value, X, or W. An X or W indicate that the pad character used in the incoming PIN block will also be used as the outgoing pad character.

`Outgoing ICV`

Field 7, the sequence number for the outgoing PIN block. This field contains 16 hexadecimal characters.

Header,$E_{MFK.E}$(KC),MAC

Field 8, the Communications Key encrypted under the MFK. This key is used in the outer or second encryption of the IBM 4731 PIN block for the outgoing PIN block. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. This field contains a 74 byte value or a volatile table location. The following headers are supported: 1c7NE000, 1c7NN000, 1c7EE000, and 1c7EN000.

Table 4-41 Command 33: Translate PIN – ANSI to IBM 4731

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	33
1	PIN translation method (ANSI to IBM 4731)	2	19
2	Header,$E_{MFK.E}$(KPE$_I$),MAC*	74	printable ASCII
3	Header,$E_{MFK.E}$(KPE$_O$),MAC*	74	printable ASCII
4	E_{KPE_I}(PIN block)	16	0 - 9, A - F
5	Incoming PAN	12	0 - 9
6	Outgoing Pad	1	0 - 9, A - F, X, W
7	Outgoing ICV	16	0 - 9, A - F
8	Header,$E_{MFK.E}$(KC),MAC*	74	printable ASCII

* Can be a volatile table location.

Responding Parameters

`43`

Field 0, the response identifier.

E_{KPE_O}(PIN Block)

Field 1, the outgoing, encrypted PIN. This field contains 16 hexadecimal characters. When a PIN sanity error is detected, the value in this field may not be correct. When a PIN sanity error is detected, and option 4B is enabled, this field will contain 16 zeros.

`Sanity Check Indicator`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

Table 4-42 Response 43: Translate PIN – ANSI to IBM 4731

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	43
1	E_{KPE_O}(PIN block)	16	0 - 9, A - F
2	Sanity check indicator	1	Y, N, L

Usage Notes

- Generate incoming and outgoing PIN Encryption Keys and the Communications Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate a PIN formatted in an ANSI PIN block to an IBM 4731 PIN block

- Clear-text incoming PIN Encryption Key (KPE_I): 07CE A74F 4607 5D8F

  The incoming PIN Encryption Key (KPE_I) in AKB format:

  1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,C47517937
1FDBE43

- Clear-text outgoing PIN Encryption Key (KPE_O): D029 23D9 AD4F E90B
  The outgoing PIN Encryption Key (KPE_O) in AKB format:
  1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53AF657A4508,7ED6957
  0CDE54F30
- Incoming PAN: 1207 4108 1445
- Outgoing Pad character: D
- Sequence Number (ICV) 1234 1234 1234 1234
- Clear-text Communications Key: B302 AD91 F504 EA22
  The Communications Key in AKB format:
  1c7NE000,0B5822317FC002F870FB54EEEA2DAB004A454333FC6B60E4,45F760CD
  B48BB404

The command looks like this:

```
<33#19#1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,
C475179371FDBE43#1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53A
F657A4508,7ED69570CDE54F30#5196681F910C408C#120741081445#D#123412
3412341234#1c7NE000,0B5822317FC002F870FB54EEEA2DAB004A454333FC6B6
0E4,45F760CDB48BB404#>
```

The Network Security Processor returns the following response:

```
<43#27682B863CD388E8#Y#>
```

Translate PIN – IBM 3624 to IBM 3624 (Command 33)

Command 33 – IBM 3624 to IBM 3624. This command translates an encrypted IBM 3624 PIN block. The incoming PIN Encryption Key is designated as KPE_I, and the outgoing PIN Encryption Key is designated as KPE_O. The incoming Communications Key is designated as KC_I, and the outgoing Communications Key is designated as KC_O.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

This command will return an error if either option 46 or 47 is enabled.

Command

```
<33#22#Header,EMFK.E(KPEI),MAC#Header,EMFK.E(KPEO),MAC#
EKPEI(PIN Block)#Incoming Pad#Header,EMFK.E(KCI),MAC#
Outgoing Pad#Header,EMFK.E(KCO),MAC#>
```

Response

```
<43#EKPEO(PIN Block)#Sanity Check Indicator#
IBM 3624 Sequence Number#>[CRLF]
```

Calling Parameters

`33`

Field 0, the command identifier.

`22`

Field 1, the PIN translation method; in this command, IBM 3624 to IBM 3624.

Header, $E_{MFK.E}(KPE_I)$, MAC

Field 2, the incoming PIN Encryption Key (KPE_I) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

Header, $E_{MFK.E}(KPE_O)$, MAC

Field 3, the outgoing PIN Encryption Key (KPE_O) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, the length of the KPEo must be equal to or greater than the length of the KPEi (field 2). The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000.

E_{KPE_I}(PIN Block)

Field 4, the incoming PIN block encrypted under the incoming PIN Encryption Key. This field contains an 18 byte hexadecimal value.

`Incoming Pad`

Field 5, the pad character for the incoming PIN block. The field is one byte, it can contain a hexadecimal value, X, or W. The value X indicates any hexadecimal pad character is allowed. The value W indicates the sanity check will not be performed.

Header,$E_{MFK.E}(KC_I)$,MAC

Field 6, the incoming Communications Key encrypted under the MFK. This key is used in the outer, or second, encryption of the IBM 3624 PIN block for the incoming PIN. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1DDNE000, 1DDDE000, and 1DDEE000.

`Outgoing Pad`

Field 7, the pad character for the outgoing PIN block. This field is 1 byte, it can contain a hexadecimal value, X, or W. The value X or W indicates that the pad character for the incoming PIN block will also be used as the outgoing pad character.

Header,$E_{MFK.E}(KC_O)$,MAC

Field 8, the outgoing Communications Key encrypted under the MFK. This key is used in the outer, or second, encryption of the IBM 3624 PIN block for the outgoing PIN block. This field contains a 74 byte value or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1DDNE000, 1DDDE000, and 1DDEE000.

Table 4-43 Command 33: Translate PIN – IBM 3624 to IBM 3624

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	2	33
1	PIN block translation method (IBM 3624 to IBM 3624)	2	22
2	Header,$E_{MFK.E}(KPE_I)$,MAC	74*	printable ASCII
3	Header,$E_{MFK.E}(KPE_O)$,MAC	74*	printable ASCII
4	E_{KPE_I}(PIN block)	18	0 - 9, A - F
5	Incoming pad	1	0 - 9, A - F, X, W
6	Header,$E_{MFK.E}(KC_I)$,MAC	74*	printable ASCII

Table 4-43 Command 33: Translate PIN – IBM 3624 to IBM 3624 (continued)

Field	Contents	Length (bytes)	Legal Characters
7	Outgoing pad	1	0 - 9, A - F, X, W
8	Header,$E_{MFK.E}$(KC_O),MAC	74*	printable ASCII

* Can be a volatile table location.

Responding Parameters

`43`

Field 0, the response indicator.

`EKPEO(PIN Block)`

Field 1, the outgoing encrypted PIN. This field is an 18 byte hexadecimal value. When a PIN sanity error is detected, the value in this field may not be correct.

`Sanity Check Indicator`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

`IBM 3624 Sequence Number`

Field 3, the IBM 3624 sequence number. This field contains 2 hexadecimal characters.

Table 4-44 Response 43: Translate PIN – IBM 3624 to IBM 3624

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	2	43
1	E_{KPEO}(PIN block)	18	0 - 9, A - F
2	Sanity check indicator	1	Y, N, L
3	IBM 3624 sequence number	2	0 - 9, A - F

Usage Notes

- Generate the incoming and the outgoing PIN Encryption Keys and Communications Keys.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate a PIN formatted in an IBM 3624 PIN block to IBM 3624 PIN block

- Clear-text incoming PIN Encryption Key (KPE_I): 07CE A74F 4607 5D8F
  The incoming PIN Encryption Key (KPE_I) in AKB format:
  1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,C47517937
  1FDBE43

- Clear-text outgoing PIN Encryption Key (KPE_O): D029 23D9 AD4F E90B
  The outgoing PIN Encryption Key (KPE_O) in AKB format:
  1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53AF657A4508,7ED6957
  0CDE54F30

- Encrypted incoming PIN block: 9864 AB86 5904 8084 B8

- Incoming pad character: B.

- Clear-text incoming Communications Key: A15D BAFD F119 F701.
  The incoming Communications Key in AKB format:
  1DDNE000,740B352ADA6934929EECA387D23104F5582D59EF93C3AB88,571989DB
  B9DBC083

- Outgoing pad character: D

- Clear-text outgoing Communications Key: F72B 85D0 302D 448A
  The outgoing Communications Key in AKB format:
  1DDNE000,21FC89139C29187EA8289644DB94778329EB33A75B5E7957,7BA5D13D82
  92D947

The command looks like this:

```
<33#22#1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,C4
75179371FDBE43#1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53AF657
A4508,7ED69570CDE54F30#9864AB8659048084B8#B#1DDNE000,740B352ADA6934
929EECA387D23104F5582D59EF93C3AB88,571989DBB9DBC083#D#1DDNE000,21FC
89139C29187EA8289644DB94778329EB33A75B5E7957,7BA5D13D8292D947#>
```

The Network Security Processor returns the following response:

```
<43#843322E77167AE5384#Y#99#>
```

Translate PIN – IBM 3624 to PIN/Pad (Command 33)

Command 33 – IBM 3624 to PIN/pad. This command translates an encrypted IBM 3624 PIN block to an encrypted PIN/pad character PIN block. The incoming PIN Encryption Key is designated as KPE_I, and the outgoing PIN Encryption Key is designated as KPE_O.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

This command will return an error if either option 46 or 47 is enabled.

Command

```
<33#23#Header,E_MFK.E(KPE_I),MAC#Header,E_MFK.E(KPE_O),MAC#
E_KPE_I(PIN Block)#Incoming Pad#Header,E_MFK.E(KC),MAC#Outgoing Pad#>
```

Response

```
<43#E_KPE_O(PIN Block)#Sanity Check Indicator#
IBM 3624 Sequence Number#>[CRLF]
```

Calling Parameters

`33`

Field 0, the command identifier.

`23`

Field 1, the PIN translation method; in this command, IBM 3624 to PIN/pad.

Header,$E_{MFK.E}$(KPE_I),MAC

Field 2, the incoming PIN Encryption Key (KPE_I) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

Header,$E_{MFK.E}$(KPE_O),MAC

Field 3, the outgoing PIN Encryption Key (KPE_O) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, the length of the KPEo must be equal to or greater than the length of the KPEi (field 2). The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000.

E_{KPE_I}(PIN Block)

Field 4, the incoming PIN block encrypted under the incoming PIN Encryption Key. This

field contains 16 hexadecimal characters.

`Incoming Pad`

Field 5, the pad character for the incoming PIN block. The field is one byte, it can contain a hexadecimal value, X, or W. The value X indicates any hexadecimal pad character is allowed. The value W indicates the sanity check will not be performed.

`Header,`$E_{MFK.E}$`(KC),MAC`

Field 6, the incoming Communications Key encrypted under the MFK. This key is used in the outer, or second, encryption of the IBM 3624 PIN block for the incoming PIN. This field contains a 74 byte value. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1DDNE000, 1DDDE000, and 1DDEE000.

`Outgoing Pad`

Field 7, the pad character for the outgoing PIN block. This field is 1 byte, it can contain a hexadecimal value, X, or W. The value X or W indicates that the pad character for the incoming PIN block will also be used as the outgoing pad character.

Table 4-45 Command 33: Translate PIN – IBM 3624 to PIN/Pad

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	33
1	PIN translation method (IBM 3624 to PIN/ pad)	2	23
2	Header,$E_{MFK.E}$(KPE$_I$),MAC	74*	printable ASCII
3	Header,$E_{MFK.E}$(KPE$_O$),MAC	74*	printable ASCII
4	E_{KPE_I}(PIN Block)	18	0 - 9, A - F
5	Incoming pad	1	0 - 9, A - F, X, W
6	Header,$E_{MFK.E}$(KC),MAC	74	printable ASCII
7	Outgoing pad	1	0 - 9, A - F, X, W

* Can be a volatile table location.

Responding Parameters

`43`

Field 0, the response identifier.

`E`$_{KPE_O}$`(PIN Block)`

Field 1, the outgoing encrypted PIN. This field contains 16 hexadecimal characters. When a PIN sanity error is detected, the value in this field may not be correct.

`Sanity Check Indicator`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

`IBM 3624 Sequence Number`

Field 3, the IBM 3624 sequence number. This field contains 2 hexadecimal characters.

Table 4-46 Response 43: Translate PIN – IBM 3624 to PIN/Pad

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	43
1	E$_{KPE_O}$(PIN block)	16	0 - 9, A - F
2	Sanity check indicator	1	Y, N, L
3	IBM 3624 sequence number	2	0 - 9, A - F

Usage Notes

- Generate the incoming and outgoing PIN encryption Keys and the Communications Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate a PIN formatted in an IBM 3624 PIN block to PIN/pad PIN block

- Clear-text incoming PIN Encryption Key (KPE_I): 07CE A74F 4607 5D8F
  The incoming PIN Encryption Key (KPE_I) in AKB format:
  1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,C47517937 1FDBE43
- Clear-text outgoing PIN Encryption Key (KPE_O): D029 23D9 AD4F E90B
  The outgoing PIN Encryption Key (KPE_O) in AKB format:
  1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53AF657A4508,7ED6957 0CDE54F30
- Encrypted incoming PIN block: 9864 AB86 5904 8084 B8
- Incoming pad character: B
- Clear-text Communications Key: A15D BAFD F119 F701
  The incoming Communications Key in AKB format:
  1DDNE000,740B352ADA6934929EECA387D23104F5582D59EF93C3AB88,571989DB B9DBC083
- Outgoing pad character: D

The command looks like this:

```
<33#23#1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,
C475179371FDBE43#1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53A
F657A4508,7ED69570CDE54F30#9864AB8659048084B8#B#1DDNE000,740B352A
DA6934929EECA387D23104F5582D59EF93C3AB88,571989DBB9DBC083#D#>
```

The Network Security Processor returns the following response:

```
<43#F9081E2639080784#Y#99#>
```

Translate PIN – PIN/Pad or Docutel to PIN/Pad (Command 33)

Command 33 – PIN/pad or Docutel to PIN/pad. This command translates an encrypted PIN block in either PIN/pad or Docutel PIN block to an encrypted PIN/pad PIN block. The incoming PIN Encryption Key is designated as KPE_I, and the outgoing PIN Encryption Key is designated as KPE_O.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

This command will return an error if either option 46 or 47 is enabled.

Command

<33#33#Header,$E_{MFK.E}$(KPE$_I$),MAC#Header,$E_{MFK.E}$(KPE$_O$),MAC#
E_{KPE_I}(PIN Block)#Incoming Pad#Outgoing Pad#>

Response

<43#E_{KPE_O}(PIN Block)#Sanity Check Indicator#>[CRLF]

Calling Parameters

`33`

Field 0, the command identifier.

`33`

Field 1, the PIN translation method; in this command, PIN pad character or Docutel to PIN pad.

Header,$E_{MFK.E}$(KPE$_I$),MAC

Field 2, the incoming PIN Encryption Key (KPE_I) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

Header,$E_{MFK.E}$(KPE$_O$),MAC

Field 3, the outgoing PIN Encryption Key (KPE_O) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, the length of the KPEo must be equal to or greater than the length of the KPEi (field 2). The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000.

E_{KPE_I}(PIN Block)

Field 4, the incoming PIN block encrypted under the incoming PIN Encryption Key. This field contains 16 hexadecimal characters.

`Incoming Pad`

Field 5, the pad character for the incoming PIN block. The field is one byte, it can contain a hexadecimal value, X, or W. The value X indicates any hexadecimal pad character is allowed. The value W indicates the sanity check will not be performed.

`Outgoing Pad`

Field 6, the pad character for the outgoing PIN block. This field is 1 byte, it can contain a hexadecimal value, X, or W. The value X or W indicates that the pad character for the incoming PIN block will also be used as the outgoing pad character.

Table 4-47 Command 33: Translate PIN – PIN/Pad or Docutel to PIN/Pad

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	33
1	PIN translation method (PIN/Pad to PIN/Pad)	2	33
2	Header,$E_{MFK.E}$(KPE_I),MAC*	74	printable ASCII
3	Header,$E_{MFK.E}$(KPE_O),MAC*	74	printable ASCII
4	E_{KPE_I}(PIN block)	16	0 - 9, A - F
5	Incoming pad	1	0 - 9, A - F, X, W
6	Outgoing pad	1	0 - 9, A - F, X, W

* Can be a volatile table location.

Responding Parameters

`43`

Field 0, the response identifier.

E_{KPE_O}(PIN Block)

Field 1, the outgoing, encrypted PIN. This field contains 16 hexadecimal characters. When a PIN sanity error is detected, the value in this field may not be correct.

`Sanity Check Indicator`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

Table 4-48 Response 43: Translate PIN – PIN/Pad or Docutel to PIN/Pad

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	43
1	E_{KPE_O}(PIN block)	16	0 - 9, A - F
2	Sanity check indicator	1	Y, N, L

Usage Notes

- Generate the incoming and outgoing PIN Encryption Keys.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate a PIN formatted in PIN/pad character PIN block to PIN/pad character PIN block

- Clear-text incoming PIN Encryption Key (KPE_I): 07CE A74F 4607 5D8F
  The incoming PIN Encryption Key (KPE_I) in AKB format:
  1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,C47517937 1FDBE43
- Clear-text outgoing PIN Encryption Key (KPE_O): D029 23D9 AD4F E90B
  The outgoing PIN Encryption Key (KPE_O) in AKB format:
  1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53AF657A4508,7ED6957 0CDE54F30
- The encrypted PIN block: 86EA C4C4 F7AE 03B8
- Incoming pad character: B
- Outgoing pad character: D

The command looks like this:

```
<33#33#1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,
C475179371FDBE43#1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53A
F657A4508,7ED69570CDE54F30#86EAC4C4F7AE03B8#B#D#>
```

The Network Security Processor returns the following response:

```
<43#F9081E2639080784#Y#>
```

Translate PIN – PIN/Pad or Docutel to IBM 4731 (Command 33)

Command 33 – PIN/Pad or Docutel to IBM 4731 translates an encrypted PIN block in either a PIN/Pad or Docutel PIN block, to an encrypted IBM 4731 PIN block. The incoming PIN Encryption Key is designated as KPE_I, and the outgoing PIN Encryption Key is designated as KPE_O.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

This command will return an error if either option 46 or 47 is enabled.

Command

```
<33#39#Header,EMFK.E(KPEI),MAC#Header,EMFK.E(KPEO),MAC#
EKPEI(PIN Block)#Incoming Pad#Outgoing Pad#Outgoing ICV#
Header,EMFK.E(KC),MAC#>
```

Response

```
<43#EKPEO(PIN Block)#Sanity Check Indicator#>[CRLF]
```

Calling Parameters

`33`

Field 0, the command identifier.

`39`

Field 1, the PIN translation method; in this command, PIN/Pad or Docutel to IBM 4731.

Header,$E_{MFK.E}(KPE_I)$,MAC

Field 2, the incoming PIN Encryption Key (KPE_I) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

Header,$E_{MFK.E}(KPE_O)$,MAC

Field 3, the outgoing PIN Encryption Key (KPE_O) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, the length of the KPEo must be equal to or greater than the length of the KPEi (field 2). The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000.

E_{KPE_I}(PIN Block)

Field 4, the incoming PIN block encrypted under the incoming PIN Encryption Key. This field contains 16 hexadecimal characters.

Incoming Pad

Field 5, the pad character for the incoming PIN block. The field is one byte, it can contain a hexadecimal value, X, or W. The value X indicates any hexadecimal pad character is allowed. The value W indicates the sanity check will not be performed.

Outgoing Pad

Field 6, the pad character for the outgoing PIN block. This field is 1 byte, it can contain a hexadecimal value, X, or W. The value X or W indicates that the pad character for the incoming PIN block will also be used as the outgoing pad character.

Outgoing ICV

Field 7, the sequence number for the outgoing PIN block. This field contains 16 hexadecimal characters.

Header,$E_{MFK.E}$(KC),MAC

Field 8, the Communications Key encrypted under the MFK. This key is used in the outer or second encryption of the IBM 4731 PIN block. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1c7NE000, 1c7NN000, 1c7EE000, and 1c7EN000.

Table 4-49 Command 33: Translate PIN – PIN/Pad or Docutel to IBM 4731

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	33
1	PIN translation method (ANSI to IBM 4731)	2	39
2	Header,$E_{MFK.E}$(KPE$_I$),MAC*	74	printable ASCII
3	Header,$E_{MFK.E}$(KPE$_O$),MAC*	74	printable ASCII
4	E_{KPE_I}(PIN block)	16	0 - 9, A - F
5	Incoming Pad	12	0 - 9, A - F, X, W
6	Outgoing Pad	1	0 - 9, A - F, X, W
7	Outgoing ICV	16	0 - 9, A - F
8	Header,$E_{MFK.E}$(KC),MAC*	74	printable ASCII

* Can be a volatile table location.

Responding Parameters

`43`

Field 0, the response identifier.

`E`$_{KPE_O}$`(PIN Block)`

Field 1, the outgoing, encrypted PIN. This field contains 16 hexadecimal characters. When a PIN sanity error is detected, the value in this field may not be correct.

`Sanity Check Indicator`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

Table 4-50 Response 43: Translate PIN – PIN/Pad or Docutel To IBM 4731

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	43
1	E$_{KPE_O}$(PIN block)	16	0 - 9, A - F
2	Sanity check indicator	1	Y, N, L

Usage Notes

- Generate the incoming and outgoing PIN Encryption Keys, and the Communications Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate a PIN formatted in PIN/Pad or Docutel PIN block to IBM 4731 PIN block

- Clear-text incoming PIN Encryption Key (KPE_I): 07CE A74F 4607 5D8F
  The incoming PIN Encryption Key (KPE_I) in AKB format:
  1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,C47517937 1FDBE43
- Clear-text outgoing PIN Encryption Key (KPE_O): D029 23D9 AD4F E90B
  The outgoing PIN Encryption Key (KPE_O) in AKB format:
  1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53AF657A4508,7ED6957 0CDE54F30
- The encrypted PIN block: 86EA C4C4 F7AE 03B8
- Incoming Pad character: B
- Outgoing Pad character D
- Outgoing ICV: 1234 1234 1234 1234
- Clear-text Communications Key: B302 AD91 F504 EA22
  The Communications Key in AKB format:
  1c7NE000,0B5822317FC002F870FB54EEEA2DAB004A454333FC6B60E4,45F760CD B48BB404

The command looks like this:

```
<33#39#1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,
C475179371FDBE43#1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53A
F657A4508,7ED69570CDE54F30#86EAC4C4F7AE03B8#B#D#1234123412341234#
1c7NE000,0B5822317FC002F870FB54EEEA2DAB004A454333FC6B60E4,45F760C
DB48BB404#>
```

The Network Security Processor returns the following response:

```
<43#27682B863CD388E8#Y#>
```

Translate PIN – IBM 4731 to PIN/Pad (Command 33)

Command 33 – IBM 4731 to PIN/Pad translates an encrypted IBM 4731 PIN block to an encrypted PIN/Pad PIN block. The incoming PIN Encryption Key is designated as KPE_I, and the outgoing PIN Encryption Key is designated as KPE_O.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

This command will return an error if either option 46 or 47 is enabled.

Command

```
<33#93#Header,E_MFK.E(KPE_I),MAC#Header,E_MFK.E(KPE_O),MAC#
E_KPE_I(PIN Block)#Incoming Pad#Incoming ICV#Header,E_MFK.E(KC),MAC#
Outgoing Pad#>
```

Response

```
<43#E_KPE_O(PIN Block)#Sanity Check Indicator#>[CRLF]
```

Calling Parameters

`33`

Field 0, the command identifier.

`93`

Field 1, the PIN translation method; in this command, IBM 4731 to PIN/Pad.

Header,$E_{MFK.E}(KPE_I)$,MAC

Field 2, the incoming PIN Encryption Key (KPE_I) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

Header,$E_{MFK.E}(KPE_O)$,MAC

Field 3, the outgoing PIN Encryption Key (KPE_O) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, the length of the KPEo must be equal to or greater than the length of the KPEi (field 2). The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000.

E_{KPE_I}(PIN Block)

Field 4, the incoming PIN block encrypted under the incoming PIN Encryption Key. This

field contains 16 hexadecimal characters.

`Incoming Pad`

Field 5, the pad character for the incoming PIN block. The field is one byte, it can contain a hexadecimal value, X, or W. The value X indicates any hexadecimal pad character is allowed. The value W indicates the sanity check will not be performed.

`Incoming ICV`

Field 6, the sequence number for the incoming PIN block. This field contains 16 hexadecimal characters.

`Header,`$E_{MFK.E}$`(KC),MAC`

Field 7, the Communications Key encrypted under the MFK. This key is used in the outer or second encryption of the IBM 4731 PIN block. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1c7NE000, 1c7NN000, 1c7DE000, and 1c7DN000.

`Outgoing Pad`

Field 8, the pad character for the outgoing PIN block. This field is 1 byte, it can contain a hexadecimal value, X, or W. The value X or W indicates that the pad character for the incoming PIN block will also be used as the outgoing pad character.

Table 4-51 Command 33: IBM 4731 to PIN/Pad

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	33
1	PIN translation method (IBM 4731 to PIN/Pad)	2	93
2	Header,$E_{MFK.E}$(KPE$_I$),MAC*	74	printable ASCII
3	Header,$E_{MFK.E}$(KPE$_O$),MAC*	74	printable ASCII
4	E_{KPE_I}(PIN block)	16	0 - 9, A - F
5	Incoming Pad	1	0 - 9, A - F, X, W
6	Incoming ICV	16	0 - 9, A - F
7	Header,$E_{MFK.E}$(KC),MAC*	0, 74	printable ASCII
8	Outgoing Pad	1	0 - 9, A - F, X, W

* Can be a volatile table location.

Responding Parameters

`43`

Field 0, the response identifier.

E_{KPE_O}`(PIN Block)`

Field 1, the outgoing, encrypted PIN. This field contains 16 hexadecimal characters. When a PIN sanity error is detected, the value in this field may not be correct.

`Sanity Check Indicator`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

Table 4-52 Response 43: IBM 4731 to PIN/Pad

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	43
1	E_{KPE_O}(PIN block)	16	0 - 9, A - F
2	Sanity check indicator	1	Y, N, L

Usage Notes

- Generate the incoming and outgoing PIN Encryption Keys, and the Communications Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate a PIN formatted in IBM 4731 PIN block to PIN/Pad PIN block

- Clear-text incoming PIN Encryption Key (KPE_I): D029 23D9 AD4F E90B
  The incoming PIN Encryption Key (KPE_I) in AKB format:
  1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53AF657A4508,7ED6957 0CDE54F30
- Clear-text outgoing PIN Encryption Key (KPEO): 07CE A74F 4607 5D8F
  The outgoing PIN Encryption Key (KPEO) in AKB format:
  1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,C47517937 1FDBE43
- The encrypted PIN block: 2768 2B86 3CD3 88E8
- Incoming Pad character: D
- Incoming ICV: 1234 1234 1234 1234
- Clear-text Communications Key: B302 AD91 F504 EA22
  The Communications Key in AKB format:
  1c7NE000,0B5822317FC002F870FB54EEEA2DAB004A454333FC6B60E4,45F760CD B48BB404
- Outgoing Pad character: B

The command looks like this:

```
<33#93#1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53AF657A4508,
7ED69570CDE54F30#1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDF
CF431B0C0,C475179371FDBE43#27682B863CD388E8#D#1234123412341234#1c
7NE000,0B5822317FC002F870FB54EEEA2DAB004A454333FC6B60E4,45F760CDB
48BB404#B#>
```

The Network Security Processor returns the following response:

```
<43#86EAC4C4F7AE03B8#Y#>
```

Translate PIN – IBM 4731 to IBM 4731 (Command 33)

Command 33 – IBM 4731 to IBM 4731 translates an encrypted IBM 4731 PIN block to an encrypted IBM 4731 PIN block. The incoming PIN Encryption Key is designated as KPE_I, and the outgoing PIN Encryption Key is designated as KPE_O. The incoming Communications Key is designated as KC_I, and the outgoing Communications Key is designated as KC_O.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

This command will return an error if either option 46 or 47 is enabled.

Command

<33#99#Header,$E_{MFK.E}$(KPE$_I$),MAC#Header,$E_{MFK.E}$(KPE$_O$),MAC#
E_{KPE_I}(PIN Block)#Incoming Pad#Incoming ICV#Header,$E_{MFK.E}$(KC$_I$),MAC#
Outgoing Pad#Outgoing ICV#Header,$E_{MFK.E}$(KC$_O$),MAC#>

Response

<43#E_{KPE_O}(PIN Block)#Sanity Check Indicator#>[CRLF]

Calling Parameters

`33`

Field 0, the command identifier.

`99`

Field 1, the PIN translation method; in this command, IBM 4731 to IBM 4731.

Header,$E_{MFK.E}$(KPE$_I$),MAC

Field 2, the incoming PIN Encryption Key (KPE_I) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

Header,$E_{MFK.E}$(KPE$_O$),MAC

Field 3, the outgoing PIN Encryption Key (KPE_O) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, the length of the KPEo must be equal to or greater than the length of the KPEi (field 2). The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000.

E_{KPE_I}`(PIN Block)`

Field 4, the incoming PIN block encrypted under the incoming PIN Encryption Key. This field contains 16 hexadecimal characters.

`Incoming Pad`

Field 5, the pad character for the incoming PIN block. The field is one byte, it can contain a hexadecimal value, X, or W. The value X indicates any hexadecimal pad character is allowed. The value W indicates the sanity check will not be performed.

`Incoming ICV`

Field 6, the sequence number for the incoming PIN block. This field contains 16 hexadecimal characters.

`Header,`$E_{MFK.E}$`(`KC_I`),MAC`

Field 7, the incoming Communications Key encrypted under the MFK. This key is used in the outer or second encryption of the IBM 4731PIN block. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1c7NE000, 1c7NN000, 1c7DE000, and 1c7DN000.

`Outgoing Pad`

Field 8, the pad character for the outgoing PIN block. This field is 1 byte, it can contain a hexadecimal value, X, or W. The value X or W indicates that the pad character for the incoming PIN block will also be used as the outgoing pad character.

`Outgoing ICV`

Field 9, the sequence number for the outgoing PIN block. This field contains 16 hexadecimal characters.

`Header,`$E_{MFK.E}$`(`KC_O`),MAC`

Field ten, the outgoing Communications Key encrypted under the MFK. This key is used in the outer or second encryption of the IBM 4731 PIN block. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1c7NE000, 1c7NN000, 1c7EE000, and 1c7EN000.

Table 4-53 Command 33: IBM 4731 to IBM 4731

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	33
1	PIN translation method (IBM 4731 to IBM 4731)	2	19

Table 4-53 Command 33: IBM 4731 to IBM 4731 (continued)

Field	Contents	Length (bytes)	Legal Characters
2	Header,$E_{MFK.E}(KPE_I)$,MAC*	74	printable ASCII
3	Header,$E_{MFK.E}(KPE_O)$,MAC*	74	printable ASCII
4	E_{KPE_I}(PIN block)	16	0 - 9, A - F
5	Incoming Pad	1	0 - 9, A - F, X, W
6	Incoming ICV	16	0 - 9, A - F
7	Header,$E_{MFK.E}$(incoming KC),MAC*	74	printable ASCII
8	Outgoing Pad	1	0 - 9, A - F, X, W
9	Outgoing ICV	16	0 - 9, A - F
10	Header,$E_{MFK.E}$(outgoing KC),MAC*	74	printable ASCII

* Can be a volatile table location.

Responding Parameters

`43`

Field 0, the response identifier.

E_{KPE_O}`(PIN Block)`

Field 1, the outgoing, encrypted PIN. This field contains 16 hexadecimal characters. When a PIN sanity error is detected, the value in this field may not be correct.

`Sanity Check Indicator`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

Table 4-54 Response 43: IBM 4731 to IBM 4731

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	43
1	E_{KPE_O}(PIN block)	16	0 - 9, A - F
2	Sanity check indicator	1	Y, N, L

Usage Notes

- Generate the incoming and outgoing PIN Encryption Keys, and the Communications Keys.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate a PIN formatted in an IBM 4731 PIN block to IBM 4731 PIN block

- Clear-text incoming PIN Encryption Key: C8B3 047C F7A4 2A70
  The incoming PIN Encryption Key in AKB format:
  1PUNE000,049A29CB8226E877EB409BDE8D9FC1E3F98AA5E95A5A91A6,50971DF0B16A6017
- Clear-text outgoing PIN Encryption Key: 2222 2222 2222 2222
  The outgoing PIN Encryption Key in AKB format:
  1PUNE000,C3A625D794342980421721B126A2AB340C1C40CBB6BAEB6B,198DA9ABACCC4CF0
- The encrypted PIN Block: DE45 A161 F371 9346
- Incoming Pad character: F
- Incoming ICV: 0000 1560 0065 0039
- Clear-text incoming Communications Key: 68D5 9437 1067 794F
  The Communications Key in AKB format:
  1c7NE000,2C5444E9BB68CE838CD3239A21B7158663C3C776D540099A,CA728C6561BA524C
- Outgoing Pad Character: D
- Outgoing ICV: 1234123412341234

- Clear-text outgoing Communications Key: 0123 4567 89AB CDEF
  The outgoing Communications Key in AKB format:
  1c7NE000,F374434984B177DCE6FA16489F4B7AA8813BE2E373039AE8,3135702BF3D27400

The command looks like this:

```
<33#99#1PUNE000,049A29CB8226E877EB409BDE8D9FC1E3F98AA5E95A5A91A6,
50971DF0B16A6017#1PUNE000,C3A625D794342980421721B126A2AB340C1C40C
BB6BAEB6B,198DA9ABACCC4CF0#DE45A161F3719346#F#0000156000650039#1c
7NE000,2C5444E9BB68CE838CD3239A21B7158663C3C776D540099A,CA728C656
1BA524C#D#1234123412341234#1c7NE000,F374434984B177DCE6FA16489F4B7
AA8813BE2E373039AE8,3135702BF3D27400#>
```

The Network Security Processor returns the following response:

```
<43#BA272DB1D8BE0196#Y#>
```

Translate PIN – Double-Encrypted Input or Output (Command 35)

Command 35 – decrypts and re-encrypts an encrypted PIN block, where the input or output is double encrypted. The incoming Communication Key is designated as KC_I, and the outgoing Communications Key is designated as KC_O. The Electronic Code Book Mode of DES is used for the encryption process.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

Command

```
<35#[Header,E_MFK.E(KC_I),MAC]#[Header,E_MFK.E(KC_O),MAC]#
PIN Information#>
```

Response

```
<45#E_KPE(PIN Block)#Sanity Check Indicator#>[CRLF]
```

Calling Parameters

`35`

Field 0, the command identifier.

`[Header,`$E_{MFK.E}$`(`KC_I`),MAC]`

Field 1, the incoming Communications Key, used in the second or outer encryption of the incoming PIN, under the MFK. This field is either empty, a 74 byte value or a volatile table location. If this field is empty, the incoming PIN is single-encrypted. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1DDNE000, 1DDNN000, 1DDDE000, 1DDDN000.

`[Header,`$E_{MFK.E}$`(`KC_O`),MAC]`

Field 2, the outgoing Communications Key, used in the second or outer encryption of the outgoing PIN, encrypted under the MFK. This field is either empty, a 74 byte value, or a volatile table location. If this field is empty, the outgoing PIN is single-encrypted. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, the length of the KPEo must be equal to or greater than the length of the KPEi (field 2). The following headers are supported: 1DDNE000, 1DDNN000, 1DDEE000, and 1DDEN000.

`PIN Information`

Field 3, identical to the fields from Commands 31 and 33 that assume a single-encrypted PIN, beginning with Field 1, which specifies the PIN block type for Command 31 or the translation method for Command 33, and includes all subsequent fields to the end. This command can be used with the following PIN block types:

Value	PIN Block Type
1	ANSI
3	PIN pad / Docutel
4	IBM Encrypting PIN Pad
5	Burroughs

When option 46 is enabled, this field can only contain the value 1 (ANSI). When option 47 is enabled and option 46 is disabled, the outgoing PIN block type specified in this command must be ANSI.

Table 4-55 Command 35: Translate PIN – Double-Encrypted Input or Output

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier.	2	35
1	[Header,$E_{MFK.E}$(KC_I),MAC]*	0, 74	printable ASCII
2	[Header,$E_{MFK.E}$(KC_O),MAC]*	0, 74	printable ASCII
3	PIN information		

* Can be a volatile table location.

Responding Parameters

`45`

Field 0, the response identifier.

`E`$_{KPE}$`(PIN Block)`

Field 1, the re-encrypted PIN block. This field contains 16 hexadecimal characters. When a PIN sanity error is detected, the value in this field may not be correct. When a PIN sanity error is detected, and option 4B is enabled and the PIN block type (field 3) value is 1, this field will contain 16 zeros.

`Sanity Check Indicator`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

Table 4-56 Response 45: Translate PIN – Double-Encrypted Input or Output

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	2	45
1	E_{KPE}(PIN block)	16	0 - 9, A - F
2	Sanity check indicator	1	Y, N, L

Usage Notes

- This command utilizes the logic of commands 31 and 33, and therefore inherits their same restrictions and requirements.
- Generate the incoming and outgoing PIN Encryption Keys, and the Communications Keys.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate a double-encrypted ANSI PIN block

- Clear-text incoming Communications Key: 0123 4567 89AB CDEF
  The incoming Communications Key in AKB format:
  1DDNE000,4509C24B1C13C3798D30B3AEAEB65DE1CEFB3E831C522F7F,1CE66C3D258AF755
- Clear-text outgoing Communications Key: 3333 3333 3333 3333
  The outgoing Communications Key in AKB format:
  1DDNE000,09FFCD303017DA193AE10630F7A97FB04EAC17E4CA01A1FC,966113F44A28E527

- PIN information; the PIN block type, the cryptogram of the incoming PIN Encryption Key, the cryptogram of the outgoing PIN Encryption Key, the cryptogram of the PIN block, and PIN block data. (In the case of ANSI-formatted PINs, the PIN block data consists of the 12 rightmost digits of the Primary Account Number excluding the check digit.
    - PIN block type: ANSI (1)
    - Clear-text incoming PIN Encryption Key: 0000 1111 2222 3333
      The incoming PIN Encryption Key in AKB format:
      1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D60C5A5CA83A4D1258,0DBB2D4988A072F2
    - Clear-text outgoing PIN Encryption Key: 1111 2222 3333 4444
      The outgoing PIN Encryption Key in AKB format:
      1PUNE000,1AB5E6DC613047FD2672333FD0E02A1828ED8A4C449A5299,4421D569680BFC5C
    - Clear-text incoming ANSI PIN block: 0512 AC29 ABCD EFED
      The ANSI PIN block encrypted under the incoming PIN Encryption Key: D4B5 321B 0249 0316
    - Twelve rightmost digits of the Primary Account Number: 9876 5432 1012

The command looks like this:

```
<35#1DDNE000,4509C24B1C13C3798D30B3AEAEB65DE1CEFB3E831C522F7F,1CE
66C3D258AF755#1DDNE000,09FFCD303017DA193AE10630F7A97FB04EAC17E4CA
01A1FC,966113F44A28E527#1#1PUNE000,C118AFA8BDA9F01E832B5725DFCE45
D60C5A5CA83A4D1258,0DBB2D4988A072F2#1PUNE000,1AB5E6DC613047FD2672
333FD0E02A1828ED8A4C449A5299,4421D569680BFC5C#D4B5321B02490316#98
7654321012#>
```

The Network Security Processor returns the following response:

```
<45#83808A52FE942E1F#Y#>
```

Verify Double-Encrypted PIN (Command 36)

Command 36 decrypts and verifies an incoming, double-encrypted PIN. The PIN is encrypted using a PIN Encryption Key, the resulting cryptogram is then encrypted using a Communications Key. The Electronic Code Book Mode of DES is used for the encryption process. This command supports these PIN Verification methods: Identikey, IBM3624, Visa, Atalla DES (Bilevel), Diebold, NCR, Burroughs, and Atalla 2x2.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

If option 8C is enabled the Visa PVV key can be a 1key-3DES (single-length) key.

Command

```
<36#[Header,EMFK.E(KC),MAC]#PIN Information#>
```

Response

```
<46#Sanity Check Indicator/Verification Flag#>[CRLF]
```

Calling Parameters

`36`

Field 0, the command identifier.

`[Header,EMFK.E(KC),MAC]`

Field 1, the Communications Key, used in the second or outer encryption of the incoming PIN, encrypted under the MFK. This field is either empty, a 74 byte value, or a volatile table location. If this field is empty, the incoming PIN has been single-encrypted. The following headers are supported: 1DDNE000, 1DDNN000, 1DDDE000, 1DDDN000.

`PIN Information`

Field 2, identical to the fields from Command 32 that assume a single-encrypted PIN, beginning with Field 1 which specifies the PIN verification method and including all subsequent fields to the end. This command can be used with the following PIN block types.

Value	PIN Block Type
1	ANSI
3	PIN pad / Docutel

Value	PIN Block Type
4	IBM Encrypting PIN Pad
5	Burroughs

Table 4-57 Command 36: Verify Double-Encrypted PIN

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	36
1	Header,$E_{MFK.E}$(KC),MAC*	0, 74	printable ASCII
2	PIN information**		

* Can be a volatile table location.
** Fields from Command 32.

Responding Parameters

```
46
```

Field 0, the response identifier.

```
Sanity Check Indicator/Verification Flag
```

Field 1, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. If the PIN block passes the sanity check, the verification check is conducted. This field can contain one of the following values:

Value	Description
Y	PIN verification was successful.
N	PIN verification failed.
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

Table 4-58 Response 46: Verify Double-Encrypted PIN

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	46
1	Sanity check indicator/verification flag	1	Y, N, S, L

Usage Notes

- This command utilizes the logic of command 32, and therefore inherits the same restrictions and requirements.
- Generate the PIN Encryption Keys, Communications Keys and PIN Verification Keys.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify a double-encrypted PIN using the Visa PIN verification method

- Clear-text Communications Key: 3333 3333 3333 3333
  The Communications Key in AKB format:
  1DDNE000,09FFCD303017DA193AE10630F7A97FB04EAC17E4CA01A1FC,966113F44A28E527
- PIN information
  - Verification method: VISA (3)
  - PIN block type: PIN/pad character (3)
  - Double-encrypted PIN block: 818E 3942 0AA0 F83B
  - Clear-text incoming PIN Encryption Key: 0000 1111 2222 3333
    The incoming PIN Encryption Key in AKB format:
    1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D60C5A5CA83A4D1258,0DBB2D4988A072F2
  - Clear-text Key pair: 4CA2161637D0133E 5E151AEA45DA2A16
    The Key pair in AKB format:
    1VVNE000,C56554CDE94948D004EB47FF82A81D8C24F89AFF4C47776C,9DE7167F56F172D7
  - PIN Verification Value: 3691
  - PIN Verification Key Indicator: 3
  - PAN: 1234 5678 901
  - PIN block data:
    - — Pad character: B
    - — Twelve Primary Account Number digits: 1234 5678 9019

The command looks like this:

```
<36#1DDNE000,09FFCD303017DA193AE10630F7A97FB04EAC17E4CA01A1FC,966
113F44A28E527#3#3#818E39420AA0F83B#1PUNE000,C118AFA8BDA9F01E832B5
725DFCE45D60C5A5CA83A4D1258,0DBB2D4988A072F2#1VVNE000,C56554CDE94
948D004EB47FF82A81D8C24F89AFF4C47776C,9DE7167F56F172D7##3691#3#12
345678901#B#123456789019#>
```

The Network Security Processor returns the following response:

```
<46#Y#>
```

PIN Change – Identikey (Command 37)

Command 37 – Identikey verifies the old PIN using the Atalla Identikey method. If the old PVN verifies, a PVN, based on the new PIN, will be generated.

You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

This command has the ability to generate a new PVN without verifying the OLD PIN. This functionality has a high security exposure. You must purchase option 66 in the form of a command 105, and enable it in the Network Security Processor's security policy.

Command

```
<37#1#PIN Block Type#EKPE(Old PIN Block)#Header,EMFK.E(KPE),MAC#
Bank ID#PVN#Comparison Indicator#Partial PAN#EKPE(New PIN Block)#
PIN Block Data#>
```

Response

```
<47#Sanity Check Indicator#[PVN#][IBM 3624 Sequence Number#]>[CRLF]
```

Calling Parameters

`37`

Field 0, the command identifier.

`1`

Field 1, the PVN verification/generation technique (Identikey) to be used. This field is 1 byte, it contains the decimal value 1.

`PIN Block Type`

Field 2, the PIN block type. This field is 1 byte, it can contain the numbers 1 through 5.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN/pad character / Docutel
4	IBM encrypting PIN pad
5	Burroughs

E_{KPE}`(Old PIN Block)`

Field 3, the old encrypted PIN. When this field is empty and option 66 is enabled, the PIN verification step is not performed before the new PVN is generated. This field is either empty, or a 16, or 18 byte hexadecimal value.

`Header,`$E_{MFK.E}$`(KPE),MAC`

Field 4, the PIN Encryption Key encrypted under the MFK. This field must be a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

`Bank ID`

Field 5, the bank ID; clear-text or encrypted. The clear-text Bank ID can be a two, six, or eight digit number.

Bank ID	**Allowable Size (bytes)**
Backward index (algorithm number less than 65)	2
ISO number	6
Route and transfer number	8

The encrypted bank ID is a 74 byte AKB comprised of the following four data fields: ll, bbbbbbbb, p, and cc. The following headers are supported: 1VINE000, 1VINN000, 1VIVE000, and 1VIVN000.

ll - a two-digit number; the length of the Bank ID:

- 02 – The Bank ID in backward index format; the algorithm number must be less than 65.
- 06 – The Bank ID is a six digit ISO number.
- 08 – The Bank ID is an eight digit route-and-transfer number.

bbbbbbbb - The bank ID number (digits 0 - 9); must be the same length as ll.

p - The pad character F, right pads the combined length of the bank ID length (ll) and the bank ID value (bb - bbbbbbbb) resulting in 14 hexadecimal characters. Four pad characters are required when the bank ID is an eight digit value. Six pad characters are required when the bank ID is an six digit value. Ten pad characters are required when the bank ID is a two digit value.

cc - The two hexadecimal character comparison indicator. This field specifies the group (left, middle, or right) of four digits of the six-digit Identikey PIN Verification Number that will be used for the comparison.

- 4C – Compare the leftmost four digits.
- 4D – Compare the middle four digits.

- 52 – Compare the rightmost four digits.

`PVN`

Field 6, the PIN Verification Number. The PVN can be four, six, or eight digits in length, containing the numbers 0 to 7.

`Comparison Indicator`

Field 7, a comparison indicator that specifies which four digits (left, middle, or right) of the six-digit PVN will be compared. This field is 1 byte, and can contain the character L, M, or R. When the PVN is six or eight digits in length or field 5 contains an encrypted bank ID, the value of this field is not evaluated by the Network Security Processor.

`Partial PAN`

Field 8, the portion of the Primary Account Number to be used for verification. This field contains a 4 to 19 byte decimal value.

E_{KPE}`(New PIN Block)`

Field 9, the encrypted new PIN block in the type specified in field 2. This field contains a 16 or 18 byte hexadecimal value.

`PIN Block Data`

Field 10, PIN block data. The content and number of fields depend on the PIN block type. See PIN Block Types.

Table 4-59 Command 37: PIN Change – Identikey

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	2	37
1	PIN verification method (Identikey)	1	1
2	PIN block type	1	1 - 5
3	E_{KPE}(Old PIN Block)	0,16, 18	0 - 9, A - F
4	Header,$E_{MFK.E}$(KPE),MAC*	74	printable ASCII
5	Bank ID	2, 6, 8, or 74	0 - 9 or printable ASCII
6	PIN verification number	4,6,8	0 - 9
7	Comparison indicator	1	L,M,R
8	Partial PAN	4 - 19	0 - 9
9	E_{KPE}(New PIN Block)	16, 18	0 - 9, A - F

Table 4-59 Command 37: PIN Change – Identikey (continued)

Field	Contents	Length (bytes)	Legal Characters
10	PIN block data**	Variable	

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

`47`

Field 0, the response identifier.

`Sanity Check Indicator/Verification Flag`

Field 1, the sanity check indicator and verification flag. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	Old PIN verified successfully.
N	Old PIN failed to verify.
LO	Old PIN length error. See option A1.
NO	No Old PIN. See option 66.
SO	Old PIN sanity error. See PIN Sanity Error.
LN	New PIN length error. See option A1.
SN	New PIN sanity error. See PIN Sanity Error.

`[PVN#]`

Field 2, the PVN associated with the new PIN if the operation completed successfully. This field will be present only when field 1 contains either “Y” or “NO”.

`[IBM 3624 Sequence Number#]`

Field 3, the IBM 3624 sequence number. This field is returned only if the PIN block type is IBM 3624. When present, this field contains 2 hexadecimal characters.

Table 4-60 Response 47: PIN Change – Identikey

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	2	47
1	PIN block OK or Sanity Error	1,2	Y, N, SO, SN, LO, LN, or NO

Table 4-60 Response 47: PIN Change – Identikey

Field	Contents	Length (bytes)	Legal Characters
2	[PVN#]	0,4,6,8	0 - 9
3	IBM 3624 sequence number*	2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

- The new PIN that can be a different length than the old PIN.
- The new and old PIN blocks used in the command must always be the same PIN block type, and encrypted under the same PIN Encryption Key (KPE).

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify a PIN and Generating a PVN using the Atalla Identikey method

- Verification method: Identikey (1)
- PIN block type: ANSI (1)
- Clear-text old ANSI PIN block: 0412 26CB A9ED CBA9
  The old ANSI PIN block encrypted under the PIN Encryption Key: C84F 6825 74BB AA20.
- Clear-text PIN Encryption Key: 1111 1111 1111 1111
  The PIN Encryption Key in AKB format:
  1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC785574,003D0F146BFFC1F9
- Clear-text new ANSI PIN block: 0443 33CB A9ED CBA9
  The new ANSI PIN block encrypted under the PIN Encryption Key: 090E 8CA3 CF5D 2AD8
- Bank ID: 26
- Expected PVN: 62732551.
- Since all eight PVN digits are provided, the comparison indicator is not used. The letter “L” is being used strictly as a placeholder.
- The PAN digits used in the algorithm: 1234 5612 3456

The command looks like this:

```
<37#1#1#C84F682574BBAA20#1PUNE000,302A81872F21CE1D36DCF0204EEB0FD
CF679984BFC785574,003D0F146BFFC1F9#26#62732551#L#123456123456#090
E8CA3CF5D2AD8#123456123456#>
```

The Network Security Processor returns the following response:

```
<47#Y#31724120#>
```

Generate a new PVN without verifying the old PIN

This example shows how to use the command when there is no old PIN.

The command looks like this:

```
<37#1#1##1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC78557
4,003D0F146BFFC1F9#26#00000000#L#123456123456#090E8CA3CF5D2AD8#12
3456123456#>
```

The Network Security Processor returns the following response:

```
<47#NO#31724120#>
```

PIN Change – IBM 3624 (Command 37)

Command 37 – IBM 3624 verifies the old PIN using the IBM 3624 method of PIN verification. If the old offset is verified, an offset, based on the new PIN, will be generated.

You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

This command has the ability to generate a new offset without verifying the OLD PIN. This functionality has a high security exposure. You must purchase option 66 in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

<37#2#PIN Block Type#E$_{KPE}$(Old PIN Block)#Header,E$_{MFK.E}$(KPE),MAC#
Conversion Table#Offset#Validation Data#Pad#Check-Length#
Header,E$_{MFK.E}$(KPV),MAC#E$_{KPE}$(New PIN Block)#PIN Block Data#>

Response

```
<47#Sanity Check Indicator#[Offset#][IBM 3624 Sequence Number#]>
[CRLF]
```

Calling Parameters

`37`

Field 0, the command identifier.

`2`

Field 1, the offset verification/generation technique IBM 3624.

`PIN Block Type`

Field 2, the PIN block type. This field is 1 byte, it can contain the numbers 1 through 5.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN/pad character / Docutel
4	IBM encrypting PIN pad
5	Burroughs

E$_{KPE}$(Old PIN Block)

Field 3, the old encrypted PIN in the PIN block type specified in Field 2. When this field is empty and option 66 is enabled, the PIN verification step is not performed before the NEW offset is generated. This field is empty, or a 16 or 18 byte hexadecimal value.

`Header,`$E_{MFK.E}$`(KPE),MAC`

Field 4, the PIN Encryption Key encrypted under the MFK. This field must be a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

`Conversion Table`

Field 5, a table that maps hexadecimal digits (0 through 9, A through F) to decimal digits (0 through 9). This field contains a 16 byte decimal value containing the clear-text Conversion Table or a volatile table location. When option 48 is enabled, this field contains the conversion table in AKB format (the header must be 1nCNE000) or a volatile table location. When option 4E is enabled, all three forms of the conversion table (clear-text, decrypted, or value stored in volatile table location) must adhere to these rules:

- The conversion table must have at least eight unique digits.
- No single digit can occur more than four times.

`Offset`

Field 6, an offset value applied to the algorithm-generated PIN before comparing it with the customer-entered PIN. This field contains a 4 to 12 byte decimal value.

`Validation Data`

Field 7, validation data. This value is unique for each card holder, and is typically the account number. This field contains a 4 to 16 byte decimal value. When the PIN block type is ANSI (field 1 = 1) and option 4C is enabled, the value supplied in this field must be 12 digits in length and equal to the PIN Block Data value supplied in field 12.

`Pad`

Field 8, the pad character which right-pads the validation data. This field contains a one byte hexadecimal value. The pad character is only applied when the validation data is less than 16 digits.

`Check-Length`

Field 9, the check-length. This value is typically the PIN length and determines the number of PIN digits to be compared. This field contains one hexadecimal character in the range of 4 through C.

`Header,`$E_{MFK.E}$`(KPV),MAC`

Field 10, the PIN Verification Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 4A is enabled, this key can be a 1key-3DES (single-length) key. The following headers are supported: 1V3NE000 and 1V3NN000.

When field 3 contains an encrypted PIN block, these additional headers are also supported: 1V3VE000 and 1V3VN000. When option 66 is enabled and field 3 is empty, only these headers are supported: 1V3NE000, 1V3NN000,1V3GE000, and 1V3GN000.

`E`$_{KPE}$`(New PIN Block)`

Field 11, the encrypted new PIN block in the type specified in Field 2. This field contains a 16 or 18 byte hexadecimal value.

`PIN Block Data`

Field 12, PIN block data. The content and number of fields depend on the PIN block type. See PIN Block Types.

Table 4-61 Command 37: PIN Change - IBM 3624

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	37
1	PIN verification method (IBM 3624)	1	2
2	PIN block type	1	1 - 5
3	E$_{KPE}$(Old PIN Block)	0, 16, 18	0 - 9, A - F
4	Header,E$_{MFK.E}$(KPE),MAC*	74	printable ASCII
5	Conversion table*	16	0 - 9
6	Offset	4 - 16	0 - 9
7	Validation data	4 - 16	0 - 9, A - F
8	Pad	1	0 - 9, A - F
9	Check-Length	1	4 - 9, A - C
10	Header,E$_{MFK.E}$(KPV),MAC*	74	printable ASCII
11	E$_{KPE}$(New PIN Block)	16, 18	0 - 9, A - F
12	PIN block data**	variable	

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

`47`

Field 0, the response identifier.

```
Sanity Check Indicator/Verification Flag
```

Field 1, the sanity check indicator and verification flag. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	Old PIN verified successfully.
N	Old PIN failed to verify.
LO	Old PIN length error. See option A1.
NO	No Old PIN. See option 66.
SO	Old PIN sanity error. See PIN Sanity Error.
LN	New PIN length error. See option A1.
SN	New PIN sanity error. See PIN Sanity Error.

```
[Offset#]
```

Field 2, the Offset associated with the new PIN if the operation completed successfully. This field will be present only when field 1 contains either “Y” or “NO”.

```
[IBM 3624 Sequence Number#]
```

Field 3, the IBM 3624 sequence number. This field is returned only if the PIN block type is IBM 3624. When present, this field contains 2 hexadecimal characters.

Table 4-62 Response 47: PIN Change - IBM 3624

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	2	47
1	Sanity Check Indicator	1,2	Y, N, SO, SN, LO, LN, or NO
2	[Offset#]	0, 4 - 12	0 - 9
3	IBM 3624 Sequence Number*	2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

- The new PIN that can be a different length than the old PIN.
- The new and old PIN blocks used in the command must always be the same PIN block type, and encrypted using the same KPE.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify a PIN and Generating an Offset using the IBM 3624 method

- Verification method: IBM 3624 (2)
- PIN block type: IBM 3624 (2)
- Clear-text old IBM 3624 PIN block: 1234 BBBB BBBB BBBB
  The old IBM 3624 PIN block encrypted under the PIN Encryption Key: 5F08 7319 FADF 613A 47
- Clear-text PIN Encryption Key: 1111 1111 1111 1111
  The PIN Encryption Key in AKB format:
  1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC785574,003D0F146BFFC1F9
- Clear-text PIN Verification Key: 3333 3333 3333 3333
  The PIN Verification Key in AKB format:
  1V3NE000,F0BD2DA1B14F54B16E5541B6F61C35AA4F56EDEFFB7ADDD3,450CC0232392B2F5
- Clear-text new IBM 3624 PIN block: 4321 BBBB BBBB BBBB
  The new IBM 3624 PIN block encrypted under the PIN Encryption Key: 9786 21BD 6421 2AAE 92
- PIN block data:
  - Pad character: B
  - Twelve digits of the Primary Account Number: 1234 5612 3456
  - Clear-text Communications Key: 2222 2222 2222 2222
    The Communications Key in AKB format:
    1DDNE000,B6790C480725B127035165B51C9D43163EAC111AE8E47607,0BFB32AF7CFBC885

The command looks like this:

```
<37#2#2#5F087319FADF613A47#1PUNE000,302A81872F21CE1D36DCF0204EEB0FD
CF679984BFC785574,003D0F146BFFC1F9#0123456789012345#3053#1234561234
56#F#4#1V3NE000,F0BD2DA1B14F54B16E5541B6F61C35AA4F56EDEFFB7ADDD3,45
0CC0232392B2F5#978621BD64212AAE92#B#123456123456#1DDNE000,B6790C480
725B127035165B51C9D43163EAC111AE8E47607,0BFB32AF7CFBC885#>
```

The Network Security Processor returns the following response:

```
<47#Y#6140#FF#>
```

Generate an IBM 3624 offset without verifying the old PIN

The command looks like this:

```
<37#2#2##1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC78557
4,003D0F146BFFC1F9#0123456789012345#3053#123456123456#F#4#1V3NE00
0,F0BD2DA1B14F54B16E5541B6F61C35AA4F56EDEFFB7ADDD3,450CC0232392B2
F5#978621BD64212AAE92#B#123456123456#1DDNE000,B6790C480725B127035
165B51C9D43163EAC111AE8E47607,0BFB32AF7CFBC885#>
```

The Network Security Processor returns the following response:

```
<47#NO#6140#FF#>
```

PIN Change – VISA (Command 37)

Command 37 – VISA verifies the old PIN using the VISA verification method. If the old PIN Verification Value (PVV) is verified, a PVV based on the new PIN will be generated.

You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor’s security policy.

This command has the ability to generate a new PVV without verifying the OLD PIN. This functionality has a high security exposure. You must purchase option 66 in the form of a command 105, and then enable it in the Network Security Processor’s security policy.

Command

```
<37#3#PIN Block Type#E_KPE(Old PIN Block)#Header,E_MFK.E(KPE),MAC#
Header,E_MFK.E(KPV),MAC#Reserved#PVV#PVKI#PAN#E_KPE(New PIN Block)#
PIN Block Data#>
```

Response

```
<47#Sanity Check Indicator#[PVV#][IBM 3624 Sequence Number#]>[CRLF]
```

Calling Parameters

`37`

Field 0, the command identifier.

`3`

Field 1, the PVV verification/generation technique; VISA.

`PIN Block Type`

the PIN block type. This field is 1 byte, it can contain the numbers 1 through 5.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN/pad character / Docutel
4	IBM encrypting PIN pad
5	Burroughs

E_{KPE}`(Old PIN Block)`

Field 3, the old encrypted PIN. When this field is empty and option 66 is enabled, the PIN verification step is not performed before the NEW PVV is generated. This field is empty, or a 16 or 18 byte hexadecimal value.

`Header,`$E_{MFK.E}$`(KPE),MAC`

Field 4, the PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

`Header,`$E_{MFK.E}$`(KPV),MAC`

Field 5, the PIN Verification Key pair (KPV) encrypted under the MFK. Both keys of the Visa key pair must be in the same Atalla Key Block. This field contains a 74 byte value, or a volatile table location. This key must be either a 2key- or 3key-3DES key. If option 8C is enabled, this key can be a 1key-3DES (single-length) key. The following headers are supported: 1VVNE000 and 1VVNN000. When field 3 contains an encrypted PIN block, these additional headers are also supported: 1VVVE000 and 1VVVN000. When option 66 is enabled and field 3 is empty, only these headers are supported: 1VVNE000, 1VVNN000, 1VVGE000, and 1VVGN000.

`Reserved`

Field 6, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

`PVV`

Field 7, the PIN Verification Value used to compare to the calculated value. This field contains a 4 byte decimal value. If there is no old PIN to verify, this field should contain four zeros.

`PVKI`

Field 8, the PIN Verification Key Indicator used to calculate the PIN Verification Value. This field is 1 byte, it can contain the numbers 0 through 9.

`PAN`

Field 9, the rightmost eleven Primary Account Number digits, excluding the check digit. When the PIN block type is ANSI (field 1 = 1) and option 4C is enabled, this value must be present in the PIN Block Data value supplied in field 11.

E_{KPE}`(New PIN Block)`

Field 10, the new encrypted PIN. This field contains a 16 or 18 byte hexadecimal value.

`PIN Block Data`

Field 11, PIN block data. The content and number of fields depend on the PIN block type. See PIN Block Types.

Table 4-63 Command 37: PIN Change – VISA

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	37
1	PIN verification method (VISA)	1	3
2	PIN block type	1	1 - 5
3	E_{KPE}(Old PIN Block)	0, 16, 18	0 - 9, A - F
4	Header,$E_{MFK.E}$(KPE),MAC*	74	printable ASCII
5	Header,$E_{MFK.E}$(KPV),MAC*	74	printable ASCII
6	Reserved	0, 1, 2	0 - 9, A - F
7	PVV	4	0 - 9
8	PVKI	1	0 - 9
9	PAN	11	0 - 9
10	E_{KPE}(New PIN Block)	16, 18	0 - 9, A - F
11	PIN block data**	Variable	

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

```
47
```

Field 0, the response identifier.

```
Sanity Check Indicator/Verification Flag
```

Field 1, the sanity check indicator and verification flag. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	Old PIN verified successfully.
N	Old PIN failed to verify.
LO	Old PIN length error. See option A1.
NO	No Old PIN. See option 66.

Value	Description
SO	Old PIN sanity error. See PIN Sanity Error.
LN	New PIN length error. See option A1.
SN	New PIN sanity error. See PIN Sanity Error.

`[PVV#]`

Field 2, the PIN Verification Value associated with the new PIN if the operation completed successfully. This field will be present only when field 1 contains either “Y” or “NO”.

`[IBM 3624 Sequence Number#]`

Field 3, the IBM 3624 sequence number. This field is returned only if the PIN block type is IBM 3624. When present, this field contains 2 hexadecimal characters.

Table 4-64 Response 47: PIN Change – VISA

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	2	47
1	Sanity Check Indicator	1, 2	Y, N, SO, SN, LO, LN, or NO
2	[PVV#]	0, 4	0 - 9
3	IBM 3624 Sequence Number*	2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

- The new and old PIN blocks used in the command must always be the same PIN block type, and encrypted using the same KPE.
- Both keys in the Visa key pair must be in the same Atalla Key Block.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify a PIN and Generating a PVV using the Visa method

- Verification method: VISA (3)
- PIN block type: PIN/pad character (3)

- Clear-text old PIN Pad PIN block: 1234 FFFF FFFF FFFF
  The old PIN Pad PIN block encrypted under the PIN Encryption Key: EA40 9665 44AB 4654.

- Clear-text PIN Encryption Key: 1111 1111 1111 1111
  The PIN Encryption Key in AKB format:
  1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC785574,003D0F146BFFC1F9

- Clear-text Key pair: 3333 3333 3333 3333 4444 4444 4444 4444
  The Key pair in AKB format:
  1VVNE000,D6CC0AF3A106D3835D66586A2D6BB4E23357AC74510F2A30,AA35B8BB5168D3DB

- Clear-text new PIN Pad PIN block: 4321 FFFF FFFF FFFF
  The new PIN Pad PIN block encrypted under the PIN Encryption Key: B296 DB18 36A3 F011

The command looks like this:

```
<37#3#3#EA40966544AB4654#1PUNE000,302A81872F21CE1D36DCF0204EEB0FD
CF679984BFC785574,003D0F146BFFC1F9#1VVNE000,D6CC0AF3A106D3835D665
86A2D6BB4E23357AC74510F2A30,AA35B8BB5168D3DB##9015#1#12345612345#
B296DB1836A3F011#F#123456123456#>
```

The Network Security Processor returns the following response:

```
<47#Y#8449#>
```

Generate a PVV without an old PIN

The command looks like this:

```
<37#3#3##1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC78557
4,003D0F146BFFC1F9#1VVNE000,D6CC0AF3A106D3835D66586A2D6BB4E23357A
C74510F2A30,AA35B8BB5168D3DB##0000#1#12345612345#B296DB1836A3F011
#F#123456123456#>
```

The Network Security Processor returns the following response:

```
<47#NO#8449#>
```

PIN Change – Atalla DES Bilevel (Command 37)

Command 37 – Atalla DES Bilevel verifies the old PIN using the Atalla DES Bilevel method. If the old PVN-2 is verified, a PVN2, based on the new PIN, will be generated.

You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

This command has the ability to generate a new PVN2 without verifying the OLD PIN. This functionality has a high security exposure. To enable this functionality you must purchase option 66 in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<37#4#PIN Block Type#EKPE(Old PIN Block)#Header,EMFK.E(KPE),MAC#
Bank ID#Validation Data#Header,EMFK.E(KPV),MAC#PVN-2#PVN-2 Type#
PVN-1 Flag#PVN-2 Start-Compare Flag#EKPE(New PIN Block)#
PIN Block Data#>
```

Response

```
<47#Sanity Check Indicator#[PVN-2#][IBM 3624 Sequence Number#]>
[CRLF]
```

Calling Parameters

`37`

Field 0, the command identifier.

`4`

Field 1, the PVN-2 verification/generation technique; Atalla DES Bilevel.

`PIN Block Type`

Field 2, the PIN block type. This field is 1 byte, it can contain the numbers 1 through 5.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN/pad character / Docutel
4	IBM encrypting PIN pad
5	Burroughs

`E`$_{KPE}$`(Old PIN Block)`

Field 3, the old encrypted PIN.When this field is empty and option 66 is enabled, the PIN verification step is not performed before the new PVN-2 is generated. This field is empty or a 16 or 18 byte hexadecimal value.

`Header,E`$_{MFK.E}$`(KPE),MAC`

Field 4, the PIN Encryption Key encrypted under the MFK. This field must be a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

`Bank ID`

Field 5, the bank ID field for the Identikey card user. The ID is specified by the issuer and can be a 2,6, or 8 byte decimal value.

Data Type	**Allowable Size (bytes)**
Backward index (algorithm number less than 65)	2
ISO number	6
Route and transfer number	8

`Validation Data`

Field 6, validation data. The partial Primary Account Number. This field contains a 4 to 19 byte decimal value.

`Header,E`$_{MFK.E}$`(KPV),MAC`

Field 7, the PIN Verification Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 4A is enabled, this key can be a 1key-3DES (single-length) key. The following headers are supported: 1VBNE000 and 1VBNN000. When field 3 contains an encrypted PIN block, these additional headers are also supported: 1VBVE000 and 1VBVN000. When option 66 is enabled and field 3 is empty, only these headers are supported: 1VBNE000, 1VBNN000, 1VBGE000 and 1VBGN000.

`PVN-2`

Field 8, the PIN Verification Number-2. This field contains a 4 to 16 byte hexadecimal value.

If there is no old PIN to verify, this field should contain a PVN-2 of zeros that is equal in length to the desired PVN-2 length for the new PIN. For example, if the PVN-2 length for the new PIN is 6 digits, this field would contain a six zeros.

`PVN-2 Type`

Field 9, the PVN-2 type. This field indicates whether the PVN-2 should be converted to a decimal value. This field is 1 byte, it contains the numbers 0 or 1. The following table identifies the Value for each type of PVN-2.

Value	Action
0	Convert PVN-2 to a decimal value
1	Don't convert PVN-2; leave it as a hexadecimal value

`PVN-1 Flag`

Field 10, a flag which indicates that 8 digits of the PVN-1 value are used to compare to PVN-2. This field is 1 byte, it contains the number 8.

`PVN-2 Start-Compare Flag`

Field 11, a PVN-2 Start-Compare Flag that specifies the starting position within the generated PVN-2 for the comparison. This field is 1 byte, it contains the number 1.

E_{KPE}`(New PIN Block)`

Field 12, the new encrypted PIN. This field contains a 16 or 18 byte hexadecimal value.

`PIN Block Data`

Field 13, PIN block data. The content and number of fields depend on the PIN block type. See PIN Block Types.

Table 4-65 Command 37: PIN Change – Atalla DES BiLevel

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	37
1	PIN verification method (Atalla DES BiLevel)	1	4
2	PIN block type	1	1 - 5
3	E_{KPE}(Old PIN Block)	0, 16, 18	0 - 9, A - F
4	Header,$E_{MFK.E}$(KPE),MAC*	74	printable ASCII
5	Bank ID	2, 6, 8	0 - 9
6	Validation Data	4 - 19	0 - 9
7	Header,$E_{MFK.E}$(KPV),MAC*	74	printable ASCII
8	PVN-2	4 - 16	0 - 9, A - F
9	PVN-2 Type	1	0, 1

Table 4-65 Command 37: PIN Change – Atalla DES BiLevel (continued)

Field	Contents	Length (bytes)	Legal Characters
10	PVN-1 Flag	1	8
11	PVN-2 Start-Compare Flag	1	1
12	E_{KPE}(New PIN Block)	16, 18	0 - 9, A - F
13	PIN block data**		

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

`47`

Field 0, the response identifier.

`Sanity Check Indicator/Verification Flag`

Field 1, the sanity check indicator and verification flag. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	Old PIN verified successfully.
N	Old PIN failed to verify.
LO	Old PIN length error. See option A1.
NO	No Old PIN. See option 66.
SO	Old PIN sanity error. See PIN Sanity Error.
LN	New PIN length error. See option A1.
SN	New PIN sanity error. See PIN Sanity Error.

`[PVN-2#]`

Field 2, the PVN-2 associated with the new PIN if the operation completed successfully. This field will be present only when field 1 contains either “Y” or “NO”.

`[IBM 3624 Sequence Number#]`

Field 3, the IBM 3624 sequence number. This field is returned only if the PIN block type is IBM 3624. When present, this field contains 2 hexadecimal characters.

Table 4-66 Response 47: PIN Change – Atalla DES BiLevel

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	2	47
1	Sanity Check Indicator	1, 2	Y, N, SO, SN, LO, LN, NO
2	[PVN-2#]	0, 4 - 16	0 - 9, A -F
3	IBM 3624 Sequence Number*	2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

- The design of Command 37 allows the customer to select a new PIN that can be a different length than their old PIN.
- The new and old PIN blocks used in the command must always be the same PIN block type, and encrypted using the same KPE.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify a PIN and Generating a PVN-2 using the Atalla Bilevel method

- Verification method: Atalla DES BiLevel (4)
- PIN block type: IBM Encrypting PIN Pad (4)
- Clear-text old IBM Encrypting PIN Pad PIN block: 4123 4FFF FFFF FF00.
  The old IBM Encrypting PIN Pad PIN block encrypted under the PIN Encryption Key: 214A 1EFD CFFD 0A1C
- Clear-text PIN Encryption Key: 1111 1111 111 1111
  The PIN Encryption Key in AKB format:
  1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC785574,003D0F146BFFC1F9
- Clear-text PIN Verification Key: 3333 3333 3333 3333
  The PIN Verification Key in AKB format:
  1VBNE000,D8E739F7B88B5517141879935CBD87F52C454CB65CBEC862,73514A360A56F0F3

- Clear-text new IBM Encrypting PIN Pad PIN block: 4321 FFFF FFFF FF00
  The new IBM Encrypting PIN Pad PIN block encrypted under the PIN Encryption Key: 0A94 856C 8E80 DF5C

The command looks like this:

```
<37#4#4#214A1EFDCFFD0A1C#1PUNE000,302A81872F21CE1D36DCF0204EEB0FD
CF679984BFC785574,003D0F146BFFC1F9#26#123456123456#1VBNE000,D8E73
9F7B88B5517141879935CBD87F52C454CB65CBEC862,73514A360A56F0F3#35D9
6902C6D972C0#1#8#1#0A94856C8E80DF5C#123456123456#>
```

The Network Security Processor returns the following response:

```
<47#Y#990BE68EF7ECAB92#>
```

Generate a PVN-2 without verifying the old PIN

The command looks like this:

```
<37#4#4##1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC78557
4,003D0F146BFFC1F9#26#123456123456#1VBNE000,D8E739F7B88B551714187
9935CBD87F52C454CB65CBEC862,73514A360A56F0F3#0000000000000000#1#8
#1#0A94856C8E80DF5C#123456123456#>
```

The Network Security Processor returns the following response:

```
<47#NO#990BE68EF7ECAB92#>
```

PIN Change – Diebold (Command 37)

Command 37 – Diebold verifies the old PIN using Diebold method. If the old offset is verified, an offset, based on the new PIN, will be generated.

You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

This command has the ability to generate a new offset without verifying the OLD PIN. This functionality has a high security exposure. You must purchase option 66 in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<37#5#PIN Block Type#EKPE(Old PIN Block)#Header,EMFK.E(KPE),MAC#
Validation Data#Offset#Algorithm Number#
Diebold Number Table Location#EKPE(New PIN Block)#PIN Block Data#>
```

Response

```
<47#Sanity Check Indicator#[Offset#][IBM 3624 Sequence Number#]>
[CRLF]
```

Calling Parameters

`37`

Field 0, the command identifier.

`5`

Field 1, the offset verification/generation technique; Diebold.

`PIN Block Type`

Field 2, the PIN block type. This field is 1 byte, it can contain the numbers 1 through 5.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN/pad character / Docutel
4	IBM encrypting PIN pad
5	Burroughs

`E`$_{KPE}$`(Old PIN Block)`

Field 3, the old encrypted PIN. When this field is empty and option 66 is enabled, the PIN verification step is not performed before the new offset is generated. This field is empty or a 16 or 18 byte hexadecimal value.

`Header,E`$_{MFK.E}$`(KPE),MAC`

Field 4, the PIN Encryption Key encrypted under the MFK. This field must be a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

`Validation Data`

Field 5, validation data. The Primary Account Number (PAN). This field contains a 4 to 19 byte decimal value.

`Offset`

Field 6, an offset value applied to the algorithm-generated PIN before comparing it with the customer entered PIN. This field contains a 4 byte decimal value. If there is no old PIN to verify, this field should contain an offset of four zeros.

`Algorithm Number`

Field 7, the Diebold algorithm number. This field contains a 2 byte decimal value.

`Diebold Number Table Location`

Field 8, the index to the first volatile table location where the Diebold Number Table is stored. This field contains a 1 to 4 byte decimal value.

`E`$_{KPE}$`[New PIN Block]`

Field 9, the new encrypted PIN. This field contains a 16 or 18 byte hexadecimal value.

`PIN Block Data`

Field 10, PIN block data. The content and number of fields depend on the PIN block type. See PIN Block Types.

Table 4-67 Command 37: PIN Change – Diebold

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	37
1	PIN verification method (Diebold)	1	5
2	PIN block type	1	1 - 5
3	E$_{KPE}$(Old PIN Block)	0, 16, 18	0 - 9, A - F
4	Header,E$_{MFK.E}$(KPE),MAC*	74	printable ASCII

Table 4-67 Command 37: PIN Change – Diebold (continued)

Field	Contents	Length (bytes)	Legal Characters
5	Validation data	4 - 19	0 - 9
6	Offset	4	0 - 9
7	Algorithm Number	2	0 - 9
8	Diebold Number Table Location	1 - 4	0 - 9
9	E_{KPE}(New PIN Block)	16, 18	0 - 9, A - F
10	PIN block data**		

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

`47`

Field 0, the response identifier.

`Sanity Check Indicator/Verification Flag`

Field 1, the sanity check indicator and verification flag. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	Old PIN verified successfully.
N	Old PIN failed to verify.
LO	Old PIN length error. See option A1.
NO	No Old PIN. See option 66.
SO	Old PIN sanity error. See PIN Sanity Error.
LN	New PIN length error. See option A1.
SN	New PIN sanity error. See PIN Sanity Error.

`[Offset#]`

Field 2, the Offset associated with the new PIN if the operation completed successfully. This field will be present only when field 1 contains either “Y” or “NO”.

`[IBM 3624 Sequence Number#]`

Field 3, the IBM 3624 sequence number. This field is returned only if the PIN block type is

IBM 3624. When present, this field contains 2 hexadecimal characters.

Table 4-68 Response 47: PIN Change – Diebold

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	2	47
1	Sanity Check Indicator	1,2, 20	Y, N, SO, SN, LO, LN, NO, INVALID NUMBER TABLE.
2	[Offset#]	0, 4 - 12	0 - 9
3	IBM 3624 Sequence Number*	2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

- The design of Command 37 allows the customer to select a new PIN that can be a different length than their old PIN.
- The new and old PIN blocks used in the command must always be the same PIN block type, and encrypted using the same PIN Encryption Key.
- Load the Diebold Number Table using command 74.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify a PIN and Generating a offset using the Diebold method

The Diebold Number Table must be loaded using command 74 prior to executing this PIN verification command.

- Verification method: Diebold (5)
- PIN block type: Burroughs (5)
- Clear-text old PIN block: 3132 3334 FFFF FFFF
  The old PIN block encrypted under the PIN Encryption Key: 8814 2C26 5175 6E94
- Clear-text PIN Encryption Key: 1111 1111 1111 1111
  The PIN Encryption Key in AKB format:
  1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC785574,003D0F146
  BFFC1F9

- Clear-text new PIN block: 3433 3231 FFFF FFFF
  The new PIN block encrypted under the PIN Encryption Key: 190A 2878 81D7 1524

The command looks like this:

```
<37#5#5#88142C2651756E94#1PUNE000,302A81872F21CE1D36DCF0204EEB0FD
CF679984BFC785574,003D0F146BFFC1F9#1234567890#5222#
82#250#190A287881D71524#F#123456123456#>
```

The Network Security Processor returns the following response:

```
<47#Y#2135#>
```

Generate a new offset without verifying the old PIN

The command looks like this:

```
<37#5#5##1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC78557
4,003D0F146BFFC1F9#1234567890#0000#82#250#
190A287881D71524#F#123456123456#>
```

The Network Security Processor returns the following response:

```
<47#NO#2135#>
```

PIN Change – NCR (Command 37)

Command 37 – NCR verifies the old PIN using the NCR method. If the old offset is verified, an offset, based on the new PIN, will be generated.

You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

This command has the ability to generate a new offset without verifying the OLD PIN. This functionality has a high security exposure. You must purchase option 66 in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<37#6#PIN Block Type#EKPE(Old PIN Block)#Header,EMFK.E(KPE),MAC#
Conversion Table#Offset#Validation Data#Pad#PLEN#
Header,EMFK.E(KPV),MAC#Padding Flag#Counting Flag#
Start-Count Position#Select-PLEN Position#EKPE(New PIN Block)#
PIN Block Data#>
```

Response

```
<47#Sanity Check Indicator#[Offset#][IBM 3624 Sequence Number#]>
[CRLF]
```

Calling Parameters

`37`

Field 0, the command identifier.

`6`

Field 1, the offset verification/generation technique NCR.

`PIN Block Type`

Field 2, the PIN block type. This field is 1 byte, it can contain the numbers 1 through 5.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN/pad character / Docutel
4	IBM encrypting PIN pad
5	Burroughs

`E`$_{KPE}$`(Old PIN Block)`

Field 3, the old encrypted PIN. When this field is empty and option 66 is enabled, the PIN verification step is not performed before the new offset is generated. This field is empty or a 16 or 18 byte hexadecimal value.

`Header,E`$_{MFK.E}$`(KPE),MAC`

Field 4, the PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

`Conversion Table`

Field 5, a table that maps hexadecimal digits (0 through 9, A through F) to decimal digits (0 through 9). This field contains a 16 byte decimal value containing the clear-text Conversion Table or a volatile table location. When option 48 is enabled, this field contains the conversion table in AKB format (the header must be 1nCNE000) or a volatile table location. When option 4E is enabled, all three forms of the conversion table (clear-text, decrypted, or value stored in volatile table location) must adhere to these rules:

- The conversion table must have at least eight unique digits.
- No single digit can occur more than four times.

`Offset`

Field 6, an offset value applied to the algorithm-generated PIN before comparing it with the customer-entered PIN. This field contains a 4 to 16 byte decimal value.

`Validation Data`

Field 7, validation data. This value is unique for each card holder and is typically the account number. This field contains a 4 to 16 byte hexadecimal value. When the PIN block type is ANSI (field 1 = 1) and option 4C is enabled, the value supplied in this field must be 12 digits in length and equal to the PIN Block Data value supplied in field 16.

`Pad`

Field 8, a pad character used to fill out the partial PAN. This field contains a one byte hexadecimal value.

`PLEN`

Field 9, the number of contiguous PIN digits selected for verification; the PIN length, or PLEN. This field is 1 byte, it contains a hexadecimal value in the range of 4 through C.

`Header,E`$_{MFK.E}$`(KPV),MAC`

Field 10, the PIN Verification Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 4A is enabled, this key can be a 1key -3DES (single-length) key. The following headers are supported: 1VNNE000 and 1VNNN000. When field 3 contains an encrypted PIN block, these additional headers are also supported: 1VNVE000 and 1VNVN000. When option 66 is enabled and field 3 is empty, only these headers are supported: 1VNNE000, 1VNNN000, 1VNGE000 and 1VNGN000.

`Padding Flag`

Field 11, a flag that indicates whether the validation data (Field 7) is to be padded on the left or right. This field is 1 byte, it contains the character L or R.

`Counting Flag`

Field 12, a flag that indicates whether the counting scheme for selecting the PIN digit for verification is left or right. This field is 1 byte, it contains the character L or R.

`Start-Count Position`

Field 13, the field that indicates the starting position for the counting scheme measured from either the left or right of the entered PIN depending on field 12. This field is 1 byte, it can contain the character 1 through 9.

`Select-PLEN Position`

Field 14, the field that indicates the beginning position (from the left or right, depending upon the counting flag, starting with 0) for selecting PLEN characters from the output of the DES encryption step. This field is 1 byte, it can contain the character 0 through 9, A through C.

E_{KPE}(New PIN Block)

Field 15, the encrypted new PIN block. This field contains a 16 or 18 byte hexadecimal value.

`PIN Block Data`

Field 16, PIN block data. The content and number of fields depend on the PIN block type. See PIN Block Types.

Table 4-69 Command 37: PIN Change – NCR

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	37
1	PIN verification method (NCR)	1	6
2	PIN block type	1	1 - 5
3	E_{KPE}(Old PIN Block)	16, 18	0 - 9, A - F
4	Header,$E_{MFK.E}$(KPE),MAC*	74	printable ASCII

Table 4-69 Command 37: PIN Change – NCR (continued)

Field	Contents	Length (bytes)	Legal Characters
5	Conversion table*	16	0 - 9
6	Offset	4 - 16	0 - 9
7	Validation data	4 - 16	0 - 9
8	Pad	1	0 - 9, A - F
9	PLEN	1	4 - 9, A - C
10	Header,$E_{MFK.E}$(KPV),MAC*	74	printable ASCII
11	Padding Flag	1	L, R
12	Counting Flag	1	L, R
13	Start-Count Position	1	1 - 9
14	Select-PLEN Position	1	0 - 9, A -C
15	E_{KPE}(New PIN Block)	16, 18	0 - 9, A - F
16	PIN block data**		

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

`47`

Field 0, the response identifier.

`Sanity Check Indicator/Verification Flag`

Field 1, the sanity check indicator and verification flag. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	Old PIN verified successfully.
N	Old PIN failed to verify.
LO	Old PIN length error. See option A1.
NO	No Old PIN. See option 66.

Value	Description
SO	Old PIN sanity error. See PIN Sanity Error.
LN	New PIN length error. See option A1.
SN	New PIN sanity error. See PIN Sanity Error.

```
[Offset#]
```

Field 2, the Offset associated with the new PIN if the operation completed successfully. This field will be present only when field 1 contains either “Y” or “NO”.

```
[IBM 3624 Sequence Number#]
```

Field 3, the IBM 3624 sequence number. This field is returned only if the PIN block type is IBM 3624. When present, this field contains 2 hexadecimal characters.

Table 4-70 Response 47: PIN Change – NCR

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	2	47
1	Sanity Check Indicator	1,2	Y, N, SO, SN, LO, LN, NV
2	[Offset#]	0, 4 - 12	0 - 9
3	IBM 3624 Sequence Number*	2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

- The new and old PINs must be the same length.
- The new and old PIN blocks used in the command must always be the same PIN block type, and encrypted using the same PIN Encryption Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify a PIN and Generating a offset using the NCR method

- Verification method: NCR (6)
- PIN block type: ANSI (1)

- Clear-text old PIN block: 0412 26CB A9ED CBA9
  The old PIN, ANSI PIN block encrypted under the PIN Encryption Key: C84F 6825 74BB AA20.

- Clear-text PIN Encryption Key: 1111 1111 1111 1111
  The PIN Encryption Key in AKB format:
  1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC785574,003D0F146BFFC1F9

- Clear-text PIN Verification Key: 68BA 0794 F140 641C
  The PIN Verification Key in AKB format:
  1VNNE000,71013C7998BA9F5D6C4134C2B782851138021646F1C0065B,61FCCC1CA36B188E

- Clear-text new PIN block: 0443 33CB A9ED CBA9
  The new PIN block encrypted under the PIN Encryption Key: 090E 8CA3 CF5D 2AD8.

The command looks like this:

```
<37#6#1#C84F682574BBAA20#1PUNE000,302A81872F21CE1D36DCF0204EEB0FD
CF679984BFC785574,003D0F146BFFC1F9#0123456789012345#0919#27004552
40000121#F#4#1VNNE000,71013C7998BA9F5D6C4134C2B782851138021646F1C
0065B,61FCCC1CA36B188E#R#L#1#6#090E8CA3CF5D2AD8#123456123456#>
```

The Network Security Processor returns the following response:

```
<47#Y#3006#>
```

Generate a new offset without verifying the old PIN

The command looks like this:

```
<37#6#1##1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC78557
4,003D0F146BFFC1F9#0123456789012345#0000#2700455240000121#F#4#1VN
NE000,71013C7998BA9F5D6C4134C2B782851138021646F1C0065B,61FCCC1CA3
6B188E#R#L#1#6#090E8CA3CF5D2AD8#123456123456#>
```

The Network Security Processor returns the following response:

```
<47#NO#3006#>
```

Translate PIN And Generate MAC (Command 39)

Command 39 – translates an encrypted PIN from encryption under one key to encryption under another and generates a Message Authentication Code (MAC) from data contained in the command. The incoming PIN Encryption Key is designated as KPE_I, and the outgoing PIN Encryption Key is designated as KPE_O.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

Command

```
<39#PIN Block Type#Header,EMFK.E(KPEI),MAC#Header,EMFK.E(KPEO),MAC#
EKPEI(PIN Block)#PIN Block Data#Header,EMFK.E(KMAC),MAC#Flag#Data#>
```

Response

```
<49#EKPEO(ANSI PIN block)#Sanity Check Indicator#
[IBM 3624 Sequence Number#]MAC#KMAC Check Digits#>[CRLF]
```

Calling Parameters

`39`

Field 0, the command identifier.

`PIN Block Type`

Field 1, incoming PIN block type. This field is 1 byte, it can contain the numbers 1 through 5. When option 46 is enabled, this field can only contain the value 1 (ANSI).

.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN/pad character / Docutel
4	IBM encrypting PIN pad
5	Burroughs

`Header,EMFK.E(KPEI),MAC`

Field 2, the incoming PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

`Header,E`$_{MFK.E}$`(KPE`$_O$`),MAC`

Field 3, the outgoing PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, the length of the KPEo must be equal to or greater than the length of the KPEi (field 2). The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000.

`E`$_{KPEI}$`(Encrypted PIN Block)`

Field 4, the incoming encrypted PIN encrypted under the PIN Encryption Key. This field contains a 16 or 18 byte hexadecimal value.

`PIN Block Data`

Field 5, PIN block data. Its contents depend on the PIN block type used. See PIN Block Types for information on PIN block data.

`Header,E`$_{MFK.E}$`(KMAC),MAC`

Field 6, the Message Authentication Code key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1MDNE000, 1MDNN000, 1MDGE000 and 1MDGN000.

`Flag`

Field 7, a flag. If you will be including the translated PIN block in the MAC generation, set this field to 1, otherwise, set this field to 0. If the flag is set to 1, the translated PIN block will precede the data to be MACed.

`Data`

Field 8, the data to be authenticated. This field can be up to 239 bytes long and can contain the numbers 0 through 9 and the characters A through Z, as well as commas, periods, and blanks.

Table 4-71 Command 39: Translate PIN and Generate MAC

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	2	39
1	PIN block type	1	1 - 5
2	Header, E$_{MFK.E}$(KPE$_I$),MAC*	74	printable ASCII
3	Header, E$_{MFK.E}$(KPE$_O$),MAC*	74	printable ASCII
4	E$_{KPEI}$(Encrypted PIN block)	16, 18	0 - 9, A - F

Table 4-71 Command 39: Translate PIN and Generate MAC (continued)

Field	Contents	Length (bytes)	Legal Characters
5	PIN block data**		
6	Header, $E_{MFK.E}$(KMAC),MAC*	74	printable ASCII
7	Flag	1	0, 1
8	Data	1 - 239	0 - 9, A - Z “,”, “.”, “ ”

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

`49`

Field 0, the response identifier.

E_{KPEO}`(ANSI PIN Block)`

Field 1, the encrypted outgoing PIN. This field contains 16 hexadecimal characters. When a PIN sanity error is detected, the value in this field may not be correct. When a PIN sanity error is detected, and option 4B is enabled, this field will contain 16 zeros.

`Sanity Check Indicator`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

`[IBM 3624 Sequence Number #]`

Field 3, the IBM 3624 sequence number. This field is returned only if the PIN block type is IBM 3624. When present, this field contains 2 hexadecimal characters.

`MAC`

Field 4, the Message Authentication Code. This field contains an 8 byte hexadecimal value. This field is empty when the PIN block fails the sanity check.

`KMAC Check Digits`

Field 5, check digits; the first four digits that result of encrypting zeros using the Message Authentication Code key. If option 88 is enabled, this field will contain six check digits. This field is empty when the PIN block fails the sanity check.

Table 4-72 Response 49: Translate PIN and Generate MAC

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	2	49
1	E_{KPEO}(ANSI PIN block)	16	0 - 9, A - F
2	Sanity check indicator	1	Y, N, L
3*	IBM 3624 Sequence Number	2	0 - 9, A - F
4	MAC	8	0 - 9, A - F
5	KMAC Check Digits	4 or 6	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

Perform the following tasks before using Command 39:

- Generate the incoming and outgoing PIN Encryption Keys.
- Generate the Message Authentication Code key.
- If the incoming PIN block is in an IBM 3624 PIN block, generate the ATM Communications Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate an ANSI formatted PIN and generating Message Authentication Code

- PIN block type: ANSI (1)
- Clear-text incoming PIN Encryption Key: 0000 1111 2222 3333
  The incoming PIN Encryption Key in AKB format:
  1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D60C5A5CA83A4D1258,
  0DBB2D4988A072F2

- Clear-text outgoing PIN Encryption Key: 1111 2222 3333 4444
  The outgoing PIN Encryption Key in AKB format:
  1PUNE000,1AB5E6DC613047FD2672333FD0E02A1828ED8A4C449A5299,
  4421D569680BFC5C

- Clear-text incoming PIN block: 0C12 3456 7890 12FF
  The incoming PIN block encrypted under the PIN Encryption Key:
  4476 A5ED F270 3FF8

- PIN block data; in this case, the 12 digits of the Primary Account Number:
  7788 9900 0000

- Clear-text Message Authentication Code key: FEDC BA98 7654 3210
  The Message Authentication Code key in AKB format:
  1MDNE000,EEC942756FEE2710A60A4410F2A12215153C05BC7A4A2B9B,
  96A475A137989605

- Flag: 0

- Data to be authenticated: ABCD 1234 ABCD 1234

The command looks like this:

```
<39#1#1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D60C5A5CA83A4D1258,0D
BB2D4988A072F2#1PUNE000,1AB5E6DC613047FD2672333FD0E02A1828ED8A4C44
9A5299,4421D569680BFC5C#4476A5EDF2703FF8#778899000000#1MDNE000,EEC
942756FEE2710A60A4410F2A12215153C05BC7A4A2B9B,96A475A137989605#0#A
BCD1234ABCD1234#>
```

The Network Security Processor returns the following response:

```
<49#1371A72D914FDE41#Y#68AE2DD2#A68C#>
```

Verify Card and PIN - IBM3624 (Command 3A)

Command 3A is a combination command that performs both card and PIN verification. Three card verification algorithms are supported: VISA Card Verification Value (CVV), MasterCard Card Validation Code (CVC), and American Express Card Security Code (CSC). If the card verification is successful, the command then attempts to verify a customer PIN using the IBM3624 PIN verification algorithm. The ANSI PIN block to be verified can be DES or 3DES encrypted under a master/session PIN encryption key or a Derived Unique Key Per Transaction (DUKPT) key.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<3A#PIN Algorithm,Card Algorithm#CVV/CVC/CSC Data#CVV/CVC/CSC#
Header,EMFK.E(KCVV/KCVC/KCSC),MAC#EKPE(ANSI PIN Block)#
Header,EMFK.E(KPE/DK),MAC#Conversion Table#Offset#Validation Data#
Pad#Check-Length#Header,EMFK.E(KPV),MAC#PIN Block Data#>
```

Response

```
<4A#Card Verification Indicator#PIN Verification/Sanity Indicator#
KCVV/KCVC/KCSC check digits#>[CRLF]
```

Calling Parameters

`3A`

Field 0, the command identifier.

`PIN Algorithm,Card Algorithm`

Field 1, the PIN algorithm value followed by a comma "," followed by the card verification algorithm. The PIN algorithm must be set to "2" for IBM3624. The card verification algorithms are:

Value	Card Algorithm
2	VISA Card Verification Value (CVV)
2	MasterCard Card Validation Code (CVC)
3	American Express Card Security Code (CSC) Version 1.0

For example, IBM3624 PIN verification with AMEX CSC card verification is 2,3.

`CVV/CVC/CSC Data`

Field 2, the data required to verify the CVV/CVC/CSC. For CVV/CVC card verification this field contains a 1 to 32 digit value, which should include the PAN followed by the expiration date in YYMM format, followed by the service code.

```
PAN|Expiration Date|Service Code
```

For CSC card verification, this field must contain a 19 digit value in the form of a 15 digit PAN followed by a 4 digit expiration date in YYMM format. The leftmost two PAN digits must be 34 or 37.

```
PAN|Expiration Date
```

`CVV/CVC/CSC`

Field 3, the CVV, CVC, or CSC(s) to be verified.

The CVV/CVC can be a 1 to 8 digit value. When option 4D is enabled, this field must contain a 3 to 8 digit CVV/CVC value.

There are three lengths of CSC values that can be verified: a 5-digit CSC, a 4-digit CSC, and a 3-digit CSC. To verify multiple CSC values they must be specified in decreasing length order and be separated by a comma. Only one value per CSC length can be specified per command.

Header,$E_{MFK.E}$(KCVV/KCVC/KCSC),MAC

Field 4, for CVV or CVC card verification this field must contain the 2key-3DES (double-length) key pair encrypted under the MFK. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. This field contains a 74 byte value. The following headers are supported: 1CDNE000, 1CDNN000,1CDVE000, and 1CDVN000.

For CSC card verification this field must contain the 2key-3DES (double-length) key pair encrypted under the MFK. This field contains a 74 byte value. The following headers are supported: 1mXNE000, 1mXNN000, 1mXVE000, and 1mXVN000.

E_{KPE}(ANSI PIN Block)

Field 5, the DES or 3DES encrypted ANSI PIN block to be verified. This field must contain a 16 hexadecimal character value.

Header,$E_{MFK.E}$(KPE/DK),MAC

Field 6, the PIN Encryption Key or Base Derivation Key encrypted under the MFK. This field contains a 74 byte value. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The AKB header for the PIN encryption key (KPE) must be one of the following values: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000. The VISA DUKPT Base Derivation Key (DK) AKB header must be one of the following values:1dDNE000 and 1dDNN000.

`Conversion Table`

Field 7, a table that maps hexadecimal digits (0 through 9, A through F) to decimal digits (0 through 9). This field contains a 16 byte decimal value containing the clear-text Conversion Table or a volatile table location. When option 48 is enabled, this field contains the conversion table in AKB format (the header must be 1nCNE000) or a volatile table location. When option 4E is enabled, all three forms of the conversion table (clear-text, decrypted, or value stored in volatile table location) must adhere to these rules:

- The conversion table must have at least eight unique digits.
- No single digit can occur more than four times.

`Offset`

Field 8, an offset value applied to the algorithm-generated PIN before comparing it with the customer-entered PIN. This field contains a 4 to 12 byte decimal value.

`Validation Data`

Field 9, validation data. This value is unique for each card holder, and is typically the account number. This field contains a 4 to 16 byte decimal value. When option 4C is enabled, the value supplied in this field must be 12 digits in length and equal to the PIN Block Data value supplied in field 13.

`Pad`

Field 10, the pad character which right-pads the validation data. This field contains a one byte hexadecimal value. The pad character is only applied when the validation data is less than 16 digits.

`Check-Length`

Field 11, the check-length. This value is typically the PIN length and determines the number of PIN digits to be compared. This field contains one hexadecimal character in the range of 4 through C.

Header, $E_{MFK.E}$(KPV), MAC

Field 12, the PIN Verification Key encrypted under the MFK. This field contains a 74 byte value. When option 4A is enabled, this key can be a 1key -3DES (single-length) key. The following headers are supported: 1V3NE000, 1V3NN000, 1V3VE000, and 1V3VN000.

`PIN Block Data`

Field 13, PIN block data. When a master/session KPE encrypts the ANSI PIN block, this field must contain the rightmost 12 PAN digits excluding the check digit.

When a DUKPT key encrypts the ANSI PIN block, three fields are present. See VISA Derived Unique Key Per Transaction PIN Block for more information on the contents of these three fields.

Table 4-73 Command 3A: Verify Card and PIN - IBM3624

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	3A
1	PIN Algorithm,Card Algorithm	3	2, 3, and “,”
2	CVV/CVC/CSC Data	1 - 32 or 19	0 - 9
3	CVV/CVC/CSC	1-8, varies	0 - 9 and “,”
4	Header,$E_{MFK.E}$(KCVV/KCVC/KCSC),MAC	74	printable ASCII
5	E_{KPE}(ANSI PIN Block)	16	0 - 9, A - F
6	Header,$E_{MFK.E}$(KPE/DK),MAC	74	printable ASCII
7	Conversion Table	16	0 - 9
8	Offset	4 - 16	0 - 9
9	Validation data	4 - 16	0 - 9
10	Pad	1	0 - 9, A - F
11	Check Length	1	4 - 9, A - C
12	Header,$E_{MFK.E}$(KPV),MAC	74	printable ASCII
13	PIN block data KPE DUKPT**	12 **	0 - 9 **

** See VISA Derived Unique Key Per Transaction PIN Block for information on PIN block data.

Responding Parameters

`4A`

Field 0, the response identifier.

`Card Verification Indicator`

Field 1, the card verification indicator. The value will be Y if the card verified, or N if the card did not verify.

If multiple CSCs were present in field 3 of the command, this field will contain either a Y or N, separated by a comma, for each CSC.

`PIN Verification/Sanity Indicator`

Field 2, the PIN verification and sanity indicator. A sanity check is performed on the decrypted ANSI PIN block. If the sanity check is successful, the PIN verification step is performed. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN verification was successful.
N	PIN verification failed.
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.
C	Card verification failed.

`KCVV/KCVC/KCSC check digits`

Field 3, check digits; the first four hexadecimal digits that result from encrypting zeros using the KCVV/KCVC/KCSC. If option 88 is enabled, this field will contain six check digits.

Table 4-74 Response 4A: Verify Card and PIN - IBM3624

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	4A
1	Card Verification Indicator	varies	Y, N, “,”
2	PIN Verification /Sanity Check Indicator	1	Y, N, S, L, C
3	KCVV/KCVC/KCSC check digits	4, 6	0 - 9, A - F

Usage Notes

- Generate the PIN Encryption, Card Verification, and PIN Verification keys.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

CVV card verification and 3DES master/session PIN verification

- Clear-text KCVV: 0123456789ABCDEF FEDCBA9876543210
  The KCVV in AKB format:
  1CDNE000,4F7EFE3F44984427CF46B823CE4BDE1839E35E6F46EB2814,
  26A5545D383FA701
- PAN: 1234567890123456
- Expiration Date: 0806 (June 2008)
- Service Code: 101
- CVV: 921
- PIN: 1234
- ANSI PIN Block: 0412719876FEDCBA
- Clear-text PIN Encryption Key: 1111111111111111 2222222222222222
  The PIN Encryption Key in AKB format:
  1PUNE000,302A81872F21CE1D0A8791AC9B0C7A6ABB5A9D73F6EB4872,
  B8E0CBEEB718723D
- Encrypted ANSI PIN Block: 6EF2480ACCF9FC50
- Conversion Table: 0123456789012345
- Offset: 9926
- Validation data:1234567890123456
- Pad for validation data: F
- Check-Length: 4
- Clear-text PIN Verification Key: 3333 3333 3333 3333.
  The PIN Verification Key in AKB format:
  1V3NE000,F0BD2DA1B14F54B16E5541B6F61C35AA4F56EDEFFB7ADDD3,
  450CC0232392B2F5
- The PIN block data: 456789012345

The command looks like this:

```
<3A#2,2#12345678901234560806101#921#1CDNE000,4F7EFE3F44984427CF46
B823CE4BDE1839E35E6F46EB2814,26A5545D383FA701#6EF2480ACCF9FC50#1P
UNE000,302A81872F21CE1D0A8791AC9B0C7A6ABB5A9D73F6EB4872,B8E0CBEEB
718723D#0123456789012345#9926#1234567890123456#F#4#1V3NE000,F0BD2
DA1B14F54B16E5541B6F61C35AA4F56EDEFFB7ADDD3,450CC0232392B2F5#4567
89012345#>
```

The Network Security Processor returns the following response:

```
<4A#Y#Y#08D7#>
```

CSC card verification and DES master/session PIN verification

- Clear-text KCSC: 0123456789ABCDEF FEDCBA9876543210
  The KCSC in AKB format:
  1mXNE000,823244EC4A4DFA0198097F067749BE0C9448676EB51ABAE2,
  459E9F9EE15CE583
- PAN: 345678901234567
- Expiration Date: 0806 (June 2008)
- CSCs: 25199,0365,187
- PIN: 1234
- ANSI PIN Block: 041262876FEDCBA9
- Clear-text PIN Encryption Key: 1111 1111 1111 1111
  The PIN Encryption Key in AKB format:
  1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC785574,
  003D0F146BFFC1F9
- Encrypted ANSI PIN Block: F2106C85AF495833
- Conversion Table: 0123456789012345
- Offset: 4204
- Validation data: 345678901234567
- Pad for validation data: F
- Check-Length: 4
- Clear-text PIN Verification Key: 3333 3333 3333 3333
  The PIN Verification Key in AKB format:
  1V3NE000,F0BD2DA1B14F54B16E5541B6F61C35AA4F56EDEFFB7ADDD3,
  450CC0232392B2F5
- The PIN block data: 567890123456

The command looks like this:

```
<3A#2,3#3456789012345670806#25199,0365,187#1mXNE000,823244EC4A4DFA
0198097F067749BE0C9448676EB51ABAE2,459E9F9EE15CE583#F2106C85AF4958
33#1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC785574,003D0
F146BFFC1F9#0123456789012345#4204#345678901234567#F#4#1V3NE000,F0B
D2DA1B14F54B16E5541B6F61C35AA4F56EDEFFB7ADDD3,450CC0232392B2F5#567
890123456#>
```

The Network Security Processor returns the following response:

```
<4A#Y,Y,Y#Y#08D7#>
```

CVC card verification and 3DES DUKPT PIN verification

- Clear-text KCVC: 0123456789ABCDEF FEDCBA9876543210
  The KCVV in AKB format:
  1CDNE000,4F7EFE3F44984427CF46B823CE4BDE1839E35E6F46EB2814,
  26A5545D383FA701
- PAN: 1234567890123456
- Expiration Date: 0806 (June 2008)
- Service Code: 101
- CVV: 921
- PIN: 1234
- ANSI PIN Block: 0412719876FEDCBA
- Clear-text Base Derivation Key: 0123456789ABCDEF FEDCBA9876543210
  The Base Derivation Key in AKB format:
  1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,
  342946FE884AA8B2
- Encrypted ANSI PIN Block: 39510B23F12E4ABD
- Conversion Table: 0123456789012345
- Offset: 9926
- Validation data:1234567890123456
- Pad for validation data: F
- Check-Length: 4
- Clear-text PIN Verification Key: 3333 3333 3333 3333
  The PIN Verification Key in AKB format:
  1V3NE000,F0BD2DA1B14F54B16E5541B6F61C35AA4F56EDEFFB7ADDD3,
  450CC0232392B2F5
- The PIN block data:
  - Rightmost 12 PAN digits excluding the check digit: 456789012345
  - KSN: 9876543210E00008
  - Algorithm: D

The command looks like this:

```
<3A#2,2#12345678901234560806101#921#1CDNE000,4F7EFE3F44984427CF46
B823CE4BDE1839E35E6F46EB2814,26A5545D383FA701#39510B23F12E4ABD#1d
DNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,342946FE8
84AA8B2#0123456789012345#9926#1234567890123456#F#4#1V3NE000,F0BD2
DA1B14F54B16E5541B6F61C35AA4F56EDEFFB7ADDD3,450CC0232392B2F5#4567
89012345#9876543210E00008#D#>
```

The Network Security Processor returns the following response:

```
<4A#Y#Y#08D7#>
```

Verify Card and PIN - VISA (Command 3A)

Command 3A is a combination command that performs both card and PIN verification. Three card verification algorithms are supported: VISA Card Verification Value (CVV), MasterCard Card Validation Code (CVC), and American Express Card Security Code (CSC). If the card verification is successful, the command then attempts to verify a customer PIN using the VISA PIN verification algorithm. The ANSI PIN block to be verified can be DES or 3DES encrypted using either a master/session PIN encryption key or a Derived Unique Key Per Transaction (DUKPT) key.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<3A#PIN Algorithm,Card Algorithm#CVV/CVC/CSC Data#CVV/CVC/CSC#
Header,EMFK.E(KCVV/KCVC/KCSC),MAC#EKPE(ANSI PIN Block)#
Header,EMFK.E(KPE/DK),MAC#Header,EMFK.E(KPV),MAC##PVV#PVKI#PAN#
PIN Block Data#>
```

Response

```
<4A#Card Verification Indicator#PIN Verification/Sanity Indicator#
KCVV/KCVC/KCSC check digits#>[CRLF]
```

Calling Parameters

`3A`

Field 0, the command identifier.

`PIN Algorithm,Card Algorithm`

Field 1, the PIN algorithm value followed by a comma ",” followed by the card verification algorithm. The PIN algorithm must be set to "3" for VISA. The card verification algorithms are:

Value	Card Algorithm
2	VISA Card Verification Value (CVV)
2	MasterCard Card Validation Code (CVC)
3	American Express Card Security Code (CSC) Version 1.0

For example, VISA PIN verification with AMEX CSC card verification is 3,3.

`CVV/CVC/CSC Data`

Field 2, the data required to verify the CVV/CVC/CSC. For CVV/CVC card verification this field contains a 1 to 32 digit value, which should include the PAN followed by the expiration date in YYMM format, followed by the service code.

```
PAN|Expiration Date|Service Code
```

For CSC card verification, this field must contain a 19 digit value in the form of a 15 digit PAN followed by a 4 digit expiration date in YYMM format. The leftmost two PAN digits must be 34 or 37.

```
PAN|Expiration Date
```

`CVV/CVC/CSC`

Field 3, the CVV, CVC, or CSC(s) to be verified.

The CVV/CVC can be a 1 to 8 digit value. When option 4D is enabled, this field must contain a 3 to 8 digit CVV/CVC value.

There are three lengths of CSCs that can be verified, this field can contain 1, 2, or all 3 CSC values, each separated by a comma. When present, the CSC(s) must be 3, 4, or 5 digits in length.

Header,$E_{MFK.E}$(KCVV/KCVC/KCSC),MAC

Field 4, for CVV or CVC card verification this field must contain the 2key-3DES (double-length) key pair encrypted under the MFK. When option 6C is enabled, this field can contain a
1key-3DES (single-length) key. This field contains a 74 byte value. The following headers are supported: 1CDNE000, 1CDNN000,1CDVE000, and 1CDVN000.

For CSC card verification, this field must contain the 2key-3DES (double-length) key pair encrypted under the MFK. This field contains a 74 byte value. The following headers are supported: 1mXNE000, 1mXNN000, 1mXVE000, and 1mXVN000.

E_{KPE}(ANSI PIN Block)

Field 5, the DES or 3DES encrypted ANSI PIN block to be verified. This field must contain a 16 hexadecimal character value.

Header,$E_{MFK.E}$(KPE/DK),MAC

Field 6, the PIN Encryption Key or Base Derivation Key encrypted under the MFK. This field contains a 74 byte value. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The AKB header for the PIN encryption key (KPE) must be one of the following values: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000. For the base Derivation Key (DK), used to process a DUKPT encrypted PIN block, the AKB header must be one of the following values:1dDNE000 and 1dDNN000.

Header,$E_{MFK.E}$(KPV),MAC

Field 7, the PIN Verification Key encrypted under the MFK. This field contains a 74 byte value. This key must be either a 2key- or 3key-3DES key. If option 8C is enabled, this key can be a 1key-3DES (single-length) key. The following headers are supported:

1VVNE000, 1VVNN000, 1VVVE000, and 1VVVN000.

`Reserved`

Field 8, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

`PVV`

Field 9, the PIN Verification Value (PVV) used to compare to the calculated value. This field contains a four digit value.

`PVKI`

Field 10, the PIN Verification Key Indicator (PVKI) used in the algorithm that calculates the PIN verification value. This field contains a decimal value in the range of 0 to 9.

`PAN`

Field 11, the rightmost eleven Primary Account Number digits, excluding the check digit. When option 4C is enabled, this value must be present in the PIN Block Data value supplied in field 12.

`PIN Block Data`

Field 12, PIN block data. When a master/session KPE encrypts the ANSI PIN block, this field must contain the rightmost 12 PAN digits excluding the check digit.

When a DUKPT key encrypts the ANSI PIN block, three fields are present. See VISA Derived Unique Key Per Transaction PIN Block for more information on the contents of these three fields.

Table 4-75 Command 3A: Verify Card and PIN - VISA

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	2	3A
1	PIN Algorithm,Card Algorithm	3	2, 3, and “,”
2	CVV/CVC/CSC Data	1 - 32 or 19	0 - 9
3	CVV/CVC/CSC	1-8, varies	0 - 9 and “,”
4	Header,$E_{MFK.E}$(KCVV/KCVC/KCSC),MAC	74	printable ASCII
5	E_{KPE}(ANSI PIN Block)	16	0 - 9, A - F
6	Header,$E_{MFK.E}$(KPE/DK),MAC	74	printable ASCII
7	Header,$E_{MFK.E}$(KPV),MAC	74	printable ASCII

Table 4-75 Command 3A: Verify Card and PIN - VISA (continued)

Field	Contents	Length (bytes)	Legal Characters
8	Reserved	0	printable ASCII
9	PVV	4	0 - 9
10	PVKI	1	0 - 9
11	PAN	11	0 - 9
12	PIN block data KPE DUKPT**	12**	0 - 9**

** See VISA Derived Unique Key Per Transaction PIN Block for information on PIN block data.

Responding Parameters

`4A`

Field 0, the response identifier.

`Card Verification Indicator`

Field 1, the card verification indicator. The value will be Y if the card verified, or N if the card did not verify.

If multiple CSCs were present in field 3 of the command, this field will contain either a Y or N, separated by a comma, for each CSC.

`PIN Verification/Sanity Indicator`

Field 2, the PIN verification and sanity indicator. A sanity check is performed on the decrypted ANSI PIN block. If the sanity check is successful, the PIN verification step is performed. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN verification was successful.
N	PIN verification failed.
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.
C	Card verification failed.

`KCVV/KCVC/KCSC check digits`

Field 3, check digits; the first four hexadecimal digits that result from encrypting zeros using the KCVV/KCVC/KCSC. If option 88 is enabled, this field will contain six check digits.

Table 4-76 Response 4A: Verify Card and PIN - VISA

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	4A
1	Card Verification Indicator	varies	Y, N, ","
2	PIN Verification /Sanity Check Indicator	1	Y, N, S, L, C
3	KCVV/KCVC/KCSC check digits	4, 6	0 - 9, A - F

Usage Notes

- Generate the PIN Encryption, Card Verification, and PIN Verification keys.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

CVV card verification and 3DES master/session PIN verification

- Clear-text KCVV: 0123456789ABCDEF FEDCBA9876543210
  The KCVV in AKB format:
  1CDNE000,4F7EFE3F44984427CF46B823CE4BDE1839E35E6F46EB2814,
  26A5545D383FA701
- PAN: 1234567890123456
- Expiration Date: 0806 (June 2008)
- Service Code: 101
- CVV: 921
- PIN: 1234
- ANSI PIN Block: 0412719876FEDCBA

- Clear-text PIN Encryption Key: 1111111111111111 2222222222222222
  The PIN Encryption Key in AKB format:
  1PUNE000,302A81872F21CE1D0A8791AC9B0C7A6ABB5A9D73F6EB4872,
  B8E0CBEEB718723D

- Encrypted ANSI PIN Block: 6EF2480ACCF9FC50

- Clear-text PIN Verification Key: 0123456789ABCDEF FEDCBA9876543210
  The PIN Verification Key in AKB format:
  1VVNE000,F78F26AB6F5763645E41E3FC8EC48F0C68F828FB78FB5AF9,
  3FA7A4230E019EA0

- PVV: 7880

- PVKI: 1

- PAN: 56789012345

- The PIN block data: 456789012345

The command looks like this:

```
<3A#3,2#12345678901234560806101#921#1CDNE000,4F7EFE3F44984427CF46B
823CE4BDE1839E35E6F46EB2814,26A5545D383FA701#6EF2480ACCF9FC50#1PUN
E000,302A81872F21CE1D0A8791AC9B0C7A6ABB5A9D73F6EB4872,B8E0CBEEB718
723D#1VVNE000,F78F26AB6F5763645E41E3FC8EC48F0C68F828FB78FB5AF9,3FA
7A4230E019EA0##7880#1#56789012345#456789012345#>
```

The Network Security Processor returns the following response:

```
<4A#Y#Y#08D7#>
```

CSC card verification and DES master/session PIN verification

- Clear-text KCSC: 0123456789ABCDEF FEDCBA9876543210
  The KCSC in AKB format:
  1mXNE000,823244EC4A4DFA0198097F067749BE0C9448676EB51ABAE2,
  459E9F9EE15CE583

- PAN: 345678901234567

- Expiration Date: 0806 (June 2008)

- CSCs: 25199,0365,187

- PIN: 1234

- ANSI PIN Block: 041262876FEDCBA9

- Clear-text PIN Encryption Key: 1111 1111 1111 1111
  The PIN Encryption Key in AKB format:
  1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC785574,
  003D0F146BFFC1F9
- Encrypted ANSI PIN Block: F2106C85AF495833
- Clear-text PIN Verification Key: 0123456789ABCDEF FEDCBA9876543210
  The PIN Verification Key in AKB format:
  1VVNE000,F78F26AB6F5763645E41E3FC8EC48F0C68F828FB78FB5AF9,
  3FA7A4230E019EA0
- PVV: 7880
- PVKI: 1
- PAN: 56789012345
- The PIN block data: 567890123456

The command looks like this:

```
<3A#3,3#3456789012345670806#25199,0365,187#1mXNE000,823244EC4A4DF
A0198097F067749BE0C9448676EB51ABAE2,459E9F9EE15CE583#F2106C85AF49
5833#1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC785574,00
3D0F146BFFC1F9#1VVNE000,F78F26AB6F5763645E41E3FC8EC48F0C68F828FB7
8FB5AF9,3FA7A4230E019EA0##7880#1#56789012345#567890123456#>
```

The Network Security Processor returns the following response:

```
<4A#Y,Y,Y#Y#08D7#>
```

CVC card verification and 3DES DUKPT PIN verification

- Clear-text KCVC: 0123456789ABCDEF FEDCBA9876543210
  The KCVV in AKB format:
  1CDNE000,4F7EFE3F44984427CF46B823CE4BDE1839E35E6F46EB2814,
  26A5545D383FA701
- PAN: 1234567890123456
- Expiration Date: 0806 (June 2008)
- Service Code: 101
- CVV: 921
- PIN: 1234
- ANSI PIN Block: 0412719876FEDCBA

- Clear-text Base Derivation Key: 0123456789ABCDEF FEDCBA9876543210
  The Base Derivation Key in AKB format:
  1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,
  342946FE884AA8B2
- Encrypted ANSI PIN Block: 39510B23F12E4ABD
- Clear-text PIN Verification Key: 0123456789ABCDEF FEDCBA9876543210
  The PIN Verification Key in AKB format:
  1VVNE000,F78F26AB6F5763645E41E3FC8EC48F0C68F828FB78FB5AF9,
  3FA7A4230E019EA0
- PVV: 7880
- PVKI: 1
- PAN: 56789012345
- The PIN block data:
  - Rightmost 12 PAN digits excluding the check digit: 456789012345
  - KSN: 9876543210E00008
  - Algorithm: D

The command looks like this:

```
<3A#3,2#12345678901234560806101#921#1CDNE000,4F7EFE3F44984427CF46B8
23CE4BDE1839E35E6F46EB2814,26A5545D383FA701#39510B23F12E4ABD#1dDNE0
00,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,342946FE884AA8B
2#1VVNE000,F78F26AB6F5763645E41E3FC8EC48F0C68F828FB78FB5AF9,3FA7A42
30E019EA0##7880#1#56789012345#456789012345#9876543210E00008#D#>
```

The Network Security Processor returns the following response:

```
<4A#Y#Y#08D7#>
```

Verify Card and PIN - Atalla Bilevel (Command 3A)

Command 3A is a combination command that performs both card and PIN verification. Three card verification algorithms are supported: VISA Card Verification Value (CVV), MasterCard Card Validation Code (CVC), and American Express Card Security Code (CSC). If the card verification is successful, the command then attempts to verify a customer PIN using the Atalla Bilevel PIN verification algorithm. The ANSI PIN block to be verified can be DES or 3DES encrypted using either a master/session PIN encryption key or a Derived Unique Key Per Transaction (DUKPT) key.

Support for the Atalla Bilevel PIN algorithm has been added in version 1.50.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<3A#PIN Algorithm,Card Algorithm#CVV/CVC/CSC Data#CVV/CVC/CSC#
Header,E_MFK.E(KCVV/KCVC/KCSC),MAC#E_KPE(ANSI PIN Block)#
Header,E_MFK.E(KPE/DK),MAC#Bank ID#Partial PAN#Header,E_MFK.E(KPV),MAC#
PVN-2#PVN-2 Type#PVN-1 Flag#PVN-2 Start Compare Flag#
PIN Block Data#>
```

Response

```
<4A#Card Verification Indicator#PIN Verification/Sanity Indicator#
KCVV/KCVC/KCSC check digits#>[CRLF]
```

Calling Parameters

`3A`

Field 0, the command identifier.

`PIN Algorithm,Card Algorithm`

Field 1, the PIN algorithm value followed by a comma "," followed by the card verification algorithm. The PIN algorithm must be set to "4" for Atalla Bilevel. The card verification algorithms are:

Value	Card Algorithm
2	VISA Card Verification Value (CVV)
2	MasterCard Card Validation Code (CVC)
3	American Express Card Security Code (CSC) Version 1.0

For example, Atalla Bilevel PIN verification with AMEX CSC card verification is 4,3.

`CVV/CVC/CSC Data`

Field 2, the data required to verify the CVV/CVC/CSC. For CVV/CVC card verification this field contains a 1 to 32 digit value, which should include the PAN followed by the expiration date in YYMM format, followed by the service code.

```
PAN|Expiration Date|Service Code
```

For CSC card verification, this field must contain a 19 digit value in the form of a 15 digit PAN followed by a 4 digit expiration date in YYMM format. The leftmost two PAN digits must be 34 or 37.

```
PAN|Expiration Date
```

`CVV/CVC/CSC`

Field 3, the CVV, CVC, or CSC(s) to be verified.

The CVV/CVC can be a 1 to 8 digit value. When option 4D is enabled, this field must contain a 3 to 8 digit CVV/CVC value.

There are three lengths of CSCs that can be verified, this field can contain 1, 2, or all 3 CSC values, each separated by a comma. When present, the CSC(s) must be 3, 4, or 5 digits in length.

Header,$E_{MFK.E}$(KCVV/KCVC/KCSC),MAC

Field 4, for CVV or CVC card verification this field must contain the 2key-3DES (double-length) key pair encrypted under the MFK. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. This field contains a 74 byte value. The following headers are supported: 1CDNE000, 1CDNN000,1CDVE000, and 1CDVN000.

For CSC card verification this field must contain the 2key-3DES (double-length) key pair encrypted under the MFK. This field contains a 74 byte value. The following headers are supported: 1mXNE000, 1mXNN000, 1mXVE000, and 1mXVN000.

E_{KPE}(ANSI PIN Block)

Field 5, the DES or 3DES encrypted ANSI PIN block to be verified. This field must contain a 16 hexadecimal character value.

Header,$E_{MFK.E}$(KPE/DK),MAC

Field 6, the PIN Encryption Key or Base Derivation Key encrypted under the MFK. This field contains a 74 byte value. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The AKB header for the PIN encryption key (KPE) must be one of the following values: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000. For the Base Derivation Key (BDK), used to process a DUKPT encrypted PIN block, the AKB header must be one of the following values:1dDNE000 and 1dDNN000.

`Bank ID`

Field 7, the bank ID field for the Identikey card issuer. The ID is specified by the issuer and can be a two-, six-, or eight byte decimal value.

Data Type	Allowable Size (bytes)
Backward index (algorithm number less than 65)	2
ISO number	6
Route and transfer number	8

`Partial PAN`

Field 8, validation data. This value is unique for each card holder, and in the case of this command, is the partial Primary Account Number (PAN). This field contains a 4 to 19 byte decimal value.

Header,$E_{MFK.E}$(KPV),MAC

Field 9, the PIN Verification Key encrypted under the MFK. This field contains a 74 byte value. When option 4A is enabled, this key can be a 1key -3DES (single-length) key. The following headers are supported: 1VBNE000, 1VBNN000, 1VBVE000, and 1VBVN000.

`PVN-2`

Field 10, the PIN Verification Number to be verified. This field contains a 4 to 16 byte hexadecimal value.

`PVN-2 Type`

Field 11, the PVN-2 type. This field indicates whether the PVN-2 should be converted to a decimal value. This field is 1 byte, it contains the numbers 0 or 1. The following table identifies the value for each type of PVN-2.

Value	Action
0	Convert PVN-2 to a decimal value
1	Don’t convert PVN-2; leave it as a hexadecimal value

`PVN-1 Flag`

Field 12, a flag that indicates that 8 digits of the PVN-1 value should be used to compute the PVN-2. This field is 1 byte, it contains the number 8.

`PVN-2 Start-Compare Flag`

Field 13, a flag that specifies the starting position within the generated PVN-2 for the comparison. This field is 1 byte, it contains the number 1.

`PIN Block Data`

Field 14, PIN block data. When a master/session KPE encrypts the ANSI PIN block, this field must contain the rightmost 12 PAN digits excluding the check digit.

When a DUKPT key encrypts the ANSI PIN block, three fields are present. See VISA Derived Unique Key Per Transaction PIN Block for more information on the contents of these three fields.

Table 4-77 Command 3A: Verify Card and PIN - Atalla Bilevel

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	3A
1	PIN Algorithm,Card Algorithm	3	2, 3, 4, and “,”
2	CVV/CVC/CSC Data	1 - 32 or 19	0 - 9
3	CVV/CVC/CSC	1-8, varies	0 - 9 and “,”
4	Header,$E_{MFK.E}$(KCVV/KCVC/KCSC),MAC	74	printable ASCII
5	E_{KPE}(ANSI PIN Block)	16	0 - 9, A - F
6	Header,$E_{MFK.E}$(KPE/DK),MAC	74	printable ASCII
7	Bank ID	2, 6, 8	0 - 9
8	Partial PAN	4 - 19	0 - 9
9	Header,$E_{MFK.E}$(KPV),MAC	74	printable ASCII
10	PVN-2	4 - 16	0 - 9, A - F
11	PVN-2 type	1	0, 1
12	PVN-1 flag	1	8
13	PVN-2 start-compare flag	1	1
14	PIN block data KPE DUKPT**	12**	0 - 9**

** See VISA Derived Unique Key Per Transaction PIN Block for information on PIN block data.

Responding Parameters

`4A`

Field 0, the response identifier.

`Card Verification Indicator`

Field 1, the card verification indicator. The value will be Y if the card verified, or N if the card did not verify.

If multiple CSCs were present in field 3 of the command, this field will contain either a Y or N, separated by a comma, for each CSC.

`PIN Verification/Sanity Indicator`

Field 2, the PIN verification and sanity indicator. A sanity check is performed on the decrypted ANSI PIN block. If the sanity check is successful, the PIN verification step is performed. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN verification was successful.
N	PIN verification failed.
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.
C	Card verification failed.

`KCVV/KCVC/KCSC check digits`

Field 3, check digits; the first four hexadecimal digits that result from encrypting zeros using the KCVV/KCVC/KCSC. If option 88 is enabled, this field will contain six check digits.

Table 4-78 Response 4A: Verify Card and PIN - Atalla Bilevel

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	4A
1	Card Verification Indicator	varies	Y, N, “,”
2	PIN Verification /Sanity Check Indicator	1	Y, N, S, L, C
3	KCVV/KCVC/KCSC check digits	4, 6	0 - 9, A - F

Usage Notes

Perform the following task before using Command 3A:

- Generate the AKBs for the PIN Encryption, Card Verification and PIN Verification keys.

- When option A2 = “D” or “B”, the Base Derivation Key provided in field 6 must be a 2key-3DES Key. If it is not, an error that starts with <00#0106..#> will be returned.
- When option A2 = “S”, an error that starts with <00#0316..#>will be returned if field 16 contains the letter “D”.
- When option A2 = “D”, an error that starts with <00#0316..#>will be returned if field 16 contains the letter “S”. The number “1” in field 16 is allowed only if the Base Derivation Key provided in field 6 is a 2key-3DES Key.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Master/Session KPE encrypted PIN block - Visa CVV

- PAN: 12345678901234560
- Expiration Date: 0806 (June 2008)
- Service Code: 101
- CVV/CVC Data: 123456789012345600806101
- CVV/CVC: 505
- Clear-text KCVV: 0123456789ABCDEF FEDCBA9876543210,
  the KCVV in AKB format =
  1CDNE000,4F7EFE3F44984427CF46B823CE4BDE1839E35E6F46EB2814,
  26A5545D383FA701
- Clear-text PIN Encryption Key (KPE): 0000111122223333
  the KPE in AKB format =
  1PUNE000,C118AFA8BDA9F01E832B5725DFCE45D60C5A5CA83A4D1258,
  0DBB2D4988A072F2
- ANSI PIN Block: 051262276FEDCBA9
- E_{KPE}(ANSI PIN block): 40C7F0A40E679F82
- Bank ID: 591210
- Partial PAN: 5678901
- Clear-text PIN Verification Key (KPV): ABCDEF0123456789
  the KPV in AKB format =
  1VBNE000,44DF0C0126B29C120B5FF8DC45E4CEF3983E3F17A91A3A95,
  E52824C52AA5A291

- PVN-2: 6341608139743500
- PVN-2 type: 0
- PVN-1 flag: 8
- PVN-2 start-compare flag: 1
- PIN block data: 567890123456

  The command looks like this:

  ```
  <3A#4,2#12345678901234560080610l#505#1CDNE000,4F7EFE3F44984427CF4
  6B823CE4BDE1839E35E6F46EB2814,26A5545D383FA701#40C7F0A40E679F82#1
  PUNE000,C118AFA8BDA9F01E832B5725DFCE45D60C5A5CA83A4D1258,0DBB2D49
  88A072F2#591210#5678901#1VBNE000,44DF0C0126B29C120B5FF8DC45E4CEF3
  983E3F17A91A3A95,E52824C52AA5A291#6341608139743500#0#8#1#56789012
  3456#>
  ```

  The Network Security Processor returns the following response:

  ```
  <4A#Y#Y#08D7#>
  ```

DUKPT encrypted PIN block - AMEX CSC version 1.0

- PAN: 371234567890123
- Expiration Date: 9912
- CSC Data: 3712345678901239912
- CSCs: 61247, 8720, 552
- Clear-text Card Security Code Key (KCSC): 0123456789ABCDEF FEDCBA9876543210 the KCSC in AKB format = 1mXNE000,823244EC4A4DFA0198097F067749BE0C9448676EB51ABAE2,459E9F9E E15CE583
- Clear-text Base Derivation Key (BDK): 0123456789ABCDEF FEDCBA9876543210 the BDK in AKB format = 1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,342946FE 884AA8B2
- ANSI PIN Block: 0512266BA9876FED
- E_{KPE}(ANSI PIN block): 66B04757AD9043D2
- Bank ID: 591210
- Partial PAN: 5678901

- Clear-text PIN Verification Key (KPV): ABCDEF0123456789
  the KPV in AKB format =
  1VBNE000,44DF0C0126B29C120B5FF8DC45E4CEF3983E3F17A91A3A95,
  E52824C52AA5A291
- PVN-2: 6341608139743500
- PVN-2 type: 0
- PVN-1 flag: 8
- PVN-2 start-compare flag: 1
- PIN block data: 123456789012
- KSN: FFFF9876543210E00001
- PIN Encryption Key Derivation Algorithm: D

  The command looks like this:

```
<3A#4,3#3712345678901239912#61247,8720,552#1mXNE000,823244EC4A4DF
A0198097F067749BE0C9448676EB51ABAE2,459E9F9EE15CE583#66B04757AD90
43D2#1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,34
2946FE884AA8B2#591210#5678901#1VBNE000,44DF0C0126B29C120B5FF8DC45
E4CEF3983E3F17A91A3A95,E52824C52AA5A291#6341608139743500#0#8#1#12
3456789012#FFFF9876543210E00001#D#>
```

  The Network Security Processor returns the following response:

```
<4A#Y,Y,Y#Y#08D7#>
```

Verify Card and PIN - NCR (Command 3A)

Command 3A is a combination command that performs both card and PIN verification. Three card verification algorithms are supported: VISA Card Verification Value (CVV), MasterCard Card Validation Code (CVC), and American Express Card Security Code (CSC). If the card verification is successful, the command then attempts to verify a customer PIN using the NCR PIN verification algorithm. The ANSI PIN block to be verified can be DES or 3DES encrypted using either a master/session PIN encryption key or a Derived Unique Key Per Transaction (DUKPT) key.

This command is enabled in the Network Security Processor’s default security policy.

Command

```
<3A#PIN Algorithm,Card Algorithm#CVV/CVC/CSC Data#CVV/CVC/CSC#
Header,EMFK.E(KCVV/KCVC/KCSC),MAC#EKPE(ANSI PIN Block)#
Header,EMFK.E(KPE/DK),MAC#Conversion Table#Offset#Validation Data#
Pad#PLEN#Header,EMFK.E(KPV),MAC#Padding Flag#Start-Count Position#
Select-PLEN Position#PIN Block Data#>
```

Response

```
<4A#Card Verification Indicator#PIN Verification/Sanity Indicator#
KCVV/KCVC/KCSC check digits#>[CRLF]
```

Calling Parameters

`3A`

Field 0, the command identifier.

`PIN Algorithm,Card Algorithm`

Field 1, the PIN algorithm value followed by a comma “,” followed by the card verification algorithm. The PIN algorithm must be set to “6” for NCR. The card verification algorithms are:

Value	Card Algorithm
2	VISA Card Verification Value (CVV)
2	MasterCard Card Validation Code (CVC)
3	American Express Card Security Code (CSC) Version 1.0

For example, NCR PIN verification with AMEX CSC card verification is 6,3.

`CVV/CVC/CSC Data`

Field 2, the data required to verify the CVV/CVC/CSC. For CVV/CVC card verification this field contains a 1 to 32 digit value, which should include the PAN followed by the expiration date in YYMM format, followed by the service code.

```
PAN|Expiration Date|Service Code
```

For CSC card verification, this field must contain a 19 digit value in the form of a 15 digit PAN followed by a 4 digit expiration date in YYMM format. The leftmost two PAN digits must be 34 or 37.

```
PAN|Expiration Date
```

`CVV/CVC/CSC`

Field 3, the CVV, CVC, or CSC(s) to be verified.

The CVV/CVC can be a 1 to 8 digit value. When option 4D is enabled, this field must contain a 3 to 8 digit CVV/CVC value.

There are three lengths of CSCs that can be verified, this field can contain 1, 2, or all 3 CSC values, each separated by a comma. When present, the CSC(s) must be 3, 4, or 5 digits in length.

Header,$E_{MFK.E}$(KCVV/KCVC/KCSC),MAC

Field 4, for CVV or CVC card verification this field must contain the 2key-3DES (double-length) key pair encrypted under the MFK. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. This field contains a 74 byte value. The following headers are supported: 1CDNE000, 1CDNN000,1CDVE000, and 1CDVN000.

For CSC card verification this field must contain the 2key-3DES (double-length) key pair encrypted under the MFK. This field contains a 74 byte value. The following headers are supported: 1mXNE000, 1mXNN000, 1mXVE000, and 1mXVN000.

E_{KPE}(ANSI PIN Block)

Field 5, the DES or 3DES encrypted ANSI PIN block to be verified. This field must contain a 16 hexadecimal character value.

Header,$E_{MFK.E}$(KPE/DK),MAC

Field 6, the PIN Encryption Key or Base Derivation Key encrypted under the MFK. This field contains a 74 byte value. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The AKB header for the PIN encryption key (KPE) must be one of the following values: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000. For the base Derivation Key (DK), used to process a DUKPT encrypted PIN block, the AKB header must be one of the following values:1dDNE000 and 1dDNN000.

`Conversion Table`

Field 7, a table that maps hexadecimal digits (0 through 9, A through F) to decimal digits (0 through 9). This field contains a 16 byte decimal value containing the clear-text Conversion Table or a volatile table location. When option 48 is enabled, this field contains the conversion table in AKB format (the header must be 1nCNE000) or a volatile table location. When option 4E is enabled, all three forms of the conversion table (clear-text, decrypted, or value stored in volatile table location) must adhere to these rules:

- The conversion table must have at least eight unique digits.
- No single digit can occur more than four times.

`Offset`

Field 8, an offset value applied to the algorithm-generated PIN before comparing it with the customer-entered PIN. This field contains a 4 to 12 byte decimal value.

`Validation Data`

Field 9, validation data. This value is unique for each card holder, and is typically the account number. This field contains a 4 to 16 byte decimal value. When option 4C is enabled, the value supplied in this field must be 12 digits in length and equal to the PIN Block Data value supplied in field 17.

`Pad`

Field 10, the pad character which right-pads the validation data. This field contains a one byte hexadecimal value. The pad character is only applied when the validation data is less than 16 digits.

`PLEN`

Field 11, the number of contiguous PIN digits selected for verification; the PIN length, or PLEN. This field is 1 byte, it contains a hexadecimal value in the range of 4 through C.

Header,$E_{MFK.E}$(KPV),MAC

Field 12, the PIN Verification Key encrypted under the MFK. This field contains a 74 byte value. When option 4A is enabled, this key can be a 1key -3DES (single-length) key. The following headers are supported: 1V3NE000, 1V3NN000, 1V3VE000, and 1V3VN000.

`Padding Flag`

Field 13, a flag that indicates whether the validation data (Field 7) is to be padded on the left or on the right. This field contains the character L or R.

`Counting Flag`

Field 14, a flag that indicates whether the counting scheme for selecting the PIN digit for verification is left or right. This field contains the character L or R.

`Start-Count Position`

Field 15, the field that indicates the starting position for the counting scheme measured from either the left or right of the entered PIN depending on field 12. This field is 1 decimal digit in the range of 1 through 9.

`Select-PLEN Position`

Field 16, the field that indicates the beginning position (from the direction of the counting flag, starting with 0) for selecting PLEN characters from the output of the decimalization step. This field contains one hexadecimal character. If the counting flag is “L”, the leftmost digit of the decimalized result is position zero. If the counting flag is “R,” the rightmost digit of the decimalized result is position zero.

`PIN Block Data`

Field 17, PIN block data. When a master/session KPE encrypts the ANSI PIN block, this field must contain the rightmost 12 PAN digits excluding the check digit.

When a DUKPT key encrypts the ANSI PIN block, three fields are present. See VISA Derived Unique Key Per Transaction PIN Block for more information on the contents of these three fields.

Table 4-79 Command 3A: Verify Card and PIN - NCR

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	3A
1	PIN Algorithm,Card Algorithm	3	2, 3, and “,”
2	CVV/CVC/CSC Data	1 - 32 or 19	0 - 9
3	CVV/CVC/CSC	1-8, varies	0 - 9 and “,”
4	Header,$E_{MFK.E}$(KCVV/KCVC/KCSC),MAC	74	printable ASCII
5	E_{KPE}(ANSI PIN Block)	16	0 - 9, A - F
6	Header,$E_{MFK.E}$(KPE/DK),MAC	74	printable ASCII
7	Conversion Table	16	0 - 9
8	Offset	4 - 16	0 - 9
9	Validation data	4 - 16	0 - 9
10	Pad for validation data	1	0 - 9, A - F
11	PLEN	1	4 - 9, A - C
12	Header,$E_{MFK.E}$(KPV),MAC	74	printable ASCII
13	Padding Flag	1	L or R
14	Counting Flag	1	L or R

Table 4-79 Command 3A: Verify Card and PIN - NCR (continued)

Field	Contents	Length (bytes)	Legal Characters
15	Start-Count Position	1	1 - 9
16	Select-PLEN Position	1	0 - 9, A - F
17	PIN block data KPE DUKPT**	12 **	0 - 9 **

** See VISA Derived Unique Key Per Transaction PIN Block for information on PIN block data.

Responding Parameters

`4A`

Field 0, the response identifier.

`Card Verification Indicator`

Field 1, the card verification indicator. The value will be Y if the card verified, or N if the card did not verify.

If multiple CSCs were present in field 3 of the command, this field will contain either a Y or N, separated by a comma, for each CSC.

`PIN Verification/Sanity Indicator`

Field 2, the PIN verification and sanity indicator. A sanity check is performed on the decrypted ANSI PIN block. If the sanity check is successful, the PIN verification step is performed. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN verification was successful.
N	PIN verification failed.
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.
C	Card verification failed.

`KCVV/KCVC/KCSC check digits`

Field 3, check digits; the first four hexadecimal digits that result from encrypting zeros using the KCVV/KCVC/KCSC. If option 88 is enabled, this field will contain six check digits.

Table 4-80 Response 4A: Verify Card and PIN - NCR

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	4A
1	Card Verification Indicator	varies	Y, N, “,”
2	PIN Verification /Sanity Check Indicator	1	Y, N, S, L, C
3	KCVV/KCVC/KCSC check digits	4, 6	0 - 9, A - F

Usage Notes

- Generate the PIN Encryption, Card Verification, and PIN Verification keys.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

CVV card verification and 3DES master/session PIN verification

- Clear-text KCVV: 0123456789ABCDEF FEDCBA9876543210
  The KCVV in AKB format:
  1CDNE000,4F7EFE3F44984427CF46B823CE4BDE1839E35E6F46EB2814,26A5545D3
  83FA701
- PAN: 1234567890123456
- Expiration Date: 0806 (June 2008)
- Service Code: 101
- CVV: 921
- PIN: 1234
- ANSI PIN Block: 0412719876FEDCBA
- Clear-text PIN Encryption Key: 1111111111111111 2222222222222222
  The PIN Encryption Key in AKB format:
  1PUNE000,302A81872F21CE1D0A8791AC9B0C7A6ABB5A9D73F6EB4872,B8E0CBEE
  B718723D
- Encrypted ANSI PIN Block: 6EF2480ACCF9FC50

- Conversion Table: 0123456789012345
- Offset: 9926
- Validation data:1234567890123456
- Pad for validation data: F
- Check-Length: 4
- Clear-text PIN Verification Key: 3333 3333 3333 3333
  The PIN Verification Key in AKB format:
  1VNNE000,4367E5C600089BC8AA8E8921BEA94E5DAAFEF53835D45C1C,70147571C43DF493
- Padding Flag: R
- Counting Flag: L
- Start-Count Position: 1
- Select-PLEN Position: 0
- The PIN block data: 456789012345

The command looks like this:

```
<3A#6,2#12345678901234560806101#921#1CDNE000,4F7EFE3F44984427CF46
B823CE4BDE1839E35E6F46EB2814,26A5545D383FA701#6EF2480ACCF9FC50#1P
UNE000,302A81872F21CE1D0A8791AC9B0C7A6ABB5A9D73F6EB4872,B8E0CBEEB
718723D#0123456789012345#9926#1234567890123456#F#4#1VNNE000,4367E
5C600089BC8AA8E8921BEA94E5DAAFEF53835D45C1C,70147571C43DF493#R#L#
1#0#456789012345#>
```

The Network Security Processor returns the following response:

```
<4A#Y#Y#08D7#>
```

CSC card verification and DES master/session PIN verification

- Clear-text KCSC: 0123456789ABCDEF FEDCBA9876543210
  The KCSC in AKB format:
  1mXNE000,823244EC4A4DFA0198097F067749BE0C9448676EB51ABAE2,459E9F9EE15CE583
- PAN: 345678901234567
- Expiration Date: 0806 (June 2008)
- CSCs: 25199,0365,187

- PIN: 1234
- ANSI PIN Block: 041262876FEDCBA9
- Clear-text PIN Encryption Key: 1111 1111 1111 1111
  The PIN Encryption Key in AKB format:
  1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC785574,003D0F146BFFC1F9
- Encrypted ANSI PIN Block: F2106C85AF495833
- Conversion Table: 0123456789012345
- Offset: 4204
- Validation data: 345678901234567
- Pad for validation data: F
- Check-Length: 4
- Clear-text PIN Verification Key: 3333 3333 3333 3333
  The PIN Verification Key in AKB format:
  1VNNE000,4367E5C600089BC8AA8E8921BEA94E5DAAFEF53835D45C1C,70147571C43DF493
- Padding Flag: R
- Counting Flag: L
- Start-Count Position: 1
- Select-PLEN Position: 0
- The PIN block data: 567890123456

The command looks like this:

```
<3A#6,3#3456789012345670806#25199,0365,187#1mXNE000,823244EC4A4DFA0
198097F067749BE0C9448676EB51ABAE2,459E9F9EE15CE583#F2106C85AF495833
#1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC785574,003D0F14
6BFFC1F9#0123456789012345#4204#345678901234567#F#4#1VNNE000,4367E5C
600089BC8AA8E8921BEA94E5DAAFEF53835D45C1C,70147571C43DF493#R#L#1#0#
567890123456#>
```

The Network Security Processor returns the following response:

```
<4A#Y,Y,Y#Y#08D7#>
```

CVC card verification and 3DES DUKPT PIN verification

- Clear-text KCVC: 0123456789ABCDEF FEDCBA9876543210
  The KCVV in AKB format:
  1CDNE000,4F7EFE3F44984427CF46B823CE4BDE1839E35E6F46EB2814,26A5545D3
  83FA701
- PAN: 1234567890123456
- Expiration Date: 0806 (June 2008)
- Service Code: 101
- CVV: 921
- PIN: 1234
- ANSI PIN Block: 0412719876FEDCBA
- Clear-text Base Derivation Key: 0123456789ABCDEF FEDCBA9876543210
  The Base Derivation Key in AKB format:
  1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,342946FE
  884AA8B2
- Encrypted ANSI PIN Block: 39510B23F12E4ABD
- Conversion Table: 0123456789012345
- Offset: 9926
- Validation data:1234567890123456
- Pad for validation data: F
- Check-Length: 4
- Clear-text PIN Verification Key: 3333 3333 3333 3333
  The PIN Verification Key in AKB format:
  1VNNE000,4367E5C600089BC8AA8E8921BEA94E5DAAFEF53835D45C1C,70147571
  C43DF493
- Padding Flag: R
- Counting Flag: L
- Start-Count Position: 1
- Select-PLEN Position: 0
- The PIN block data:

- Rightmost 12 PAN digits excluding the check digit: 456789012345
- KSN: 9876543210E00008
- Algorithm: 1

The command looks like this:

```
<3A#6,2#12345678901234560806101#921#1CDNE000,4F7EFE3F44984427CF46
B823CE4BDE1839E35E6F46EB2814,26A5545D383FA701#39510B23F12E4ABD#1d
DNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,342946FE8
84AA8B2#0123456789012345#9926#1234567890123456#F#4#1VNNE000,4367E
5C600089BC8AA8E8921BEA94E5DAAFEF53835D45C1C,70147571C43DF493#R#L#
1#0#456789012345#9876543210E00008#D#>
```

The Network Security Processor returns the following response:

```
<4A#Y#Y#08D7#>
```

Generate PVN and IBM Offset (Command 3D)

Command 3D generates both an Identikey PVN, and an IBM3624 offset from the account number and encrypted PIN.

This command is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<3D#PIN Block Type#EKPE(PIN Block)#Header,EMFK.E(KPE),MAC#Bank ID#
Partial PAN#Conversion Table#Validation Data#Pad#
Header,EMFK.E(KPV),MAC#PIN Block Data#>
```

Response

```
<4D#PVN/Sanity Check Indicator#[IBM 3624 Offset#]
[IBM 3624 Sequence Number#]>[CRLF]
```

Calling Parameters

`3D`

Field 0, the command identifier.

`PIN Block Type`

Field 1, the incoming PIN block type. This field is 1 byte, it can contain the numbers 1,2 or 3.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN/pad character / Docutel

E_{KPE}(PIN Block)

Field 2, the encrypted PIN block. This field contains 16 hexadecimal characters.

Header,$E_{MFK.E}$(KPE),MAC

Field 3, the PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

`Bank ID`

Field 4, the Bank ID is specified by the user. This field can be a 2, 6, or 8 byte decimal value.

`Partial PAN`

Field 5, the portion of the Primary Account Number to be used in the Identikey PVN generation process. This field contains a 4 to 19 byte decimal value.

`Conversion Table`

Field 6, a table that maps hexadecimal digits (0 through 9, A through F) to decimal digits (0 through 9). This field contains a 16 byte decimal value containing the clear-text Conversion Table or a volatile table location. When option 48 is enabled, this field contains the conversion table in AKB format (the header must be 1nCNE000) or a volatile table location. When option 4E is enabled, all three forms of the conversion table (clear-text, decrypted, or value stored in volatile table location) must adhere to these rules:

- The conversion table must have at least eight unique digits.
- No single digit can occur more than four times.

`Validation Data`

Field 7, validation data. This value is unique for each card holder, and is typically the account number. This field contains a 4 to 16 byte decimal value. When option 4C is enabled, the value supplied in this field must be 12 digits in length and equal to the PIN Block Data value supplied in field 10.

`Pad`

Field 8, the pad character which right-pads the validation data. This field contains a one byte hexadecimal value. The pad character is only applied when the validation data is less than 16 digits.

Header,$E_{MFK.E}$(KPV),MAC

Field 9, the PIN Verification Key encrypted under the MFK. This field contains a 74 byte value. When option 4A is enabled, this key can be a 1key -3DES (single-length) key. The following headers are supported: 1V3NE000, 1V3NN000, 1V3VE000, and 1V3VN000.

`PIN Block Data`

Field 10, PIN block data. Its contents depend on the PIN block type. See PIN Block Types.

Table 4-81 Command 3D: Generate PVN and IBM Offset

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	3D
1	PIN block type	1	1 - 3

Table 4-81 Command 3D: Generate PVN and IBM Offset (continued)

Field	Contents	Length (bytes)	Legal Characters
2	E_{KPE}(PIN Block)	16	0 - 9, A - F
3	Header,$E_{MFK.E}$(KPE),MAC*	74	printable ASCII
4	Bank ID	2, 6, 8	0 - 9
5	Partial PAN	4 - 16	0 - 9
6	Conversion Table*	16	0 - 9
7	Validation data	4 - 16	0 - 9, A - F
8	Pad for validation data	1	0 - 9, A - F
9	Header,$E_{MFK.E}$(KPV),MAC*	74	printable ASCII
10	PIN block data**		

* Can be a volatile table location.
** See PIN Block Types for information on PIN block data.

Responding Parameters

```
4D
```

Field 0, the response identifier.

```
PVN or a Sanity Check Indicator
```

Field 1, the PVN if the PIN block passes the sanity test. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

```
[IBM 3624 Offset#]
```

Field 2, the offset associated with the PIN. The length of the offset is equal to the length of the PIN supplied in the command. This field is present only when the PIN passes the sanity test.

```
[IBM 3624 Sequence Number#]
```

Field 3, the IBM 3624 PIN block sequence number. This field is returned only if the PIN Block type is IBM 3624. This field contains a 2 byte hexadecimal value.

Table 4-82 Response 4D: Generate PVN and IBM Offset

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	4D
1	PVN/Sanity Check Indicator	1, 8	0 - 9, S, L
2	IBM 3624 Offset	0, 4 - 12	0 - 9
3	IBM 3624 Sequence Number*	0, 2	0 - 9, A - F

* Optional field; returned only if the PIN block type is IBM 3624.

Usage Notes

Perform the following task before using Command 3D:

- Generate the PIN Encryption Key, and PIN Verification Key.

Example

Generate a PVN and IBM offset from an ANSI PIN block

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- PIN block type: ANSI (1)
- Encrypted PIN block: 090E8CA3CF5D2AD8
- Clear-text PIN Encryption Key: 1111 1111 1111 1111
  The PIN Encryption Key in AKB format:
  1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF679984BFC785574,003D0F146
  BFFC1F9
- Bank ID: 26
- The partial PAN used for Identikey: 123456123456
- Conversion Table: 0123456789012345
- Validation data: 123456123456
- Pad for validation data: F

- Clear-text PIN Verification Key: 3333 3333 3333 3333
  The PIN Verification Key in AKB format:
  1V3NE000,F0BD2DA1B14F54B16E5541B6F61C35AA4F56EDEFFB7ADDD3,450CC023 2392B2F5

- The PIN block data: 123456123456

The command looks like this:

```
<3D#1#090E8CA3CF5D2AD8#1PUNE000,302A81872F21CE1D36DCF0204EEB0FDCF
679984BFC785574,003D0F146BFFC1F9#26#123456123456#0123456789012345
#123456123456#F#1V3NE000,F0BD2DA1B14F54B16E5541B6F61C35AA4F56EDEF
FB7ADDD3,450CC0232392B2F5#123456123456#>
```

The Network Security Processor returns the following response:

```
<4D#31724120#6140#>
```

Decrypt PIN (Command 90)

Command 90 decrypts an incoming PIN block and returns the clear-text PIN.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To enable this command, you must purchase this command in the form of a command 105. To use this command, you must add it to the Network Security Processor's security policy.

Command

```
<90#PIN Block Type#EKPE(PIN Block)#Header,EMFK.E(KPE),MAC#
PIN Block Data#>
```

Response

```
<A0#Clear-Text PIN or Sanity Check Indicator#>[CRLF]
```

Calling Parameters

`90`

Field 0, the command identifier.

`PIN Block Type`

Field 1, the type of the incoming PIN block. The PIN block types that this command supports are listed in the following table:

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN/pad character
3	Docutel

E_{KPE}`(PIN Block)`

Field 2, the encrypted PIN block. This field contains 16 hexadecimal characters.

`Header,`$E_{MFK.E}$`(KPE),MAC`

Field 3, the PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

`PIN Block Data`

Field 4, PIN block data. Its contents depend on the PIN block type. See PIN Block Types.

Table 4-83 Command 90: Decrypt PIN

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	90
1	PIN block type	1	1 - 3
2	E_{KPE}(PIN Block)	16	0 - 9, A - F
3	Header,$E_{MFK.E}$(KPE),MAC*	74	printable ASCII
4	PIN block data**		

* Can be a volatile table location.
** See PIN Block Types information on PIN block data.

Responding Parameters

`A0`

Field 0, the response identifier.

`Clear-Text PIN or Sanity Check Indicator`

Field 1, the clear-text PIN is present if the PIN block passed the sanity test. When the sanity test fails, this field will contain one of the following values:

Value	Description
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

Table 4-84 Response A0: Decrypt PIN

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	A0
1	Clear-text PIN or sanity check indicator	1 - 12 if clear-text PIN is returned, otherwise, 1	0 - 9, S, L

Usage Notes

- This command is used for verifying host-based PINs in proprietary networks and for verifying PINs in on-others transactions initiated in shared networks.
- Before using this command, generate the PIN Encryption Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Decrypt an encrypted PIN block

- Clear-text PIN block: 1234 5FFF FFFF FFFF
  The PIN block encrypted under the PIN Encryption Key: 7B58 719B 354B 147A
- Clear-text PIN Encryption Key: 2233 2233 2233 2233
  The PIN Encryption Key in AKB format:
  1PUNE000,267DD77F01ACDF6DD16B86FE817BA59C6C6E3FF08CD44BD4,0D03D1F
  B57932253
- PIN block data; 12 digits of the Primary Account Number: 9876 5432 1012

The command looks like this:

```
<90#1#7B58719B354B147A#1PUNE000,267DD77F01ACDF6DD16B86FE817BA59C6
C6E3FF08CD44BD4,0D03D1FB57932253#987654321012#>
```

The Network Security Processor returns the following response:

```
<A0#12345#>
```

PIN Translate (ANSI to PIN/Pad) and MAC Verification (Command BA)

Command BA is used to accomplish two functions in a single command. It translates PINs from encryption under one key to encryption under another, and it verifies a Message Authentication Code (MAC). The incoming PIN Encryption Key is designated as KPE_I, and the outgoing PIN Encryption Key is designated as KPE_O.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

This command will return an error if either option 46 or 47 is enabled.

Command

```
<BA#13#Header,EMFK.E(KPEI),MAC#Header,EMFK.E(KPEO),MAC#
EKPEI(PIN Block)#Pad#ANSI PAN Digits#[Header,EMFK.E(KMAC),MAC]#Data#
MAC#>
```

Response

```
<CA#EKPEO(PIN/Pad PIN block)#Sanity Check#KMAC Check Digits#
Verification Flag#>[CRLF]
```

Calling Parameters

`BA`

Field 0, the command identifier.

`13`

Field 1, ANSI PIN block to PIN/pad block.

Header,$E_{MFK.E}(KPE_I)$,MAC

Field 2, the incoming PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

Header,$E_{MFK.E}(KPE_O)$,MAC

Field 3, the outgoing PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, the length of the KPEo must be equal to or greater than the length of the KPEi (field 2). The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000.

`E`$_{KPEI}$`(PIN Block)`

Field 4, the incoming PIN Block encrypted under the incoming PIN Encryption Key. This field contains 16 hexadecimal characters.

`Pad`

Field 5, the pad character in the PIN pad block. This field is 1 byte, it can contain a hexadecimal value, X or W. When this field contains the value X or W, the pad character used in the incoming PIN block will also be used as the outgoing pad character.

`ANSI PAN Digits`

Field 6, the Primary Account Number digits used in the incoming ANSI PIN block. This field contains a 12 byte decimal value.

`Header,E`$_{MFK.E}$`(KMAC),MAC`

Field 7, the MAC Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1MDNE000, 1MDNN000, 1MDVE000, and 1MDVN000.

`Data`

Field 8, this data will be authenticated according to ANSI specification X9.19. This field can be from one to 240 bytes long and can contain the characters A to Z, the numbers 0 through 9, and ",", ".", and " ".

`MAC`

Field 9, the MAC to be verified. This is an 8 byte hexadecimal value.

Table 4-85 Command BA: PIN Translate (ANSI to PIN/Pad) and MAC Verification

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	2	BA
1	ANSI PIN block	2	13
2	Header,E$_{MFK.E}$(KPE$_I$),MAC*	74	printable ASCII
3	Header,E$_{MFK.E}$(KPE$_O$),MAC*	74	printable ASCII
4	E$_{KPEI}$(Encrypted PIN Block)	16	0 - 9, A - F
5	PIN/Pad Character	1	0 - 9, A - F, W, X
6	ANSI PAN Data	12	0 - 9
7	Header,E$_{MFK.E}$(KMAC),MAC*	74	printable ASCII

Table 4-85 Command BA: PIN Translate (ANSI to PIN/Pad) and MAC Verification (continued)

Field	Contents	Length (bytes)	Legal Characters
8	Data per ANSI X9.19 only	1-240	0 - 9, A - Z, , . " "
9	MAC	8	0 - 9, A - F

* Can be a volatile table location.

Responding Parameters

`CA`

Field 0, the response identifier.

`E`$_{KPE_O}$ `(PIN/Pad PIN block)`

Field 1, the module decrypts the incoming PIN, verifies the logical validity of the data, uses the PIN block data to reformat the PIN block to the PIN/Pad PIN Block, and encrypts the PIN block with the outgoing KPE [E$_{KPE_O}$ (ANSI PIN block)].

`Sanity Check`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

`KMAC Check Digits`

Field 3, check digits; the first four digits that result from encrypting zeros using the message authentication key. If option 88 is enabled, this field will contain six check digits.

`Verification Flag`

Field 4, the MAC verification flag. This field returns Y if the MAC is verified (meaning that the data has not been modified in transit), or N if the MAC is not verified.

Table 4-86 Response CA: PIN Translate (ANSI to PIN/Pad) and MAC Verification

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	CA
1	E$_{KPE_O}$ (PIN/Pad PIN block)	16	0 - 9, A - F

Table 4-86 Response CA: PIN Translate (ANSI to PIN/Pad) and MAC Verification (continued)

Field	Contents	Length (bytes)	Legal Characters
2	Sanity Check	1	Y, N, L
3	KMAC Check Digits	4 or 6	0 - 9, A - F
4	Verification Flag	1	Y, N

Usage Notes

- Generate the Message Authentication Code Key and the incoming and outgoing PIN Encryption Keys.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate a PIN from ANSI to PIN/Pad PIN block and verifying a MAC

- Clear-text incoming PIN Encryption Key: 07CE A74F 4607 5D8F
  The incoming PIN Encryption Key in AKB format:
  1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,C47517937
  1FDBE43
- Clear-text outgoing PIN Encryption Key: D029 23D9 AD4F E90B
  The outgoing PIN Encryption Key in AKB format:
  1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53AF657A4508,7ED6957
  0CDE54F30
- The PIN block encrypted under the PIN Encryption Key: 5196 681F 910C 408C
- ANSI Primary Account Number digits: 1207 4108 1445
- Pad character: F
- Clear-text Message Authentication Code Key: 8FF4 98F1 B661 5151
  The Message Authentication Code Key in AKB format:
  1MDNE000,E210B2B1F67CE7F0F22434D315EA9361BE734B0079844FAF,381990A89
  D696C75
- Data to be authenticated:
  A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4Y5Z6A1B2C3D4E5F6
  G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X

- The MAC to be verified: 4316C2C1

The command looks like this:

```
<BA#13#1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,
C475179371FDBE43#1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53A
F657A4508,7ED69570CDE54F30#5196681F910C408C#F#120741081445#1MDNE0
00,E210B2B1F67CE7F0F22434D315EA9361BE734B0079844FAF,381990A89D696
C75#A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4Y5Z6A1B2C3D4E
5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X#4316C2C1#>
```

The Network Security Processor returns the following response:

```
<CA#7DBE8020E51B8C36#Y#1DE3#Y#>
```

Translate PIN (ANSI to PLUS) and Verify MAC (Command BB)

Command BB translates PINs from encryption under one key to another and verifies a MAC. The Network Security Processor decrypts the incoming ANSI PIN block, verifies the MAC, and encrypts the outgoing PIN block. The incoming PIN Encryption Key is designated as KPE_I, and the outgoing PIN Encryption Key is designated as KPE_O.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

Command

```
<BB#11#Header,E_MFK.E(KPE_I),MAC#Header,E_MFK.E(KPE_O),MAC#
E_KPEI(ANSI PIN Block)#ANSI PAN Digits#PLUS PAN Digits#
Header,E_MFK.E(KMAC),MAC#Data#MAC#>
```

Response

```
<CB#E_KPEO(PLUS PIN Block)#Sanity Check Indicator#KMAC Check Digits#
Verification Flag#>[CRLF]
```

Calling Parameters

`BB`

Field 0, the command identifier.

`11`

Field 1, the PIN block type; in this command, ANSI.

Header, $E_{MFK.E}(KPE_I)$, MAC

Field 2, the incoming PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

Header, $E_{MFK.E}(KPE_O)$, MAC

Field 3, the outgoing PIN Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, the length of the KPEo must be equal to or greater than the length of the KPEi (field 2). The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000.

E_{KPEI}(ANSI PIN Block)

Field 4, the incoming ANSI PIN Block encrypted under the incoming PIN Encryption Key.

This field contains 16 hexadecimal characters.

`ANSI PAN Digits`

Field 5, the ANSI Primary Account Number; the 12 rightmost digits of the Primary Account Number excluding the check digit. This field contains a 12 byte decimal value. When either option 46 or 47 is enabled, the value of this field and field 6 must be identical.

`PLUS PAN Digits`

Field 6, the PLUS Primary Account Number; the 12 leftmost digits of the Primary Account Number. This field contains a 12 byte decimal value.

Header,$E_{MFK.E}$(KMAC),MAC

Field 7, the Message Authentication Code Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1MDNE000, 1MDNN000, 1MDVE000, and 1MDVN000.

`Data`

Field 8, the data to be authenticated. This field can be up to 240 bytes long and can contain the numbers 0 through 9, the characters A to Z, “,”, “.”, and “ ”.

`MAC`

Field 9, the MAC to be verified. This field contains an 8 byte hexadecimal value.

Table 4-87 Command BB: Translate PIN (ANSI to PLUS) and Verify MAC

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	BB
1	ANSI PIN block	2	11
2	Header,$E_{MFK.E}$(KPE_I),MAC*	74	printable ASCII
3	Header,$E_{MFK.E}$(KPE_O),MAC*	74	printable ASCII
4	E_{KPEI}(ANSI PIN Block)	16	0 - 9, A - F
5	ANSI PAN Digits	12	0 - 9
6	PLUS PAN Digits	12	0 - 9
7	Header,$E_{MFK.E}$(KMAC),MAC*	74	printable ASCII
8	Data	1 - 240	0 - 9, A - Z, , . “ ”
9	MAC	8	0 - 9, A - F

* Can be a volatile table location.

Responding Parameters

`CB`

Field 0, the response identifier.

`E`$_{KPEO}$`(PLUS PIN Block)`

Field 1, the PIN block in a PLUS PIN block encrypted using the outgoing PIN Encryption Key.

`Sanity Check Indicator`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
S	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

`KMAC Check Digits`

Field 3, check digits; the first four digits that result from encrypting zeros using the Message Authentication Code Key. If option 88 is enabled, this field will contain six check digits.

`Verification Flag`

Field 4, the MAC verification flag. This field returns Y if the MAC is verified, otherwise, it returns N.

Table 4-88 Response BC: Translate PIN (ANSI to PLUS) and Verify MAC

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	CB
1	E_{KPEI}(PLUS PIN Block)	16	0 - 9, A - F
2	Sanity check indicator	1	Y, N, L
3	KMAC Check Digits	4 or 6	0 - 9, A - F
4	Verification flag	1	Y, N

Usage Notes

- Generate the incoming and outgoing PIN Encryption Keys.
- Generate the Message Authentication Code key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate a PIN from ANSI to PLUS PIN block and verifying a MAC

- Clear-text incoming PIN Encryption Key: 07CE A74F 4607 5D8F
  The incoming PIN Encryption Key in AKB format:
  1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,C47517937
  1FDBE43
- Clear-text outgoing PIN Encryption Key: D029 23D9 AD4F E90B
  The outgoing PIN Encryption Key in AKB format:
  1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53AF657A4508,7ED6957
  0CDE54F30
- The PIN block encrypted under the incoming PIN Encryption Key: 5196 681F 910C 408C
- ANSI Primary Account Number digits: 1207 4108 1445
- PLUS Primary Account Number digits: 2074 1081 4457
- Clear-text Message Authentication Code Key: 8FF4 98F1 B661 5151
  The Message Authentication Code Key in AKB format:
  1MDNE000,E210B2B1F67CE7F0F22434D315EA9361BE734B0079844FAF,381990A89
  D696C75
- Data to be authenticated:
  A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4Y5Z6A1B2C3D4E5F6
  G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X
- The MAC to be verified: 4316C2C1

The command looks like this:

```
<BB#11#1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,
C475179371FDBE43#1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53A
F657A4508,7ED69570CDE54F30#5196681F910C408C#120741081445#20741081
```

```
4457#1MDNE000,E210B2B1F67CE7F0F22434D315EA9361BE734B0079844FAF,38
1990A89D696C75#A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4Y5
Z6A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X#4316C2C1#>
```

The Network Security Processor returns the following response:

```
<CB#7BB41A6FAA3BF848#Y#1DE3#Y#>
```

Translate PIN and Generate MAC (Command BD)

Command BD translates PINs from encryption under one key to encryption under another and generates a Cipher Block Chaining Message Authentication Code (CBC-MAC). The translated PIN block cryptogram can be included in the data used in the MAC generation process. The incoming PIN Encryption Key is designated as KPE_I, and the outgoing PIN Encryption Key is designated as KPE_O.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

Command

```
<BD#[PIN Block Type]#[Header,EMFK.E(KPEI),MAC]#
[Header,EMFK.E(KPEO),MAC]#[EKPEI(PIN Block)]#
Header,EMFK.E(KMAC),MAC#MAC Length#[Header,EMFK.E(IV),MAC]#
[Insertion Position]#[Insertion Type]#Data Type#Data Length#Data#
[Reserved]#[PIN Block Data]#[MAC Type#]>
```

Response

```
<CD#[EKPEO(ANSI PIN block)]#[Sanity Check Indicator]#
[IBM 3624 Sequence Number]#MAC Length#
MAC or Header,EMFK.E(Ending IV),MAC#KMAC Check Digits#
[Incoming PIN Encryption Key Check Digits]#
[Outgoing PIN Encryption Key Check Digits]#>[CRLF]
```

Calling Parameters

`BD`

Field 0, the command identifier.

`[PIN Block Type]`

Field 1, the incoming PIN block type. If this field is empty, only the MAC generation operation will be performed and fields 2, 3, 4, 8, 9 and 14 must also be empty. This field contains a 1 byte decimal value which can be 1 through 5 or 9, or is empty. When option 46 is enabled, this field can only contain the value 1 (ANSI) or be empty.

The following table identifies the value for each PIN block type.

Value	PIN Block Type
1	ANSI
2	IBM 3624
3	PIN/pad character

Value	PIN Block Type
3	Docutel
4	IBM encrypting PIN pad
5	Burroughs
9	IBM 4731

[Header,$E_{MFK.E}$(KPE$_I$),MAC]

Field 2, the incoming PIN Encryption Key encrypted under the MFK. If this field is empty, only the MAC generation operation will be performed and fields 1, 3, 4, 8, 9 and 14 must also be empty. This field contains a 74 byte value, or volatile table location, or is empty. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

[Header,$E_{MFK.E}$(KPE$_O$),MAC]

Field 3, the outgoing PIN Encryption Key encrypted under the MFK. If this field is empty, only the MAC generation operation will be performed and fields 1, 2, 4, 8, 9 and 14 must also be empty. This field contains a 74 byte value, a volatile table location, or is empty. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. When option 49 is enabled, the length of the KPEo must be equal to or greater than the length of the KPEi (field 2). The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000.

[E_{KPE_I}(PIN Block)]

Field 4, the incoming PIN block encrypted under the incoming PIN Encryption Key. If this field is empty, only the MAC generation operation will be performed and fields 1, 2, 3, 8, 9 and 14 must also be empty. This field contains a 16 byte hexadecimal value, or is empty.

Header,$E_{MFK.E}$(KMAC),MAC

Field 5, the Message Authentication Code Key encrypted under the MFK. This field contains a 74 byte value, or volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1MDNE000, 1MDNN000, 1MDGE000, and 1MDGN000.

`MAC Length`

Field 6, the size of the Message Authentication Code to be generated. The following table indicates the possible MAC sizes and the value to enter in this field for each one. If this field is set to zero, field 4 must be empty.

Value	Returned-MAC Size
0	More data expected; no MAC verified
1	32 bits

Value	**Returned-MAC Size**
2	48 bits
3	64 bits

A 32-bit Message Authentication Code is expressed as eight hexadecimal digits (0-9, A-F) and written as two groups of four digits, separated by a space. A 48- or 64-bit Message Authentication Code is expressed as three or four groups of four hexadecimal digits, separated by a space.

[Header,$E_{MFK.E}$(IV),MAC]

Field 7, the Initialization Vector encrypted under the MFK. If this command contains the first block of multiple blocks of data, or if you will be authenticating only one block of data, this field must be empty; the Network Security Processor will use its default Initialization Vector of all zeros. If this command contains data subsequent to the first block in a multi-block series, this field should contain the ending Initialization Vector from the previously sent data block. This field contains a 74 byte value, a volatile table location, or is empty. The following headers are supported: 1IDNE000 and 1IDNN000.

`[Insertion Position]`

Field 8, the number indicates where the translated PIN block is inserted into the data in this command for MAC generation. If you will not be including the translated PIN block in the MAC generation, set this field to 0. The number *n* means inserting the PIN block in between binary data position *n*-1 and *n.* For unpacked data, the same rule applies, but two unpacked characters are considered one binary data. This field can also contain character 'F' which indicates the first location in the data and 'L' indicates the last location in the data. If the data type is Unpacked ('U'), this field must contain an even number. This field must be empty when fields 1, 2, 3, 4, 9 and 14 are empty.

Example:

31 32 33 34 35 36	input data in unpacked
12 34 56	input data in binary
31 32^33 34 35 36	position 2 in unpacked data
12^34 56	position 2 in binary data
^31 32 33 34 35 36	position F
12 34 56 12 34 56^	position L

^ denotes where the encrypted PIN block goes. The PIN block is converted to the type specified in Field 9 before including into the data to be MACed.

`[Insertion Type]`

Field 9, the PIN block insertion type (A, B, E).

- A – PIN block will be converted to ASCII (unpacked) hex before including in the MACed data at the position indicated in Field 8.

- B – PIN block will be converted to binary before including in the MACed data at the position indicated in Field 8.
- E – PIN block will be converted to EBCDIC before including in the MACed data at the position indicated in Field 8.

These conversions will take place regardless of the value indicated in Field 10.

This field must be empty when fields 1, 2, 3, 4, 8 and 14 are empty.

`Data Type`

Field 10, the data type. The types of data that the Network Security Processor supports are:

Value	Data Type
U	Unpacked ASCII hexadecimal
B	Binary

`Data Length`

Field 11, the data's length. This command will authenticate up to 4096 bytes of data. If more data is being sent in the next command – indicated by Field 6 being set to zero – then the data length must be a multiple of eight. If no more data is being sent, the Network Security Processor will right -pad the data field with binary zeros (nulls) such that the resulting length will be a multiple of eight. This field contains a 1 to 4 byte decimal value. When the data type (field 10) is set to "U", the data length must be an even number.

`Data`

Field 12, the input data. This field can be from one to 4096 bytes long and in binary or unpacked ASCII hexadecimal format. If the data is in unpacked ASCII hexadecimal format, this field can contain the numbers 0 through 9 and the characters A through F.

`Reserved`

Field 13, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor.

`[PIN Block Data]`

Field 14, the PIN block data. Its contents depend on the PIN block type used. See PIN Block Types. If this field is empty, it indicates that only MAC generation will be performed and fields 1, 2, 3, 4, 8 and 9 must be empty.

`[MAC Type#]`

Field n, the algorithm used to generate the CBC-MAC. This field is optional. If it is not present, the ISO-9797-1 MAC algorithm will be used to generate the CBC-MAC. If this field is present, it is the last field in the command. The following table shows the supported CBC-MAC types and the numerical value to enter in this field for each CBC-MAC type.

Value	MAC Type
Empty, or 1-6	ISO - 9797-1 Algorithm 1 - 1key-, 2key-, or 3key- 3DES Cipher Block Chaining.
7	ISO 9797-1 Algorithm 3 - Only the last data block is processed using 3DES-Cipher Block Chaining, all previous data blocks are processed using 1key DES-Cipher Block Chaining.

Table 4-89 Command BD: Translate PIN and Generate ATM MAC

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	BD
1	[PIN block type]	0, 1	1-5, 9
2	[Header,$E_{MFK.E}$(KPE$_I$),MAC]*	0, 74	printable ASCII
3	[Header,$E_{MFK.E}$(KPE$_O$),MAC]*	0, 74	printable ASCII
4	[E_{KPE_I}(PIN Block)]	0, 16, 18	0 - 9, A - F
5	Header,$E_{MFK.E}$(KMAC),MAC*	74	printable ASCII
6	MAC Length	1	0 - 3
7	[Header,$E_{MFK.E}$(IV),MAC]	0, 74	printable ASCII
8	[Insertion Position]	0 - 4	0 - 9, L, F
9	[Insertion Type]	0, 1	A, B, E
10	Data Type	1	U or B
11	Data Length	1 - 4	0 - 9
12	Data	1 - 4096	0 - 9, A - F if unpacked ASCII
13	[Reserved]	0	empty
14	[PIN Block data]**	Variable	
n	[MAC Type#]	0, 1	1-7

* Can be a volatile table location.

** See PIN Block Types for information on PIN block data.

Responding Parameters

`CD`

Field 0, the response identifier.

`[E`$_{KPE_O}$`(ANSI PIN block)]`

Field 1, the outgoing PIN block encrypted under the PIN Encryption Key. This field contains 16 hexadecimal characters. This field is empty if no PIN translation operation is performed.

`[Sanity Check Indicator]`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

note This field is empty if no PIN translation operation is performed.

`[IBM 3624 Sequence Number]`

Field 3, the IBM 3624 sequence number. When the PIN block type is IBM 3624, this field contains 2 hexadecimal characters, otherwise it is empty.

`MAC Length`

Field 4, the length of the Message Authentication Code. This field contains the value of field 6 supplied in the command.

`MAC or Header,E`$_{MFK.E}$`(Ending IV),MAC`

Field 5, If Field 4 is set to 0, this field will contain the ending Initialization Vector encrypted under the MFK. If Field 4 is set to 1, 2, or 3, this field will contain the Message Authentication Code. If your use of this command results in the generation of an ending Initialization Vector in this field, use it as the starting initialization vector in the subsequent Message Authentication Code command to continue generating Message Authentication Code. This field contains a 9, 14, 16, 19, or 74 byte value that contains hexadecimal values or spaces.

`KMAC Check Digits`

Field 6, check digits; the first four digits that result from encrypting zeros using the

Message Authentication Code key. If option 88 is enabled, this field will contain six check digits.

`[Incoming PIN Encryption Key Check Digits]`

Field 7, check digits; the first four digits that result from encrypting zeros using the incoming PIN Encryption Key. If option 88 is enabled, this field will contain six check digits. This field is empty if a sanity error is returned, or no PIN translation operation is performed.

`[Outgoing PIN Encryption Key Check Digits]`

Field 8, check digits; the first four digits that result from encrypting zeros using the outgoing PIN Encryption Key. If option 88 is enabled, this field will contain six check digits. This field is empty if a sanity error is returned, or no PIN translation operation is performed.

Table 4-90 Response CD: Translate PIN and Generate ATM MAC

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	CD
1	[E$_{KPE_O}$(PIN block)]	0, 16	0 - 9, A - F
2	[Sanity check indicator]	0, 1	Y, N, L
3	[IBM 3624 Sequence Number]	0, 2	0 - 9
4	MAC Length	1	0 - 3
5	MAC or Header,E$_{MFK.E}$(Ending IV), MAC	9, 14, 16, 19, 74	0 - 9, A - F, “ ”, “,” Printable ASCII
6	KMAC Check Digits	4 or 6	0 - 9, A - F
7	[Incoming PIN Encryption Key Check Digits]	0, 4, or 6	0 - 9, A - F
8	[Outgoing PIN Encryption Key Check Digits]	0, 4, or 6	0 - 9, A - F

Usage Notes

- Generate the incoming and outgoing PIN Encryption Keys.
- Generate the Message Authentication Code Key.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear text ANSI PIN block: 0412 26CB A987 6FED

- Clear-text incoming PIN Encryption Key: 07CE A74F 4607 5D8F
  The incoming PIN Encryption Key in AKB format:
  1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,C47517937 1FDBE43

- Clear-text outgoing PIN Encryption Key: D029 23D9 AD4F E90B
  The outgoing PIN Encryption Key in AKB format:
  1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53AF657A4508,7ED6957 0CDE54F30

- Encrypted ANSI PIN block: 9AA4 3B94 C012 04F3

- Clear-text Message Authentication Code Key: 8FF4 98F1 B661 5151
  The Message Authentication Code Key in AKB format:
  1MDNE000,E210B2B1F67CE7F0F22434D315EA9361BE734B0079844FAF,381990A89 D696C75

- Insertion Position: 2

- Insertion Type: A

- Data Type: U

- Data: 595A313233343536

  If the insertion type is 'A', the encrypted PIN block is converted to ASCII (each byte in the cryptogram is converted into two ASCII characters) before being inserted into the data. In these examples the outgoing encrypted PIN block is EBA0BA4387C3771A. The data to be used in the MAC generation follows, the inserted outgoing encrypted PIN block converted to ASCII is underlined.

  Data to be MACed: 59454241304241343338374333373731415A313233343536

MAC generated using ISO 9797-1 Algorithm 1

The command looks like this:

```
<BD#1#1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,C
475179371FDBE43#1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53AF
657A4508,7ED69570CDE54F30#9AA43B94C01204F3#1MDNE000,E210B2B1F67CE
7F0F22434D315EA9361BE734B0079844FAF,381990A89D696C75#1##2#A#U#16#
595A313233343536##123456789012#>
```

The Network Security Processor returns the following response:

```
<CD#EBA0BA4387C3771A#Y##1#DCEC 636A#1DE3#DCFF#9F5E#>
```

MAC generated using ISO 9797-1 Algorithm 3

- Clear-text Message Authentication Code Key: 0123 4567 89AB CDEF FEDC BA98 7654 3210
  The Message Authentication Code Key in AKB format:
  1MDNE000,FF4577DFBBBADDAD9CF2CE06838DA93D2EE687AF7380444F,835992A675B58556

The command looks like this:

```
<BD#1#1PUNE000,24E42C54BA2DD3F1781BBBB84298B1193BD0CDFCF431B0C0,C
475179371FDBE43#1PUNE000,9BA4CBE61A4247783D4DED9A7E2FAD879CAC53AF
657A4508,7ED69570CDE54F30#9AA43B94C01204F3#1MDNE000,FF4577DFBBBAD
DAD9CF2CE06838DA93D2EE687AF7380444F,835992A675B58556#1##2#A#U#16#
595A313233343536##123456789012#7#>
```

The Network Security Processor returns the following response:

```
<CD#EBA0BA4387C3771A#Y##1#80BB E6AA#08D7#DCFF#9F5E#>
```

Verify Clear PIN (Command D0)

Command D0 verifies a clear-text PIN according to the technique you specify when you issue the command.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<D0#Verification Method#0#PIN#Reserved#PIN Information#>
```

Response

```
<E0#Verification Flag#>[CRLF]
```

Calling Parameters

`D0`

Field 0, the command identifier.

`Verification Method`

Field 1, the PIN verification method. The following table identifies the PIN verification methods that this command supports.

Value	Verification Method
1	Identikey
2	IBM 3624
3	VISA

`0`

Field 2, an indicator that this command is verifying a clear-text PIN.

`PIN`

Field 3, the clear-text PIN. This field contains a 4 to 12 byte decimal value.

`Reserved`

Field 4, reserved for future use. This field is ignored, and must be empty.

`PIN Information`

Field 5, identical to the fields from Command 32 starting at Field 5 and continuing until the second-to-last field.

Table 4-91 Command D0: Verify Clear PIN

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	D0
1	Verification method	1	1 - 3
2	Clear-text PIN indicator	1	0
3	PIN	4 - 12	0 - 9
4	Reserved	0	
5	PIN information*		

* See Command 32 for parameter information.

Responding Parameters

`E0`

Field 0, the response identifier.

`Verification Flag`

Field 1, the verification flag. This field returns Y if the PIN block is successfully verified or N if the PIN block is not successfully verified.

Table 4-92 Response E0: Verify Clear PIN

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	E0
1	Verification flag	1	Y, N

Usage Notes

- This command is typically used for verifying host-based PINs in a proprietary network and for verifying PINs in on-others transactions initiated in a shared network.
- This command does not check the PIN to be sure its length is legal.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify a clear-text PIN using the IBM 3624 method

- Verification method: IBM 3624 (2)
- PIN: 3614 3614 3
- PIN information:.
  - Conversion table: 8351 2964 7746 1538
    The conversion table in AKB format:
    1nCNE000,2162CD77E8293FE4DC328EAB53BC3A2B0A3AFE1B299F07D2,111746BEB588C65B
  - Offset: 6694 537
  - Validation data: 3333 3333
  - Pad character: D
  - Check-length parameter: 7
  - Clear-text PIN Verification Key: 89B0 7B35 A1B3 F47E
    The PIN Verification Key in AKB format:
    1V3NE000,85F6B20112A3BCCCECBEB2BB9F050B788524A06CD21CFCC8,2442B579937AF5BB

The command looks like this:

```
<D0#2#0#361436143##1nCNE000,2162CD77E8293FE4DC328EAB53BC3A2B0A3AF
E1B299F07D2,111746BEB588C65B#6694537#33333333#D#7#1V3NE000,85F6B2
0112A3BCCCECBEB2BB9F050B788524A06CD21CFCC8,2442B579937AF5BB#>
```

The Network Security Processor returns the following response:

```
<E0#Y#>
```

Generate Atalla 2x2 PVN (Command 11E)

Command 11E generates a PIN Verification Number using the Atalla 2x2 method.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<11E#I#Header,EMFK.E(KPV),MAC#Reserved#PVN Format#PVN Length#
Data Type#Data Length#Data#>
```

Response

```
<21E#PVN#>[CRLF]
```

Calling Parameters

`11E`

Field 0, the command identifier.

`I`

Field 1, the Atalla 2x2 algorithm identifier. This field contains the letter I.

`Header,E`$_{MFK.E}$`(KPV),MAC`

Field 2, the PIN Verification Key pair encrypted under the MFK. This field contains a 74 byte value or a volatile table location. This key must be either a 2key- or 3key-3DES key. When option 4A is enabled, this key can be a 1key -3DES (single-length) key. The following headers are supported: 1VaNE000, 1VaNN000, 1VaGE000, and 1VaGN000.

`Reserved`

Field 3, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

`PVN Format`

Field 4, this field specifies the format of the PVN. The choices are H for hexadecimal or D for decimal. For decimal format this field should contain the letter D, followed by the 16 byte decimalization table. If you use the default decimalization table of 0123456789012345, this field should contain only the letter D.

`PVN Length`

Field 5, defines the length of the generated PVN. This field contains a 1 to 2 byte decimal value in the range of 6 to 16.

`Data Type`

Field 6, the data type. The types of data that the Network Security Processor supports are:

Value	Data Type
U	Unpacked ASCII hexadecimal
B	Binary

`Data Length`

Field 7 defines the length of the data. This field contains a 2 byte decimal value in the range of 16 -24.

`Data`

Field 8, is the clear-text PIN and the 12 account number digits. This field contains 16 to 24 bytes decimal value.

Table 4-93 Command 11E: Generate Atalla 2x2 PVN

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	11E
1	Atalla 2x2 method	1	I
2	Header,$E_{MFK.E}$(KPV),MAC	74	printable ASCII
3	Reserved	0, 1, 2	0 - 9, A - F
4	PVN Format	1, 17	H, D, 0 - 9
5	PVN Length	1, 2	6 - 16
6	Data Type	1	B, U
7	Data Length	2	16 - 24
8	Data	varies	0 - 9

Responding Parameters

`21E`

Field 0, the response identifier.

`PVN`

Field 1, the generated PVN. This field contains a 6 to 16 byte hexadecimal value.

Table 4-94 Response 21E: Generate Atalla 2x2 PVN

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	21E
1	PVN	6-16	0 - 9, A - F

Usage Notes

- Generate the PIN Verification Keys.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Generate a PVN using the Atalla 2x2 method

- PVN method: Atalla 2x2 (I)
- Clear-text PIN Verification Key pair:
  5555 6666 7777 8888 9999 AAAA BBBB CCCC
  The PIN Verification Key pair in AKB format:
  1VaNE000,A82F67C4AEF8EDB3A8851D83181497E82137C8836060C72F,1C6D2D968
  A0A6F7D
- PVN Format: Hexadecimal
- PVN Length: 16
- Data Type: Unpacked ASCII hexadecimal
- Data Length: 18
- Data: PIN = 555555, Account Number 1234 1234 1234

The command looks like this:

```
<11E#I#1VaNE000,A82F67C4AEF8EDB3A8851D83181497E82137C8836060C72F,
1C6D2D968A0A6F7D##H#16#U#18#555555123412341234#>
```

The Network Security Processor returns the following response:

```
<21E#3436593F00F3C754#>
```

Calculate PIN Offset (Command 30A)

Command 30A uses the old account number and offset to determine the PIN. It then uses the PIN and the new account number to calculate the new IBM 3624 offset.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

<30A#Header,$E_{MFK.E}$(KPV$_{OLD}$),MAC#Header,$E_{MFK.E}$(KPV$_{NEW}$),MAC#
Validation Data 1#Validation Data 2#[Old Conversion Table#
New Conversion Table#]Old PIN Offset#>

Response

<40A#New Offset#KPV$_{OLD}$ Check Digits#KPV$_{NEW}$ Check Digits#>[CRLF]

Calling Parameters

`30A`

Field 0, the command identifier.

Header,$E_{MFK.E}$(KPV$_{OLD}$),MAC

Field 1, the PIN Verification Key encrypted under the MFK. This key is used to generate the old offset. This field can contain a 74 byte value, or be a volatile table location. When option 4A is enabled, this key can be a 1key -3DES (single-length) key. The following headers are supported: 1V3NE000, 1V3NN000, 1V3VE000, and 1V3VN000.

Header,$E_{MFK.E}$(KPV$_{NEW}$),MAC

Field 2, the PIN Verification Key encrypted under the MFK. This key is used to generate the new offset. This field can contain a 74 byte value, or be a volatile table location. If the KPV$_{OLD}$ is a 2key-3DES (double-length) key, this key must also be a 2key-3DES key. If the KPV$_{OLD}$ is a 1key-3DES (single-length) key, this key can be either a 1key or 2key-3DES key. When option 4A is enabled, this key can be a 1key -3DES (single-length) key. The following headers are supported: 1V3NE000, 1V3NN000, 1V3GE000, and 1V3GN000.

`Old Validation Data`

Field 3, the old validation data consists of the old account number digits used to generate the old offset. If less than 16 account number digits were used to generate the old offset, this field must also contain the pad characters. This field must contain a 16 byte hexadecimal value.

`New Validation Data`

Field 4, the new validation data consists of the new account number digits. If less than 16 account number digits will be used to generate the new offset, this field must also contain the pad characters. This field must contain a 16 byte hexadecimal value.

`[Old Conversion Table#`

Field 5, the old Conversion Table is used to generate the old offset. This field is optional and is only required if the old Conversion Table is not 0123456789012345. If the new conversion table is not 0123456789012345, it must be supplied in field 6 and this field must exist but can be empty. If present, this field must be a 16 byte decimal value, a volatile table location, or an empty field if the old conversion table is 0123456789012345.

When option 48 is enabled, this field contains the conversion table in AKB format (the header must be 1nCNE000) or a volatile table location. When option 4E is enabled, and all three forms of the conversion table (clear-text, decrypted, or value stored in volatile table location) must adhere to these rules:

- The conversion table must have at least eight unique digits.
- No single digit can occur more than four times.

`New Conversion Table#]`

Field 6, the new Conversion Table is used to generate the new offset. This field is optional and is only required if the new Conversion Table is not 0123456789012345 or if field 5 is provided in the command. If present, this field must be a 16 byte decimal value, a volatile table location, or an empty field if the new conversion table is 0123456789012345.

When option 48 is enabled, this field contains the conversion table in AKB format (the header must be 1nCNE000) or a volatile table location. When option 4E is enabled, all three forms of the conversion table (clear-text, decrypted, or value stored in volatile table location) must adhere to these rules:

- The conversion table must have at least eight unique digits.
- No single digit can occur more than four times.

`Old Offset`

Field 7, the old offset. This field must contain a 4 to 12 byte decimal value.

Table 4-95 Command 30A: Calculate PIN Offset

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	30A
1	Header,E$_{MFK.E}$(KPV$_{OLD}$),MAC*	74	printable ASCII

Table 4-95 Command 30A: Calculate PIN Offset (continued)

Field	Contents	Length (bytes)	Legal Characters
2	Header,$E_{MFK.E}(KPV_{NEW})$,MAC*	74	printable ASCII
3	Old Validation Data	16	0 - 9, A - F
4	New Validation Data	16	0 - 9, A - F
5	[Old Conversion Table#	0, 16*	0 - 9
6	New Conversion Table#]	0,16*	0 - 9
7	Old Offset	4 - 12	0 - 9

* Can be a volatile table location.

Responding Parameters

`40A`

Field 0, the response identifier.

`New Offset`

Field 1, the offset based on the PIN, new validation data, new Conversion Table, and new PIN Verification Key.

`KPV`$_{OLD}$ `Check Digits`

Field 2, check digits; the first four digits that result from encrypting zeros using the KPV_{OLD}. If the old and new conversion table fields (i.e. fields 5 and 6) are present in the command, this field will contain six check digits. If option 88 is enabled, this field will contain six check digits.

`KPV`$_{NEW}$ `Check Digits`

Field 3, check digits; the first four digits that result from encrypting zeros using the KPV_{NEW}. If the old and new conversion table fields (i.e. fields 5 and 6) are present in the command, this field will contain six check digits. If option 88 is enabled, this field will contain six check digits.

Table 4-96 Response 40A: Calculate PIN Offset

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	40A
1	New Offset	4 - 12	0 - 9

Table 4-96 Response 40A: Calculate PIN Offset (continued)

Field	Contents	Length (bytes)	Legal Characters
2	KPV_{OLD} Check Digits	4 or 6	0 - 9, A - F
3	KPV_{NEW} Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- Generate the KPV_{OLD} and KPV_{NEW} AKBs.
- Fields 5 and 6 are a pair, they either both do not exist and therefore the old and new conversion table will be 0123456789012345, or they both exist. They must both exist in these two scenarios:
  - If the old conversion table is a value other than 0123456789012345, field 5 will contain the value of the conversion table and field 6 must also exist. If the new conversion table is 0123456789012345, field 6 can be empty. If the new conversion table is not 0123456789012345, it must be provided in field 6.
  - Similarly if the old conversion table is 0123456789012345 and the new conversion table is a different value, field 5 must exist, but can be empty, and field 6 will contain the new conversion table.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Generate a new offset

- Clear-text old PIN Verification Key: 1234123412341234. Check digits are C2F2
  The old PIN Verification Key in AKB format:
  1V3NE000,F3CED719DC65AC7F9B2C232ECCA08F439A5358EF0279BB71,2D962560C4B67F6B
- Clear-text new PIN Verification Key: 4321432143214321. Check digits are 8149. The new PIN Verification Key in AKB format:
  1V3NE000,DF7DC3D37F898C6E86E179C2B853ED90CC7DD4307762E8DE,0A93C1005D0AED21
- The old validation data is 0123 4567 89FF FFFF
- The new validation data is 9876 5432 10FF FFFF
- The old and new Conversion Table is 0123 4567 8901 2345

- The old offset is 9920, (the clear-text PIN is 1234)

The command looks like this:

```
<30A#1V3NE000,F3CED719DC65AC7F9B2C232ECCA08F439A5358EF0279BB71,2D96
2560C4B67F6B#1V3NE000,DF7DC3D37F898C6E86E179C2B853ED90CC7DD4307762E
8DE,
0A93C1005D0AED21#0123456789FFFFFF#9876543210FFFFFF#9920#>
```

The Network Security Processor returns the following response:

```
<40A#1313#C2F2#8149#>
```

Use default old conversion table and a different new conversion table

Same data as previous example except that new conversion table is 9876543210543210

The command looks like this:

```
<30A#1V3NE000,F3CED719DC65AC7F9B2C232ECCA08F439A5358EF0279BB71,2D962
560C4B67F6B#1V3NE000,DF7DC3D37F898C6E86E179C2B853ED90CC7DD4307762E8D
E,0A93C1005D0AED21#0123456789FFFFFF#9876543210FFFFFF##98765432105432
10#9920#>
```

The Network Security Processor returns the following response:

```
<40A#6200#C2F22E#814910#>
```

Verify ePIN (Command 32C)

Command 32C is used to verify the ePIN. This command uses a form of the IBM 3624 algorithm to verify the ePIN.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To enable this command, you must purchase this command in the form of a command 105. To use this command, you must add it to the Network Security Processor's security policy.

This command requires the Object PIN Key (KOP) and the PIN Verification Key (KPV) to be either a 2key- or a 3key-3DES key.

Command

```
<32C#Offset Format#Header,EMFK.E(KPV),MAC#Header,EMFK.E(KOP),MAC#
ePIN#ePIN Object#>
```

Response

```
<42C#Verification Flag#KPV Check Digits#KOP Check Digits#>[CRLF]
```

Calling Parameters

`32C`

Field 0, the command identifier.

`Offset Format`

Field 1, the offset format must be 2. This field contains 1 byte, the decimal value 2.

Header,$E_{MFK.E}$(KPV),MAC

Field 2, the PIN Verification Key encrypted under the MFK. This key is used to generate the ePIN offset. This field must contain a 74 byte value. This key must be either a 2key- or
3key-3DES key. The following headers are supported: 1V3NE000, 1V3NN000, 1V3VE000, and 1V3VN000.

Header,$E_{MFK.E}$(KOP),MAC

Field 3, the Object PIN Key encrypted under the MFK. This key is used to decrypt the ePIN object. This field must contain a 74 byte value. This key must be either a 2key- or 3key-3DES key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

`ePIN`

Field 4, the entered ePIN. This field must contain a 16 byte hexadecimal value.

`ePIN Object`

Field 5, the ePIN Object. This field must contain a 32 byte hexadecimal value.

Table 4-97 Command 32C: Verify ePIN Offset

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	32C
1	Offset Format	1	2
2	Header,$E_{MFK.E}$(KPV),MAC	74	printable ASCII
3	Header,$E_{MFK.E}$(KOP),MAC	74	printable ASCII
4	ePIN	16	0 - 9, A - F
5	ePIN Object	32	0 - 9, A - F

Responding Parameters

`42C`

Field 0, the response identifier.

`Verification Flag`

Field 1, the verification flag. This field will contain either the letter Y if the ePIN verifies, or the letter N if the ePIN does not verify.

`KPV Check Digits`

Field 2, check digits; the first four digits that result from encrypting zeros using the PIN Verification Key. If option 88 is enabled, this field will contain six check digits.

`KOP Check Digits`

Field 3, check digits; the first four digits that result from encrypting zeros using the Object PIN Key. If option 88 is enabled, this field will contain six check digits.

Table 4-98 Response 42C: Verify ePIN Offset

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	42C
1	Verification Flag	1	Y, or N
2	KPV Check Digits	4 or 6	0 - 9, A - F

Table 4-98 Response 42C: Verify ePIN Offset (continued)

Field	Contents	Length (bytes)	Legal Characters
3	KOP Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- Generate the PIN Verification Key and the Object PIN Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify an ePIN

- Clear-text PIN Verification Key: 4321432143214321 1234123412341234. Check digits 2ABA. The PIN Verification Key in AKB format: 1V3NE000,DF7DC3D37F898C6E1916FC4AEEE7849E8300F2638934F1F4,7EFCD1B1C6D893DC

- Clear-text Object PIN Key: 5678567856785678 8765876587658765. Check digits 686F. The Object PIN Key in AKB format: 1PUNE000,8586FE8ECDA3D4EE40FD86769D5B6EF3DE0847DA7A40455B,90DAE3B79A2E73F6

- ePIN: 314A41434B2A2A2A

- ePIN Object: 27BDDE807F87DDD4589226D1F475CD0E

The command looks like this:

```
<32C#2#1V3NE000,DF7DC3D37F898C6E1916FC4AEEE7849E8300F2638934F1F4,7EFC
D1B1C6D893DC#1PUNE000,8586FE8ECDA3D4EE40FD86769D5B6EF3DE0847DA7A40455
B,90DAE3B79A2E73F6#314A41434B2A2A2A#27BDDE807F87DDD4589226D1F475CD0E#
>
```

The Network Security Processor returns the following response:

```
<42C#Y#2ABA#686F#>
```

PIN and PIN-Block Translate (Command 335)

Command 335 translates an encrypted PIN from encryption under an incoming PIN encryption key to encryption under an outgoing PIN encryption key. Another use of Command 335 is translation from one PIN block type to a different block type. The incoming PIN Encryption Key is designated as KPE_I, and the outgoing PIN Encryption Key is designated as KPE_O. The incoming Communications Key is designated as KC_I, and the outgoing Communications Key is designated as KC_O.

In version 1.42 and above, ISO-1 and ISO-2 PIN blocks are supported.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<335#Reserved#Incoming PIN Block Type#Reserved#
Outgoing PIN Block Type#Header,EMFK.E(KPEI),MAC#
Header,EMFK.E(KPEO),MAC#EKPEI(PIN Block)#Incoming PIN Block Data#
Outgoing PIN Block Data#>
```

Response

```
<435#KPEO(Outgoing PIN Block)#Sanity Check Indicator#
KPEI Check Digits#KPEO Check Digits#KCI Check Digits#
KCO Check Digits#>[CRLF]
```

Calling Parameters

`335`

Field 0, the command identifier.

`Reserved`

Field 1, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor. This field should be empty.

`Incoming PIN Block Type`

Field 2, the type of the incoming PIN block. This field is 1 byte, it can contain the numbers 1, 3, 8 or 9. When option 46 is enabled, this field can contain the value 1 (ANSI) or 8 (ISO-3).

Value	PIN Block Type
1	ANSI (ISO-0)
9	IBM 4731
C	ISO-1

Value	PIN Block Type
6	ISO-2
8	ISO-3
3	PIN/pad character / Docutel

`Reserved`

Field 3, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor. This field should be empty.

`Outgoing PIN Block Type`

Field 4, the type of the incoming PIN block. This field is 1 byte, it can contain the numbers 1, 3, 8 or 9. When option 47 is enabled, this field can contain the value 1 (ANSI) or 8 (ISO-3).

Value	PIN Block Type
1	ANSI (ISO-0)
9	IBM 4731
C	ISO-1 (Allowed only when field 2 is 3, 6, 9, or C.)
8	ISO-3
3	PIN/pad character / Docutel

Header, $E_{MFK.E}(KPE_I)$, MAC

Field 5, the incoming PIN Encryption Key (KPE_I) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, and 1PUDN000.

Header, $E_{MFK.E}(KPE_O)$, MAC

Field 6, the outgoing PIN Encryption Key (KPE_O) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. This field cannot contain a 1key-3DES (single-length) key. When option 49 is enabled, the length of the KPEo must be equal to or greater than the length of the KPEi (field 2). The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000.

E_{KPEI}(PIN Block)

Field 7, the incoming PIN Block encrypted under the incoming PIN Encryption Key.

`Incoming PIN Block Data`

Field 8, Incoming PIN Block Data.

If the incoming PIN Block type is ANSI or ISO-3, this field will contain twelve bytes; the incoming PAN digits. When any of these options 46, 47or 6B are enabled, an error is returned if the value in this field a does not match the outgoing PIN block data. When option 6B is enabled, an error is returned if this field contains all zeros.

If the incoming PIN Block type is ISO-1 or ISO-2, this field must be empty.

If the incoming PIN Block type is PIN Pad, this field will contain a one byte value; the pad character. Valid pad characters are a hexadecimal value, W, or X.

If the incoming PIN Block type is IBM 4731, this field will contain three fields:

- A one byte value; the pad character. Valid pad characters are a hexadecimal value, W, or X.
- The incoming ICV; a 16 byte hexadecimal value. When option 6B is enabled, an error is returned if this field contains all zeros or does not match the outgoing ICV.
- The incoming Communications Key (KC-I) encrypted under the MFK. This field contains a 74 byte value. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1c7NE000, 1c7NN000, 1c7DE000, and 1c7DN000.

`Outgoing PIN Block Data`

Field 9, Outgoing PIN Block Data.

If the outgoing PIN Block type is **ANSI or ISO-3**, this field will contain the twelve bytes; the outgoing PAN digits. When option 6B is enabled, an error is returned if this field contains all zeros.

If the outgoing PIN Block type is **ISO-1**, this field must be empty.

If the outgoing PIN Block type is **PIN Pad**, this field will contain one byte; the pad character. Valid pad characters are a hexadecimal value, W, or X.

If the outgoing PIN Block type is **IBM 4731**, this field will contain three fields:

- A one byte value; the pad character. Valid pad characters are a hexadecimal value, W, or X.
- The outgoing ICV; a 16 byte hexadecimal value. When option 6B is enabled, an error is returned if this field contains all zeros.
- The outgoing Communications Key (KC_O) encrypted under the MFK. This field contains a 74 byte value. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. The following headers are supported: 1c7NE000, 1c7NN000, 1c7EE000, and 1c7EN000.

Table 4-99 Command 335: PIN and PIN-Block Translate

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	335
1	Reserved	0, 1, 2	0 - 31
2	Incoming PIN Block Type	1	1, 3, 6, 8, 9, C
3	Reserved	0, 1, 2	0 - 9
4	Outgoing PIN Block Type	1	1, 3, 8, 9, C
5	Header,$E_{MFK.E}$(KPE$_I$),MAC*	74	printable ASCII
6	Header,$E_{MFK.E}$(KPE$_O$),MAC*	74	printable ASCII
7	E_{KPEI}(PIN Block)	16	0 - 9, A - F
8	Incoming PIN Block Data ANSI / ISO-3 or Incoming PIN Block Data ISO-1 or ISO-2 or Incoming PIN Block Data PIN Pad or Incoming PIN Block Data IBM 4731 - Incoming Pad - Field separator - Incoming ICV - Field separator - Header,$E_{MFK.E}$(KC$_I$),MAC	12 0 1 1 1 16 1 74	0 - 9 none 0 9, A- F, W, X 0 - 9, A- F, W, X # 0 - 9, A - F # printable ASCII
9	Outgoing PIN Block Data ANSI / ISO-3 or Outgoing PIN Block Data ISO-1 or Outgoing PIN Block Data PIN Pad or Outgoing PIN Block Data IBM 4731 - Outgoing Pad - Field separator - Outgoing ICV - Field separator - Header,$E_{MFK.E}$(KC$_O$),MAC	12 0 1 1 1 16 1 74	0 - 9 none 0 9, A- F, W, X 0 - 9, A- F, W, X # 0 - 9, A - F # printable ASCII

* Can be a volatile table location.

Responding Parameters

```
435
```

Field 0, the response identifier.

^{E}KPE$_{O}$(Outgoing PIN Block)

Field 1, the outgoing PIN block encrypted under KPE$_{O}$. This field contains 16 hexadecimal characters. When a PIN sanity error is detected, the value in this field may not be correct. When a PIN sanity error is detected, and option 4B is enabled, this field will contain 16 zeros.

Sanity Check

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

KPE$_{I}$ Check Digits

Field 3, check digits; the first four digits that result from encrypting zeros using the incoming PIN Encryption Key (KPE$_{I}$). If option 88 is enabled, this field will contain six check digits.

KPE$_{O}$ Check Digits

Field 4, check digits; the first four digits that result from encrypting zeros using the outgoing PIN Encryption Key (KPE$_{O}$). If option 88 is enabled, this field will contain six check digits.

[KC$_{I}$ Check Digits#]

Field 5, check digits; the first four digits that result from encrypting zeros using the incoming Communications Key (KC$_{I}$). If option 88 is enabled, this field will contain six check digits. This field is present only if the incoming PIN Block type is IBM 4731.

[KC$_{O}$ Check Digits#]

Field 6, check digits; the first four digits that result from encrypting zeros using the outgoing Communications Key (KC$_{O}$). If option 88 is enabled, this field will contain six check digits. This field is present only if the outgoing PIN Block type is IBM 4731.

Table 4-100 Response 435: PIN and PIN-Block Translate

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	435
1	E$_{KPEO}$(Outgoing PIN Block)	16	0 - 9, A - F

Table 4-100 Response 435: PIN and PIN-Block Translate (continued)

Field	Contents	Length (bytes)	Legal Characters
2	Sanity Check	1	Y, N, L
3	KPE_I Check Digits	4 or 6	0 - 9, A - F
4	KPE_O Check Digits	4 or 6	0 - 9, A - F
5	[KC_I Check Digits]*	0, 4, or 6	0 - 9, A - F
6	[KC_O Check Digits]*	0, 4, or 6	0 - 9, A - F

* This field exists only when either the incoming/outgoing PIN Block type is IBM 4731.

Usage Notes

- Generate the PIN Encryption Keys.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate an ANSI PIN block

- Incoming PIN Block Type is ANSI; 1
- Outgoing PIN Block Type is ANSI: 1
- Clear-text Incoming PIN Encryption Key: 4567 89AB CDEF 0123 0123 4567 89AB CDEF. Check Digits are F8DF. The Incoming PIN Encryption Key in AKB format: 1PUNE000,52C6960CC400CACD1A14545008A94E8D4A9C4D6838451FF6,29C2C3E2903CCD58
- Clear-text Outgoing PIN Encryption Key: 6789 ABCD EF01 2345 FEDC BA98 7654 3210 Check Digits are 40B5. The Outgoing PIN Encryption Key in AKB format: 1PUNE000,AC09E97F2FFA151C906845AD9A838330004912B2C4A1E8B4,0DEC5A03D7670AF6
- Clear-text incoming ANSI PIN block: 041226CBA9876FED
  The incoming ANSI PIN block encrypted under the incoming PIN Encryption Key: BF8E 1569 561D D33E
- Incoming PAN Digits: 1234 5678 9012
- Outgoing PAN Digits: 1234 5678 9012

The command looks like this:

```
<335##1##1#1PUNE000,52C6960CC400CACD1A14545008A94E8D4A9C4D6838451FF
6,29C2C3E2903CCD58#1PUNE000,AC09E97F2FFA151C906845AD9A838330004912B
2C4A1E8B4,0DEC5A03D7670AF6#BF8E1569561DD33E#123456789012#1234567890
12#>
```

The Network Security Processor returns the following response:

```
<435#53F4660894A37C67#Y#F8DF#40B5###>
```

PIN Translate - DUKPT to 3DES and Verify MAC (Command 346)

Command 346 translates a DUKPT encrypted ANSI PIN block from encryption under an incoming PIN encryption key (KPE_I) to encryption under an outgoing 3DES PIN encryption key (KPE_O), and in addition verifies a MAC on a data field. The KPE_I and the MAC key are derived using the Base Derivation Key and the key serial number. The PIN will be processed only if the MAC verifies.

This command is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

Command

```
<346#Header,EMFK.E(Base Derivation Key),MAC#Header,EMFK.E(KPEO),MAC#
EKPEN(PIN Block)#ANSI PAN Digits#Key Serial Number#MAC Data Length#
MAC Data#[MAC Type]#MAC#Session Key Length#>
```

Response

```
<446#KPEO(Outgoing ANSI PIN Block)#[Sanity Check Indicator]#
MAC Verification Indicator#[KPEI Check Digits]#KMAC Check Digits#>
[CRLF]
```

Calling Parameters

`346`

Field 0, the command identifier.

`Header,EMFK.E(Base Derivation Key),MAC`

Field 1, the Base Derivation Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. If option 6C is enabled, the AKB can contain a 1key-3DES (single-length) key. The following headers are supported: 1dDNE000 and 1dDNN000.

`Header,EMFK.E(KPEO),MAC`

Field 2, the outgoing PIN Encryption Key (KPE_O) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. If the Session Key Length is "D", the AKB must contain a 2key-3DES (double-length) key. If the Session Key Length is "S" and option 6C is enabled, the AKB can contain either a 1key-3DES (single-length) or 2key-3DES (double-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, 1PUEN000, 1PDNE000, 1PDNN000, 1PDEE000, and 1PDEN000.

`EKPEN(PIN Block)`

Field 3, the incoming ANSI PIN Block encrypted under the DUKPT PIN Encryption Key.

`ANSI PAN Digits`

Field 4, the 12 digits used to format the incoming encrypted ANSI PIN block.

`Key Serial Number`

Field 5, the 10 to 20 digit Key Serial Number (KSN) from the PIN entry device. This field contains a 10 to 20 hexadecimal digit value, leading F's will be suppressed.

`MAC Data Length`

Field 6, the number of bytes of data supplied in field 7. The minimum data length is 2, the maximum data length is 4096.

`MAC Data`

Field 7, the data in ASCII hexadecimal format that was used to generate the MAC. This field contains a 2 - 4096 hexadecimal character value. This field must contain an even number of hexadecimal characters.

`[MAC Type]`

Field 8, the type of MAC to be calculated. The possible values for this field are:

Value	MAC Type
Empty, or 1-6	ISO - 9797-1 Algorithm 1 - 1key or 2key-3DES Cipher block chaining.
7	ISO 9797-1 Algorithm 3 - Only the last data block is processed using 3DES, all previous blocks are processed using single DES.
V	Visa DUKPT (old style) as generated by command 5C

`MAC`

Field 9, the MAC to be verified, based on the data supplied in field 7. This field must contain eight hexadecimal digits (32 bits).

`Session Key Length`

Field 10, the length of the generated incoming PIN Encryption and MAC session keys.

Value	Session Key Length
S	Generate a 1key-3DES (single length) key
D	Generate a 2key-3DES (double length) key

Table 4-101 Command 346: PIN Translate - DUKPT to 3DES and Verify MAC

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	346
1	Header,$E_{MFK.E}$(Base Derivation Key),MAC	74	printable ASCII
2	Header,$E_{MFK.E}$(KPE_O),MAC	74	printable ASCII
3	$^{E}KPE_N$(PIN Block)	16	0 - 9, A - F
4	ANSI PAN Digits	12	0 - 9
5	Key Serial Number	10 - 20	0 - 9, A - F
6	MAC Data Length	1 - 4	0 - 9
7	MAC Data	2 - 4096	0 - 9, A - F
8	[MAC Type]	0, 1	empty, 1 - 7, V
9	MAC	8	0 - 9, A - F
10	Session Key Length	1	S or D

Responding Parameters

`446`

Field 0, the response identifier.

`[`$^{E}KPE_O$`(Outgoing PIN Block)]`

Field 1, the ANSI PIN block encrypted under the outgoing KPE. This field will contain a 16 hexadecimal character value only if the MAC verified and the PIN block passed the sanity check, otherwise it will be empty.

`[Sanity Check Indicator]`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

`MAC Verification Indicator`

Field 3, this field will contain:

Value	Description
Y	The MAC verified.
N	The MAC did not verify.

`[KPE`$_I$` Check Digits]`

Field 4, check digits of the generated incoming PIN encryption key. Check digits are the first six digits that result from encrypting zeros using the incoming PIN Encryption Key. This field is present only if the MAC verified.

`KMAC Check Digits`

Field 5, check digits of the generated KMAC session key. Check digits are the first six digits that result from encrypting zeros using the generated KMAC session key.

Table 4-102 Response 446: PIN Translate - DUKPT to 3DES and Verify MAC

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	446
1	[KPE$_O$(Outgoing PIN Block)]	0, 16	0 - 9, A - F
2	[Sanity Check Indicator]	0, 1	Y, N, L
3	MAC Verification Indicator	1	Y, N
4	[KPE$_I$ Check Digits]	0, 6	0 - 9, A - F
5	KMAC Check Digits	6	0 - 9, A - F

Usage Notes

- Generate the AKB for the base derivation key and the outgoing PIN encryption key

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text Base Derivation Key: 0123456789ABCDEF FEDCBA9876543210
  The Base Derivation Key in AKB format:
  1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,342946FE884AA8B2

- Clear-text Outgoing PIN Encryption Key:
  FEDCBA9876543210 0123456789ABCDEF
  The Outgoing PIN Encryption Key in AKB format:
  1PUNE000,8B672E58F4435F901BCC617C95C16388F06F9853B09A2301,DAD1F04BAC6D34BD

- Clear-text ANSI PIN block: 041274EDCBA9876F. The PIN is 1270.
  The encrypted incoming PIN block: 7A21BD10F36DC41D

- Twelve rightmost Primary Account Number digits: 041234567890

- Key serial number: 9876543210E00012

- The MAC data: 0123456789ABCEF

- The MAC Type: ISO 9797-1 Algorithm 1 - 3DES CBC

- The MAC to be verified: 6FCEDEBD

- The DUKPT session key length: 2key-3DES (double-length)

The command looks like this:

```
<346#1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,34
2946FE884AA8B2#1PUNE000,8B672E58F4435F901BCC617C95C16388F06F9853B
09A2301,DAD1F04BAC6D34BD#7A21BD10F36DC41D#041234567890#9876543210
E00012#16#0123456789ABCDEF#1#6FCEDEBD#D#>
```

The Network Security Processor returns the following response:

```
<446#7820FE6CFD54CE3A#Y#Y#A02300#B97051#>
```

PIN Translate - DUKPT to 3DES and Generate MAC (Command 347)

Command 347 translates a DUKPT encrypted ANSI PIN block from encryption under an incoming PIN encryption key (KPE_I) to encryption under an outgoing 3DES PIN encryption key (KPE_O), and also generates a MAC on a data field. The translated encrypted ANSI PIN block can also be included in the MAC calculation.

This command is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

Command

```
<347#Header,EMFK.E(Base Derivation Key),MAC#Key Serial Number#
Header,EMFK.E(KPEO),MAC#EKPEN(PIN Block)#ANSI PAN Digits#
Header,EMFK.E(KMAC),MAC#[MAC Type]#[Insertion Position]#Data Length#
MAC Data#Session Key Length#>
```

Response

```
<447#KPEO(Outgoing ANSI PIN Block)#[Sanity Check Indicator]#
[MAC]#[KPEI Check Digits]#KMAC Check Digits#>[CRLF]
```

Calling Parameters

`347`

Field 0, the command identifier.

`Header,EMFK.E(Base Derivation Key),MAC`

Field 1, the Base Derivation Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. If option 6C is enabled, the AKB can contain a 1key-3DES (single-length) key. The following headers are supported: 1dDNE000 and 1dDNN000.

`Key Serial Number`

Field 2, the 10 to 20 digit Key Serial Number (KSN) from the PIN entry device. This field contains a 10 to 20 hexadecimal digit value.

`Header,EMFK.E(KPEO),MAC`

Field 3, the outgoing PIN Encryption Key (KPE_O) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. If the Session Key Length is "D", the AKB must contain a 2key-3DES (double-length) key. If the Session Key Length is "S" and option 6C is enabled, the AKB can contain either a 1key-3DES (single-length) or 2key-3DES (double-length) key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, 1PUEN000, 1PDNE000, 1PDNN000, 1PDEE000, and 1PDEN000.

E_{KPE_N}(PIN Block)

Field 4, the incoming ANSI PIN Block encrypted under the DUKPT PIN Encryption Key. This field must contain 16 hexadecimal characters.

ANSI PAN Digits

Field 5, the 12 digits used to format the incoming encrypted ANSI PIN block.

Header,$E_{MFK.E}$(KMAC),MAC

Field 6, the Message Authentication Key (KMAC) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. If option 6C is enabled, the AKB can contain a 1key-3DES (single-length) key. The following headers are supported: 1MDNE000, 1MDNN000, 1MDGE000, and 1MDGN000.

[MAC Type]

Field 7, the type of MAC to be calculated. The possible values for this field are:

Value	**MAC Type**
Empty, or 1-6	ISO - 9797-1 Algorithm 1 - 1key or 2key-3DES Cipher block chaining
7	ISO 9797-1 Algorithm 3 - Only the last data block is processed using 3DES, all previous blocks are processed using single DES
V	Visa DUKPT (old style) as generated by command 5C

[Insertion Position]

Field 8, the number indicates where the translated encrypted ANSI PIN block is inserted into the data in this command for MAC generation. If you will not be including the translated encrypted ANSI PIN block in the MAC generation, set this field to 0. The number n means inserting the translated encrypted ANSI PIN block in between binary data position n-1 and n. This command only accepts unpacked ASCII hexadecimal data, but the same rule applies, two unpacked ASCII hexadecimal characters are considered one binary data. This field must contain an even number, or the character ‘F’ which indicates the first location in the data, or ‘L’ which indicates the last location in the data. The encrypted PIN block will be converted to unpacked ASCII hexadecimal characters before being including in the MAC data at the position indicated.

Example: Assume the data to be MACed is 0123456789ABCDEF and the outgoing encrypted ANSI PIN block is BA429B75C390ABE0 is to be inserted in position 8 of the data to be MACed, the resulting data to be MACed would be:

```
01234567BA429B75C390ABE089ABCDEF
```

MAC Data Length

Field 9, the number of bytes of data supplied in field 7. The minimum data length is 2, the maximum data length is 4096.

`MAC Data`

Field 10, the data in ASCII hexadecimal format that was used to generate the MAC. This field contains a 2 - 4096 hexadecimal character value. This field must contain an even number of hexadecimal characters.

`Session Key Length`

Field 11, the length of the generated incoming PIN Encryption.

Value	Session Key Length
S	Generate a 1key-3DES (single length) key
D	Generate a 2key-3DES (double length) key

If the Base Derivation Key, provided in field 1, is a 1key-3DES (single-length) key, this field must contain the letter S.

Table 4-103 Command 347: PIN Translate - DUKPT to 3DES and Generate MAC

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	347
1	Header,$E_{MFK.E}$(Base Derivation Key),MAC	74	printable ASCII
2	Key Serial Number	10 - 20	0 - 9, A - F
3	Header,$E_{MFK.E}$(KPE_O),MAC	74	printable ASCII
4	$^{E}KPE_N$(PIN Block)	16	0 - 9, A - F
5	ANSI PAN Digits	12	0 - 9
6	Header,$E_{MFK.E}$(KMAC),MAC	74	printable ASCII
7	[MAC Type]	0, 1	Empty, 1 - 7, V
8	[Insertion Position]	0 - 4	0 - 9, L, F
9	MAC Data Length	1 - 4	2 - 4096
10	MAC Data	2 - 4096	0 - 9, A - F
11	Session Key Length	1	S or D

Responding Parameters

`447`

Field 0, the response identifier.

`[E`$_{KPEo}$`(Outgoing PIN Block)]`

Field 1, the ANSI PIN block encrypted under the outgoing KPE. This field is present only if the PIN block passed the sanity check. If present, this field contains 16 hexadecimal characters.

`Sanity Check Indicator`

Field 2, the sanity check indicator. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. This field can contain one of the following values:

Value	Description
Y	PIN block passes the sanity check.
N	PIN failed the sanity test. Or the length of the PIN is out of range and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
L	The length of the PIN is out of range.

`[MAC]`

Field 3, the generated MAC. This field is present only if the PIN block passed the sanity check. If present, this field contains a 8 character (32 bits) hexadecimal value.

`KPE`$_I$` Check Digits`

Field 4, check digits of the generated incoming PIN encryption key. Check digits are the first six digits that result from encrypting zeros using the incoming PIN encryption key.

Table 4-104 Response 447: PIN Translate - DUKPT to 3DES and Generate MAC

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	447
1	[EKPE$_O$(Outgoing PIN Block)]	0, 16	0 - 9, A - F
2	Sanity Check Indicator	1	Y, N, L
3	[MAC]	0, 8	0 - 9, A - F
4	KPE$_I$ Check Digits	6	0 - 9, A - F

Usage Notes

- Generate the AKB for the base derivation key, message authentication key, and the outgoing PIN encryption key.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text Base Derivation Key: 0123456789ABCDEF FEDCBA9876543210
  The Base Derivation Key in AKB format:
  1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,342946FE884AA8B2
- Key serial number: 9876543210E00012
- Clear-text Outgoing PIN Encryption Key:
  FEDCBA9876543210 0123456789ABCDEF
  The Outgoing PIN Encryption Key in AKB format:
  1PUNE000,8B672E58F4435F901BCC617C95C16388F06F9853B09A2301,DAD1F04BAC6D34BD
- Clear-text ANSI PIN block: 041274EDCBA9876F. The PIN is 1270.
  The encrypted incoming PIN block: 7A21BD10F36DC41D
- Twelve rightmost Primary Account Number digits: 041234567890
- Clear-text KMAC: 0123456789ABCDEF FEDCBA9876543210
  The KMAC in AKB format:
  1MDGN000,33D236480E9998CEF8F22479E2FA6B545CD3AD9E90E55538,4BAF8087398163BC
- The MAC Type: ISO 9797-1 Algorithm 1 - 3DES CBC
- The outgoing encrypted ANSI PIN block is not used in the MAC generation process
- The MAC data: 0123456789ABCEF
- The DUKPT session key length: 2key-3DES (double-length)

The command looks like this:

```
<347#1dDNN000,4E752FC5FB33D8C8EA88FD6CE4AFABA06CF66C526FC4C976,29
F42CB5FB67327B#9876543210E00012#1PUNE000,8B672E58F4435F901BCC617C
95C16388F06F9853B09A2301,DAD1F04BAC6D34BD#7A21BD10F36DC41D#041234
567890#1MDGN000,33D236480E9998CEF8F22479E2FA6B545CD3AD9E90E55538,
4BAF8087398163BC##0#16#0123456789ABCDEF#D#>
```

The Network Security Processor returns the following response:

```
<447#7820FE6CFD54CE3A#Y#1A4D672D#A02300#>
```

Include the outgoing encrypted ANSI PIN block in the MAC calculation

- The MAC Type: ISO 9797-1 Algorithm 1 - 3DES CBC
- The outgoing encrypted ANSI PIN block is inserted in the MAC data at position 8
- The MAC Data: 0123456789ABCEF
- The DUKPT session key length: 2key-3DES (double-length)

The command looks like this:

```
<347#1dDNN000,4E752FC5FB33D8C8EA88FD6CE4AFABA06CF66C526FC4C976,29
F42CB5FB67327B#9876543210E00012#1PUNE000,8B672E58F4435F901BCC617C
95C16388F06F9853B09A2301,DAD1F04BAC6D34BD#7A21BD10F36DC41D#041234
567890#1MDGN000,33D236480E9998CEF8F22479E2FA6B545CD3AD9E90E55538,
4BAF8087398163BC##8#16#0123456789ABCDEF#D#>
```

The Network Security Processor returns the following response:

```
<447#7820FE6CFD54CE3A#Y#7DE72EEF#A02300#>
```

Generate ePIN Offset (Command 37B)

Command 37B is used to generate the ePIN Offset. This command uses a form of the IBM 3624 algorithm to generate the ePIN Offset, which is contained within the ePIN Object.

This command is not enabled in the Network Security Processor's default security policy. To enable this command, you must purchase it in the form of a command 105. To use this command, you must add it to the Network Security Processor's security policy.

Command

```
<37B#Offset Format#Header,EMFK.E(KPV),MAC#Header,EMFK.E(KOP),MAC#
ePIN#PAN#>
```

Response

```
<47B#ePIN Object#KPV Check Digits#KOP Check Digits#>[CRLF]
```

Calling Parameters

`37B`

Field 0, the command identifier.

`Offset Format`

Field 1, the offset format must be 2. This field contains 1 byte, the decimal value 2.

`Header,EMFK.E(KPV),MAC`

Field 2, the PIN Verification Key encrypted under the MFK. This key is used to generate the ePIN offset. This field must contain a 74 byte value. This key must be either a 2key-
or
3key-3DES key. The following headers are supported: 1V3NE000, 1V3NN000, 1V3GE000, and 1V3GN000.

`Header,EMFK.E(KOP),MAC`

Field 3, the Object PIN Key encrypted under the MFK. This key is used to decrypt the ePIN object. This field must contain a 74 byte value. This key must be either a 2key- or 3key-3DES key. The following headers are supported: 1PUNE000, 1PUNN000, 1PUEE000, and 1PUEN000.

`ePIN`

Field 4, the ePIN. This field must contain a 16 byte hexadecimal value.

`PAN`

Field 5, the Primary Account Number. This field must contain a 16 byte hexadecimal value.

Table 4-105 Command 37B: Generate ePIN Offset

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	37B
1	Offset Format	1	2
2	Header,$E_{MFK.E}$(KPV),MAC	74	printable ASCII
3	Header,$E_{MFK.E}$(KOP),MAC	74	printable ASCII
4	ePIN	16	0 - 9, A - F
5	PAN	16	0 - 9, A - F

Responding Parameters

`47B`

Field 0, the response identifier.

`ePIN Object`

Field 1, the ePIN Object. This field will contain a 32 byte hexadecimal value.

`KPV Check Digits`

Field 2, check digits; the first four digits that result from encrypting zeros using the PIN Verification Key. If option 88 is enabled, this field will contain six check digits.

`KOP Check Digits`

Field 3, check digits; the first four digits that result from encrypting zeros using the Object PIN Key. If option 88 is enabled, this field will contain six check digits.

Table 4-106 Response 47B: Generate ePIN Offset

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	47B
1	ePIN Object	32	0 - 9, A - F
2	KPV Check Digits	4 or 6	0 - 9, A - F
3	KOP Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- Generate the PIN Verification Key and PIN Object Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Generate an ePIN offset

- Clear-text PIN Verification Key: 4321432143214321 1234123412341234, check digits 2ABA. The PIN Verification Key in AKB format: 1V3NE000,DF7DC3D37F898C6E1916FC4AEEE7849E8300F2638934F1F4,7EFCD1B1C6D893DC

- Clear-text Object PIN Key: 5678567856785678 8765876587658765, check digits 686F. The Object PIN Key in AKB format: 1PUNE000,8586FE8ECDA3D4EE40FD86769D5B6EF3DE0847DA7A40455B,90DAE3B79A2E73F6

- ePIN: 314A41434B2A2A2A

- PAN: ABCD123456789012

The command looks like this:

```
<37B#2#1V3NE000,DF7DC3D37F898C6E1916FC4AEEE7849E8300F2638934F1F4,7E
FCD1B1C6D893DC#1PUNE000,8586FE8ECDA3D4EE40FD86769D5B6EF3DE0847DA7A4
0455B,90DAE3B79A2E73F6#314A41434B2A2A2A#ABCD123456789012#>
```

The Network Security Processor returns the following response:

```
<47B#27BDDE807F87DDD4589226D1F475CD0E#2ABA#686F#>
```

5

Processing Transaction Data

The Network Security Processor uses the Data Encryption Algorithm (DEA) as defined in the Data Encryption Standard (DES). See Federal information Processing Standard 46-3 for information on DES. See American National Standard Institute X9.52 for information on Triple DES.

Processing transaction data using Triple DES, involves three basic steps: encrypting, decrypting, and authenticating. This section explains data encryption and decryption. Authenticating Transaction Data explains data authentication.

To skip this introduction, go to Table 5-1 on page 5-4 for a list of commands.

Data Processing Tasks

Processing transaction data typically involves the following tasks:

- Establishing a Data Encryption/Decryption Key.
- Deciding which part of each message will be encrypted – the entire message or selected portions of it.
- Encrypting the data for network transmission.
- Transmitting the data.
- Decrypting it at the remote node.

Establishing a common Data Encryption/Decryption Key is discussed in Key Management. Deciding which portions of the message to encrypt and transmitting the data are site-specific tasks that are not covered in this manual.

Encrypting and Decrypting Data

Encryption is the process of using a Data Encryption Key to scramble data so that it cannot be read by someone who does not know the key. Encryption provides privacy.

Supported Encryption/Decryption Methods

Data can be encrypted or decrypted using a variety of schemes. The Network Security Processor supports the following methods, see Federal Information Processing Standard 81 Modes of DES for more information.

- Cipher block chaining (CBC)
- Cipher feedback, eight bits (CFB-8)
- Cipher feedback, 64 bits (CFB-64)
- Output feedback, 64 bits (OFB-64)
- Electronic Code Book (ECB) (CBC mode can be used indirectly to support ECB).

For the data encryption modes that Atalla supports, encryption can be expressed as the following function.

```
Encrypted data = ƒ(data, IV)
```

In other words, encryption is a function of data and an Initialization Vector. The next few paragraphs discuss Initialization Vectors.

Using Initialization Vectors

An Initialization Vector is a value that is exclusive or'd with the first block of data to be encrypted. This ensures that every clear-text string of data – including identical strings – is encrypted differently.

The following examples illustrate two ways that the device can be used to encrypt the data string and the role that Initialization Vectors play in each case.

Encryption All at Once

The first way the string can be encrypted is all at once. Thus, the starting string is "This is an idea." and the starting Initialization Vector is X. (X is a randomly generated number.

ƒ(This is an idea., E_{IV} = X)

The result is an encrypted string and the ending Initialization Vector, Y. The ending Initialization Vector is a value that depends on the starting Initialization Vector, the Data Encryption Key and the data.

(abcdefghijklmnop, E_{IV} = Y)

This method of encrypting data – all at once, with a starting and ending Initialization Vector – is sufficient whenever you are working with messages that contain fewer than 4096 bytes of data.

Encryption in Batches

The second way the string can be encrypted is in batches. In this example, the starting string is, “This is” and the starting Initialization Vector is X. X is a randomly generated number.

```
ƒ(This is , E_IV = X)
```

The result is an encrypted string and the Initialization Vector, Z. The value of this Initialization Vector depends on the starting Initialization Vector, the Data Encryption Key and the data.

```
(abcdefgh, E_IV = Z)
```

To encrypt the rest of the string, supply the remainder of the data and Z, the Initialization Vector obtained when you encrypted the first part of the string.

```
ƒ(an idea., E_IV = Z)
```

The result is an encrypted string and the ending Initialization Vector, Y. (Again, the value of this Initialization Vector depends on both the Data Encryption Key and the data.

```
(ijklmnop, E_IV = Y)
```

This method of encrypting data – in batches – must be used whenever you are working with messages that contain more than 4096 bytes of data. The Initialization Vector Z is called the **continuing Initialization Vector**. Z is the ending Initialization Vector for the first batch of data and the starting Initialization Vector for the next batch of data. Notice that both methods of encrypting data – all at once and in batches – have the same ending Initialization Vector, Y, because the same data was used. Y is dependent on both the key and the data.

note When encrypting data in batches, the length of the data encrypted in each batch – except for the last batch – must be a multiple of eight. The length of the last batch of data encrypted is not restricted to a multiple of 8.

Data Processing Commands

The rest of this section contains the command and response syntax for the Network Security Processor data processing commands.

Quick Reference

Table 5-1 on page 5-4 identifies each command by number, name, and purpose. While the table organizes the data processing commands by category, the commands themselves are presented in numerical order.

Table 5-1 Data Processing Commands

Command	Name	Purpose
Data encrypting and decrypting commands		
55	Data Encrypt, Decrypt, or Translate Link I to Link J	Encrypts clear data using one or two keys, decrypts single- or double-encrypted data, and translates single- or double-encrypted data. This command supports only the ECB mode of DES.
97	Encrypt/Decrypt Data	Encrypts clear data or decrypts ciphered data.
138	RSA Data Decrypt	Decrypts data using an RSA private key.
388	3DES DUKPT Encrypt/Decrypt Data	Encrypts clear data or decrypts ciphered data using a 3DES DUKPT data key.
390	AES Encrypt/Decrypt Data	Encrypts clear data or decrypts ciphered data using a the AES algorithm.
Initialization vector commands		
94	Generate Initialization Vector	Generates an Initialization Vector.
95	Reformat Initialization Vector	Reformats an Initialization Vector for communicating on SNA networks.
96	Verify Initialization Vector	Verifies the format and contents of Initialization Vectors transmitted and received on an SNA network.
Other commands		
93	Generate Random Number	Generates a random number.

Encrypt, Decrypt Or Translate Data (Command 55)

Command 55 encrypts clear data using one or two 3DES operations, 3DES decrypts single- and double-encrypted data, and 3DES translates single- or double-encrypted data. The 3DES ECB mode will be used for all operations.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must enable it in the Network Security Processor's security policy.

Command

<55#[Header,$E_{MFK.E}$(KD$_{I1}$),MAC]#[Header,$E_{MFK.E}$(KD$_{I2}$),MAC]#
[Header,$E_{MFK.E}$(KD$_{O1}$),MAC]#[Header,$E_{MFK.E}$(KD$_{O2}$),MAC]#Reserved#
Data#Reserved#Reserved#Reserved#Reserved#Reserved#>

Response

<65#Reserved#Data#[KC$_{I1}$ Check Digits]#[KC$_{I2}$ Check Digits]#
[KC$_{O1}$ Check Digits]#[KC$_{O2}$ Check Digits]#>[CRLF]

Calling Parameters

`55`

Field 0, the command identifier.

[Header,EMFK.E(KD$_{I1}$]

Field 1, the first incoming data-encryption key encrypted under the MFK. This key is used in the inner or first layer of encryption. If the input data is clear text, this field is empty. Otherwise, This field contains a 74 byte value, or a volatile table location. The following headers are supported in this command: 1DDNE000, 1DDNN000, 1DDDE000, and 1DDDN000.

[Header,$E_{MFK.E}$(KD$_{I2}$),MAC]

Field 2, the second incoming data-encryption key encrypted under the MFK. This key is used in the outer or second layer of encryption. If the input data is single encrypted, this field is empty. Otherwise, This field contains a 74 byte value, or a volatile table location. The following headers are supported in this command: 1DDNE000, 1DDNN000, 1DDDE000, and 1DDDN000.

[Header,$E_{MFK.E}$(KD$_{O1}$),MAC]

Field 3, the first outgoing data-encryption key encrypted under the MFK. This key is used in the inner or first layer of encryption. If the output data is clear text, this field is empty. Otherwise, This field contains a 74 byte value, or a volatile table location. The following headers are supported in this command: 1DDNE000, 1DDNN000, 1DDEE000, and 1DDEN000.

`[Header,`$E_{MFK.E}$`(`KD_{O2}`),MAC]`

Field 4, the second outgoing data-encryption key encrypted under the MFK. This key is used in the outer or second layer of encryption. If the output data is single encrypted, this field is empty. Otherwise, This field contains a 74 byte value, or a volatile table location. The following headers are supported in this command: 1DDNE000, 1DDNN000, 1DDEE000, and 1DDEN000.

`Reserved`

Field 5, reserved for future use. Its value must be 00.

`Data`

Field 6, input data. This field contains a multiple of 16 hexadecimal characters (represented as ASCII characters in the command). The number of 16 character blocks is *n*, where *n* is 1 to 10.

`Reserved`

Field 7, Its value must be 1111.

`Reserved`

Fields eight to 11, reserved for future use. All four fields must be empty.

Table 5-2 Command 55: Encrypt, Decrypt or Translate Data

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	2	55
1	Header,$E_{MFK.E}$(KD_{I1}*	0, 74	printable ASCII
2	Header,$E_{MFK.E}$(KD_{I2}),MAC*	0, 74	printable ASCII
3	Header,$E_{MFK.E}$(KD_{O1}),MAC*	0, 74	printable ASCII
4	Header,$E_{MFK.E}$(KD_{O2}),MAC*	0, 74	printable ASCII
5	Reserved	2	00
6	Data**	16*n*	0 - 9, A - F
7	Reserved	4	1111
8	Reserved	0	
9	Reserved	0	
10	Reserved	0	
11	Reserved	0	

* Can be a volatile table location.

** This field is a multiple of 16 hexadecimal characters (represented as ASCII characters in the command). The number of 16-character blocks is *n*, where *n* is from 1 to 10.

Responding Parameters

`65`

Field 0, the response identifier.

`Reserved`

Field 1, reserved for future use. This field will contain 00.

`Data`

Field 2, the output data. This field contains a multiple of 16 hexadecimal characters (represented as ASCII characters in the command). The number of 16 character blocks is *n*, where *n* is 1 to 10.

[KC_{I1} Check Digits]

Field 3, the first incoming Data Encryption Key's check digits; the first four digits that result from encrypting zeros using the first incoming Data Encryption Key. If option 88 is enabled, this field will contain six check digits. If command Field 1 is empty, this field is also empty.

[KC_{I2} Check Digits]

Field 4, the second incoming Data Encryption Key's check digits; the first four digits that result from encrypting zeros using the second incoming Data Encryption Key. If option 88 is enabled, this field will contain six check digits. If command Field 2 is empty, this field is also empty.

[KC_{O1} Check Digits]

Field 5, the first outgoing Data Encryption Key's check digits; the first four digits that result from encrypting zeros using the first outgoing Data Encryption Key. If option 88 is enabled, this field will contain six check digits. If command Field 3 is empty, this field is also empty.

[KC_{O2} Check Digits]

Field 6, the second outgoing Data Encryption Key's check digits; the first four digits that result from encrypting zeros using the second outgoing Data Encryption Key. If option 88 is enabled, this field will contain six check digits.

Table 5-3 Response 65: Encrypt, Decrypt or Translate Data

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	65
1	Reserved	2	00
2	Data*	16*n*	0 - 9, A - F
3	KC_{I1} Check Digits	0, 4 or 6	0 - 9, A - F
4	KC_{I2} Check Digits	0, 4 or 6	0 - 9, A - F
5	KC_{O1} Check Digits	0, 4 or 6	0 - 9, A - F
6	KC_{O2} Check Digits	0, 4 or 6	0 - 9, A - F

* This field is a multiple of 16 hexadecimal characters (represented as ASCII characters in the command). The number of 16 character blocks is *n*, where *n* is from 1 to 10.

Usage Notes

- Before using this command, generate the incoming and outgoing communications keys.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate data using two keys

- Clear-text first incoming Data Encryption Key: 2222 2222 2222 2222, check digits = 0096

- The first incoming Data Encryption Key in AKB format: 1DDNE000,B6790C480725B127035165B51C9D43163EAC111AE8E47607,0BFB32AF7CFBC885

- Clear-text second incoming Data Encryption Key: 3333 3333 3333 3333 check digits = ADC6. The second incoming Data Encryption Key in AKB format: 1DDNE000,09FFCD303017DA193AE10630F7A97FB04EAC17E4CA01A1FC,966113F44A28E527

- Clear-text first outgoing Data Encryption Key: 4444 4444 4444 4444, check digits = E2F2. The first outgoing Data Encryption Key in AKB format: 1DDNE000,22D9412C95CE2A5813CE91B1403C32DB89F0D4784632DE02,CBBCF7EB76B31054

- Clear-text second outgoing Data Encryption Key: 5555 5555 5555 5555. check digits = 0EE1. The second outgoing Data Encryption Key in AKB format: 1DDNE000,9F94FD3C9FC5E3B0087C46D1B1C64E7014A8B8BD6F5A0B27,3A7132E3DF808B47

- Data: 1234 5678 9012 3456

The command looks like this:

```
<55#1DDNE000,B6790C480725B127035165B51C9D43163EAC111AE8E47607,0BFB3
2AF7CFBC885#1DDNE000,09FFCD303017DA193AE10630F7A97FB04EAC17E4CA01A1
FC,966113F44A28E527#1DDNE000,22D9412C95CE2A5813CE91B1403C32DB89F0D4
784632DE02,CBBCF7EB76B31054#1DDNE000,9F94FD3C9FC5E3B0087C46D1B1C64E
7014A8B8BD6F5A0B27,3A7132E3DF808B47#00#1234567890123456#1111#####>
```

The Network Security Processor returns the following response:

```
<65#00#89F8D54F1DA00CB6#0096#ADC6#E2F2#0CD7#>
```

Generate Random Number (Command 93)

Command 93 generates a random hexadecimal or decimal number. This command is enabled in the Network Security Processor's default security policy.

Command

```
<93#[Format#Length#]>
```

Response

```
<A3#Random Number#>[CRLF]
```

Calling Parameters

note Fields one and two are optional, however field 2 cannot exist without field 1. If both fields are omitted, the command generates a 16 byte hexadecimal value.

`93`

Field 0, the command identifier.

`[Format#]`

Field 1, the random number's format. This field contains one byte; H for hexadecimal, or D for decimal.

`[Length#]`

Field 2, the number of digits in the random number. This field contains a 1 to 3 byte decimal value in the range of 4 - 128.

Table 5-4 Command 93: Generate Random Number

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	93
1	Format*	1	H, D
2	Length*	1 - 3	4 - 128

* Optional field

Responding Parameters

`A3`

Field 0, the response identifier.

`Random Number`

Field 1, the random number, a decimal or hexadecimal value.

Table 5-5 Response A3: Generate Random Number

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	A3
1	Random number	4 - 128	0 - 9, A - F

Usage Notes

- Randomly generated hexadecimal values are typically used as Initialization Vectors.
- Randomly generated decimal numbers are typically used as challenge numbers.

Examples

This command generates a random value your results will be different.

Generate a Random Number without Specifying Format or Length

The command looks like this:

```
<93#>
```

The Network Security Processor returns the following response:

```
<A3#A23D79FEDB1329AB#>
```

Generate a Four Digit Random Hexadecimal Number

The command looks like this:

```
<93#H#4#>
```

The Network Security Processor returns the following response:

```
<A3#1A7B#>
```

Generate a Six Digit Random Decimal Number

The command looks like this:

```
<93#D#6#>
```

The Network Security Processor returns the following response:

```
<A3#327179#>
```

Generate Initialization Vector (Command 94)

Command 94 generates an Initialization Vector.

This command is not enabled in the Network Security Processor’s default security policy. To use this command, you must enable it in the Network Security Processor’s security policy.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

Command

<94#Header,$E_{MFK.E}$(KD),MAC#>

Response

<A4#E_{KD}(IV)#Header,$E_{MFK.E}$(IV),MAC#>[CRLF]

Calling Parameters

`94`

Field 0, the command identifier.

Header,$E_{MFK.E}$(KD),MAC

Field 1, the Data Encryption Key (KD) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported in this command: 1DDNE000, 1DDNN000, 1DDEE000, and 1DDEN000.

Table 5-6 Command 94: Generate Initialization Vector

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	94
1	Header,$E_{MFK.E}$(KD),MAC*	74	printable ASCII

* Can be a volatile table location.

Responding Parameters

`A4`

Field 0, the response identifier.

E_{KD}(IV)

Field 1, the generated Initialization Vector encrypted using the Data Encryption Key. This field contains a 16 byte hexadecimal value.

```
Header,E_MFK.E(IV),MAC
```

Field 2, the generated Initialization Vector encrypted under the MFK. This field contains a 74 byte value.

Table 5-7 Response A4: Generate Initialization Vector

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	A4
1	E_{KD}(IV)	16	0 - 9, A - F
2	Header,$E_{MFK.E}$(IV),MAC	74	printable ASCII

Usage Notes

- This command can be used to generate the Initialization Vector for the following encryption schemes: Cipher block chaining (CBC), cipher feedback – eight bits (CFB-8), cipher feedback – 64 bits (CFB-64), and output feedback – 64 bits (OFB-64).
- Before using this command, generate the Data Encryption Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

This command generates a random value your results will be different.

Generate an Initialization Vector

- Clear-text Data Encryption Key (KD): 2222 2222 2222 2222
  The Data Encryption Key (KD) in AKB format:
  1DDNE000,B6790C480725B127035165B51C9D43163EAC111AE8E47607,0BFB32AF7C FBC885

The command looks like this:

```
<94#1DDNE000,B6790C480725B127035165B51C9D43163EAC111AE8E47607,0BFB3
2AF7CFBC885#>
```

The Network Security Processor returns the following response:

```
<A4#1EF048BB2BCB992E#1IDNE000,FC478C5312B8C751AA0B8DBBA329020C180D5
A645478E335,F05323AD7D529DA7#>
```

Reformat Initialization Vector (Command 95)

Command 95 reformats an Initialization Vector for communicating on SNA networks.

This command is not enabled in the Network Security Processor’s default security policy. To use this command, you must enable it in the Network Security Processor’s security policy.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

Command

<95#Header,$E_{MFK.E}$(KD),MAC#E_{KD}(IV)#>

Response

<A5#E_{KD}(Reformatted IV)#Header,$E_{MFK.E}$(IV),MAC#>[CRLF]

Calling Parameters

`95`

Field 0, the command identifier.

Header,$E_{MFK.E}$(KD),MAC

Field 1, the Data Encryption Key (KD) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported in this command: 1DDNE000, 1DDNN000, 1DDEE000, and 1DDEN000.

E_{KD}(IV)

Field 2, the Initialization Vector encrypted under the Data Encryption Key. This field contains a 16 byte hexadecimal value.

Table 5-8 Command 95: Reformat Initialization Vector

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	95
1	Header,$E_{MFK.E}$(KD),MAC*	74	printable ASCII
2	E_{KD}(IV)	16	0 - 9, A - F

* Can be a volatile table location.

Responding Parameters

`A5`

Field 0, the response identifier.

`E`$_{KD}$`(Reformatted IV)`

Field 1, the reformatted Initialization Vector encrypted under the Data Encryption Key. This value is distributed on SNA networks and returned to the originating node for verification. The reformatted Initialization Vector is formed by taking the complement of the original Initialization Vector's first four bytes, appending to this new value the original Initialization Vector's remaining bytes. This field contains a 16 byte hexadecimal value.

`Header,E`$_{MFK.E}$`(IV),MAC`

Field 2, the original Initialization Vector encrypted under the MFK. This field contains a 74 byte value.

Table 5-9 Response A5: Reformat Initialization Vector

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	A5
1	E$_{KD}$(Reformatted IV)	16	0 - 9, A - F
2	Header,E$_{MFK.E}$(IV),MAC	74	printable ASCII

Usage Notes

Perform the following tasks before using Command 95:

- Generate the Data Encryption Key.
- Generate the Initialization Vector.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Reformat an Initialization Vector

- Clear-text Data Encryption Key (KD): 2222 2222 2222 2222
  The Data Encryption Key (KD) in AKB format:
  1DDNE000,B6790C480725B127035165B51C9D43163EAC111AE8E47607,0BFB32AF7C FBC885

- Clear-text Initialization Vector: 1253 9610 3707 5240
  The Initialization Vector encrypted under the Data Encryption Key: 85F9 9910 AF42 D767

The command looks like this:

```
<95#1DDNE000,B6790C480725B127035165B51C9D43163EAC111AE8E47607,0BFB3
2AF7CFBC885#85F99910AF42D767#>
```

The Network Security Processor returns the following response:

```
<A5#B078D0BD55356757#1IDNE000,D9987364E9AE40C252588FDDACADA8B704C19
9BBAD910475,AA18D1DDD971AF02#>
```

Verify Initialization Vector (Command 96)

Command 96 verifies the format and contents of an Initialization Vector transmitted and received on an SNA network.

This command is not enabled in the Network Security Processor’s default security policy. To use this command, you must enable it in the Network Security Processor’s security policy.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

Command

```
<96#Header,EMFK.E(IV),MAC#Header,EMFK.E(KD),MAC#EKD(Reformatted IV)#>
```

Response

```
<A6#Verification Flag#>[CRLF]
```

Calling Parameters

`96`

Field 0, the command identifier.

`Header,EMFK.E(IV),MAC`

Field 1, the Initialization Vector encrypted under the MFK. This field contains a 74 byte value. The following headers are supported in this command: 1IDNE000 and 1IDNN000.

`Header,EMFK.E(KD),MAC`

Field 2, the Data Encryption Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported in this command: 1DDNE000, 1DDNN000, 1DDEE000, and 1DDEN000.

`EKD(Reformatted IV)`

Field 3, the reformatted Initialization Vector encrypted under the Data Encryption Key. This field contains a 16 byte hexadecimal value.

Table 5-10 Command 96: Verify Initialization Vector

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	96
1	Header,$E_{MFK.E}$(IV),MAC	74	printable ASCII
2	Header,$E_{MFK.E}$(KD),MAC*	74	printable ASCII
3	E_{KD}(Reformatted IV)	16	0 - 9, A - F

* Can be a volatile table location.

Responding Parameters

`A6`

Field 0, the response identifier.

`Verification Flag`

Field 1, the verification flag. This field returns Y if the Initialization Vectors are identical, otherwise, it returns N.

Table 5-11 Response A6: Verify Initialization Vector

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	A6
1	Verification flag	1	Y, N

Usage Notes

Perform the following tasks before using Command 96:

- Generate the Initialization Vector.
- Generate the Data Encryption Key.
- Generate the reformatted Initialization Vector.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify an Initialization Vector

- Clear-text Initialization Vector: 1253 9610 3707 5240
  The Initialization Vector in AKB format:
  1IDNE000,D9987364E9AE40C252588FDDACADA8B704C199BBAD910475,AA18D1D DD971AF02
- Clear-text Data Encryption Key (KD): 2222 2222 2222 2222
  The Data Encryption Key (KD) in AKB format:
  1DDNE000,B6790C480725B127035165B51C9D43163EAC111AE8E47607,0BFB32AF7C FBC885
- Clear-text reforrmatted Initialization Vector: EDAC 69EF 3707 5240
  The reformatted Initialization Vector encrypted under the Data Encryption Key: B078 D0BD 5535 6757

The command looks like this:

```
<96#1IDNE000,D9987364E9AE40C252588FDDACADA8B704C199BBAD910475,AA18D
1DDD971AF02#1DDNE000,B6790C480725B127035165B51C9D43163EAC111AE8E476
07,0BFB32AF7CFBC885#B078D0BD55356757#>
```

The Network Security Processor returns the following response:

```
<A6#Y#>
```

Encrypt/Decrypt Data (Command 97)

Command 97 encrypts clear data or decrypts encrypted data. Several DES methods are supported. Both binary and ASCII hexadecimal data types are supported.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

Command

<97#Operation#DES Method#Header,$E_{MFK.E}$(KD),MAC#
[Header,$E_{MFK.E}$(IV),MAC]#Data Type#Length#Data#[Reserved#]>

Response

<A7#Operation#DES Method#KD Check Digits#Header,$E_{MFK.E}$(IV),MAC#
Header,$E_{MFK.E}$(Ending IV),MAC#Data Type#Length#Data#>[CRLF]

Calling Parameters

`97`

Field 0, the command identifier.

`Operation`

Field 1, indicates the operation to be performed on the data. This field contains 1 byte, either E to indicate encryption, or D to indicate decryption.

`DES Method`

Field 2, the DES method of encryption or decryption to be used. The following table indicates the encryption and decryption methods that the Network Security Processor supports, and the value to enter in this field for each method.

Value	DES Method
1 or 6	3DES Cipher block chaining (CBC)
2	Cipher feedback – eight bits (CFB-8)
3	Cipher feedback – 64 bits (CFB-64)
4	Output feedback – 64 bits (OFB-64)

`Header,`$E_{MFK.E}$`(KD),MAC`

Field 3, the Data Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported in this command: 1DDNE000, 1DDNN000, 1DDEE000, 1DDEN000, 1DDDE000, and 1DDDN000.

`[Header,`$E_{MFK.E}$`(IV),MAC]`

Field 4, the Initialization Vector encrypted under the MFK.

If the operation is Encryption, this field can be:

- Empty, in which case a randomly generated Initialization Vector is used.
- The letter “D”, in which case a null IV (binary zeros) is used.
- A 74 byte AKB value of the Initialization Vector encrypted under the MFK.
- A 16 byte cleartext Initialization Vector (in version 1.50, and above).

If the operation is Decryption, this field can be:

- The letter “D”, in which case a null IV (binary zeros) is used.
- A 74 byte AKB value of the Initialization Vector encrypted under the MFK.
- A 16 byte cleartext Initialization Vector (in version 1.50, and above).

The following headers are supported in this command: 1IDNE000, and 1IDNN000.

`Data Type`

Field 5, the data type. The following table indicates the type of data that the Network Security Processor supports and the value to enter in this field for each type.

Type	Data Type
U	Unpacked ASCII hexadecimal
B	Binary

See Data formats for more information on data types and cautions on using Binary data.

`Length`

Field 6, the data’s length. This command will encrypt or decrypt up to 4096 bytes of data. For all methods of encryption except CFB-8, the data will be right-padded with zeros by the Network Security Processor, to achieve an 8 byte multiple. For all methods of decryption except CFB-8, the data length must be an 8 byte multiple.

Since CFB-8 operates on 8 bit values, the minimum binary data length is 1, and the minimum unpacked ASCII data length is 2.

This command will encrypt or decrypt up to 4096 bytes of data when using type 'B'. For type 'U', the command will process up to 4096 characters of data.

`Data`

Field 7, the input data, encrypted or in clear-text format. The length of this field must match what was specified in field 6.

`[Reserved#]`

Field 8, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

Table 5-12 Command 97: Encrypt/Decrypt Data

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	97
1	Operation	1	D, E
2	DES method	1	1 - 4, 6
3	Header,$E_{MFK.E}$(KD),MAC*	74	printable ASCII
4	Header,$E_{MFK.E}$(IV),MAC	0, 1, 16, 74	printable ASCII
5	Data type	1	U, B
6	Length	1 - 4	0 - 9
7	Data	1 - 4096	0 - 9, A - F, if unpacked ASCII
8	Reserved	0, 1, 2	0 - 31

* Can be a volatile table location.

Responding Parameters

`A7`

Field 0, the response identifier.

`Operation`

Field 1, the operation performed on the data: Encryption (E) or Decryption (D). This field will contain the value specified in field 1 of the command.

`DES Method`

Field 2, the DES method of encryption or decryption used. This field will contain the value specified in field 2 of the command.

`KD Check Digits`

Field 3, check digits; the first four digits that result from encrypting zeros using the Data Key. If option 88 is enabled, this field will contain six check digits.

`Header,`$E_{MFK.E}$`(IV),MAC`

Field 4, the Initialization Vector encrypted under the MFK. The Initialization Vector is the one specified in the command or the one generated by the Network Security Processor, if an Initialization Vector was not specified in the command. This field contains a 74 byte value. The letter D will be returned in this field if a D was supplied in field 4 of the command.

`Header,`$E_{MFK.E}$`(Ending IV),MAC`

Field 5, the ending Initialization Vector under the MFK. This ending Initialization Vector must be used as the starting Initialization Vector for the next block of data if the amount of data to be encrypted or decrypted will not fit in one command. This field contains a 74 byte value.

`Data Type`

Field 6, the type of data returned in Field 8: unpacked ASCII hexadecimal or binary. This field will contain the value specified in field 5 of the command.

`Length`

Field 7, the length of the returned data. This field contains a 1 to 4 byte decimal value.

If you performed an encryption in this command, the length returned here may be longer than the clear-text data length. When encrypting data, except when in CFB-8 mode, the Network Security Processor pads the input with binary zeros to achieve an 8 byte multiple.

`Data`

Field 8, the encrypted or decrypted data. This field can be from 1 to 4096 bytes long.

Table 5-13 Response A7: Encrypt/Decrypt Data

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	A7
1	Operation	1	D, E
2	DES method	1	1 - 4, 6
3	KD Check Digits	4 or 6	0 - 9, A - F
4	Header,$E_{MFK.E}$(IV),MAC	74	printable ASCII

Table 5-13 Response A7: Encrypt/Decrypt Datasn(continued)

Field	Contents	Length (bytes)	Legal Characters
5	Header,$E_{MFK.E}$(Ending IV),MAC	74	printable ASCII
6	Data type	1	U, B
7	Length	1 - 4	0 - 9
8	Data	8 - 4096	0 - 9, A - F if unpacked ASCII

Usage Notes

- If you are encrypting or decrypting large amounts of data, you should specify the ending Initialization Vector, returned in the response for the first block of data, to be the starting Initialization Vector in the encryption or decryption for the next block of data. Be sure to specify the same key and DES method for each data block.
- Generate the Data Key.
- Generate the Initialization Vector.
- This command can contain a binary data field. The Network Security Processor's and the host's serial interface must both be configured to 8 data bits, 1 stop bit, and no parity to correctly process binary data.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Encrypt Data

Message to be encrypted is 31 characters: "Sell stock when price is > $50."

- Encryption method: 3DES Cipher block chaining (6)
- Clear-text Data Key: 0123 4567 89AB CDEF FEDC BA98 7654 3210
  The Data Key in AKB format:
  1DDNE000,4509C24B1C13C379F1EC519B255CD9B96F777E7BBDCC5D47,8998D7F46 4EED3A2
- Initialization Vector: nulls (16 binary zeros)
- Data type: Unpacked ASCII hexadecimal (U)

- Data length: 62
- The data in ASCII hexadecimal format: 53656C6C2073746F636B207768656E20707269636520697320 3E202435302E

The command looks like this:

```
<97#E#6#1DDNE000,4509C24B1C13C379F1EC519B255CD9B96F777E7BBDCC5D47,8
998D7F464EED3A2#D#U#62#53656C6C2073746F636B207768656E20707269636520
6973203E202435302E#>
```

The Network Security Processor returns the following response:

```
<A7#E#6#08D7#D#1IDNE000,94EE92573FCF2628B77020FE3FEAF75712F7C02BB60
AD550,6AC939A8CD533D8D#U#64#B7924DAE35158F21FAAFFF5038557958DD01321
313398AE04A96FC47B9F52609#>
```

Decrypt Data

- Decryption method: Cipher block chaining (6)
- Clear-text Data Key: 0123 4567 89AB CDEF FEDC BA98 7654 3210
  The Data Key in AKB format:
  1DDNE000,4509C24B1C13C379F1EC519B255CD9B96F777E7BBDCC5D47,8998D7F46 4EED3A2
- Initialization Vector: nulls (16 binary zeros)
- Data type: Unpacked ASCII hexadecimal (U)
- Data length: 64
- The data to be decrypted: B7924DAE35158F21FAAFFF5038557958DD01321313398AE04A96FC47B9F52609

The command looks like this:

```
<97#D#6#1DDNE000,4509C24B1C13C379F1EC519B255CD9B96F777E7BBDCC5D47,8
998D7F464EED3A2#D#U#64#B7924DAE35158F21FAAFFF5038557958DD0132131339
8AE04A96FC47B9F52609#>
```

The Network Security Processor returns the following response:

```
<A7#D#6#08D7#D#1IDNE000,94EE92573FCF2628B77020FE3FEAF75712F7C02BB60
AD550,6AC939A8CD533D8D#U#64#53656C6C2073746F636B207768656E207072696
365206973203E202435302E00#>
```

Encrypt Multiple Blocks of Data

This example will encrypt a total of 31 bytes of data using two data encrypt commands. The first command will encrypt 16 bytes, the second command will encrypt 15 bytes. The ending IV returned in the response from the first command will be used as the IV in the second command.

Message to be encrypted is 31 characters: “Sell stock when price is > $50.”

- Encryption method: Cipher block chaining (6)
- Clear-text Data Key: 0123 4567 89AB CDEF FEDC BA98 7654 3210
  The Data Key in AKB format:
  1DDNE000,4509C24B1C13C379F1EC519B255CD9B96F777E7BBDCC5D47,8998D7F46
  4EED3A2
- Initialization Vector: nulls (16 binary zeros)
- Data type: Unpacked ASCII hexadecimal (U)
- Data length for the first command : 32
- Data for the first command: 53656C6C2073746F636B207768656E20

The first command looks like this.

```
<97#E#6#1DDNE000,4509C24B1C13C379F1EC519B255CD9B96F777E7BBDCC5D47,8
998D7F464EED3A2#D#U#32#53656C6C2073746F636B207768656E20#>
```

The Network Security Processor returns the following response:

```
<A7#E#6#08D7#D#1IDNE000,6903E0D609CACEFE109A8EB7FB8E615588F046AAFA4
E53E7,6A8ABB7415E213E7#U#32#B7924DAE35158F21FAAFFF5038557958#>
```

- Data length for the second command : 30
- Data for the second command : 7072696365206973203E202435302E
- Ending IV in AKB format from the first command:
  1IDNE000,6903E0D609CACEFE109A8EB7FB8E615588F046AAFA4E53E7,6A8ABB7415E
  213E7

The second command looks like this.

```
<97#E#6#1DDNE000,4509C24B1C13C379F1EC519B255CD9B96F777E7BBDCC5D47,8
998D7F464EED3A2#1IDNE000,6903E0D609CACEFE109A8EB7FB8E615588F046AAFA
4E53E7,6A8ABB7415E213E7#U#30#7072696365206973203E202435302E#>
```

The Network Security Processor returns the following response:

```
<A7#E#6#08D7#1IDNE000,6903E0D609CACEFE109A8EB7FB8E615588F046AAFA4E53E
7,6A8ABB7415E213E7#1IDNE000,94EE92573FCF2628B77020FE3FEAF75712F7C02BB
60AD550,6AC939A8CD533D8D#U#32#DD01321313398AE04A96FC47B9F52609#>
```

RSA Data Decrypt (Command 138)

Command 138 uses the RSA private key to decrypt data. The encrypted data must be provided as ASCII hexadecimal characters.

This command is a premium value command. It is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<138#Padding Scheme#Encrypted Data#Key Slot or AKBPRV#>
```

Response

```
<238#Decrypted Data#[Check Digits]#>[CRLF]
```

Calling Parameters

`138`

Field 0, the command identifier.

`Padding Scheme`

Field 1, this field must contain the number 2, which indicates OAEP-SHA1 padding.

`Encrypted Data`

Field 2, the data to be decrypted. The length will be twice the size of the RSA modulus (SOM). The value of the data must be less than the value of the RSA modulus, n. This field must contain hexadecimal characters.

`Key Slot or` $\texttt{AKB}_{\texttt{PRV}}$

Field 3, a key slot in the RSA key table where the RSA private key that will be used to generate the signature is stored. Command 120 or 121 must be used to store the private key in the RSA key table.

Or

$\texttt{AKB}_{\texttt{PRV}}$

Field 3, the private key, in AKB format, that will be used to generate the signature. The header of the AKB must be 1pRGE000.

Table 5-14 Command 138: RSA Decrypt Data

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	138
1	Padding Scheme	1	2
2	Encrypted Data	2*SOM	0 - 9, A -F
3	Key Slot	1-3	0 - 199
or			
3	AKB$_{PRV}$	Variable	printable ASCII

Responding Parameters

`238`

Field 0, the response identifier.

`Decrypted Data`

Field 1, the decrypted data. The minimum size will be 12 hexadecimal characters. When using padding scheme 2 (OEAP with SHA1), the maximum will be (length of modulus in bits) / 4 - 80 - 4. For a 2048-bit modulus, the maximum length will be 428 hexadecimal characters.

`[Check Digits]`

Field 2, The check digits of the RSA key pair. The check digits are defined to be the leftmost 8 characters of SHA1 of the RSA key. This field will contain 8 hexadecimal characters only when field 3 of the command is 0-199.

Table 5-15 Response 238: RSA Decrypt Data

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	238
1	Decrypted Data	varies	0 - 9, A - F
2	[Check Digits]	0, 8	0 - 9, A - F

Usage Notes

- Generate the AKB for the RSA private key.
- Magnetic stripe track data contains the hexadecimal value 0x1D, which is an unprintable ASCII character used as group separator.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- RSA encrypted Data:
  3212369DD9B8111F1AB432959ACF2513303745FA6AD847AB488B04AE1F5A4164966
  67A2507908763B3A46DD72143D72DCD4F1D5311F94A5A672566568BBCC1AE0D822
  FF4638F293DD5BC30166490DD1EFE1BEB859FA1E8909EDD4BD36E0AFCC17E9726
  DE1367E561EE1E19DF8FF97A5D4D8631F45506D981700F64476BC83407

- Public Key (in Atalla format):
  0001030080BFCD17064869B6EF638806300D3B620EA0A3EEC76A82A6E49363087
  2034D96B626A3B7891995748986F2400C41BCCA7204562E9498DA59D2E0DFDF50
  76CAA904918A148117EEC2C2A054247CAACCA2AF5A81CD04E6B17B04E8EF48A4F
  8883FC0845182CF7823E858F5471B3F8337144FC13F4959BD44DA1CE944BC0C47C3
  5EB9

- Private Key:
  1pRGE000,17067CDFB29AA51323D92EB9BE05741AC0E03C2BE2A94FE3E431E1CFB
  D4A2CAF479BE2D780B09813A4BAC27DE6955CA1D9958334E1474F2E3650775BF01
  16E6C07EC4891F781D7DE4BAAD806FE6B3CDA78F23BF90A4BE3D87196E9AA31521
  A81E3908AAF43C152C94AB5814EF466E73BD8F4746A60A7547D7B6940A027B5B1
  C11C258471A576C7E4A717544F9A31472FD547F7905904A785FADB60AB223E22C50
  E5D344D2B2669B09D114C895F26AA0B3A8FC59DEDC7909A349EAD41D38D9BB3C
  A6010455A930B88784C3E681624CC6B1168A9526C7A98B55C5DBEE51C4AFE5AA90
  85FA85F5F797E135BE772080140C81C58CFEABC853E09E33FDD1C3CD0099B9D16A
  E04D43C4A67EA313459D980F95A15D9517E143ABEDA2C4E72662C144C7F1317D747
  8BBC3DA2CBE8C6DFE200BD507224C3426C65A76B9A5492C6EB398633CE2767FA3
  FDC9ED6B82513ECD2BA9FBCA047BE28B700F9B0E15F862A3E4E1B52F20C9229E6
  E71BE220CCB27B110C4775C66E8D910EDFAAABF88AA99342F9B135AD40A2F972B
  CEF810B38AFF1F189D294032069F0F4DDD5C0D34B1B588AC124B9077B1D3BA09E
  F3B6B5008EFAEA0722CD7D23A40708DCB6BCF2716E8A1A83BBC858EABE2C9D0A
  D14269276C0C01797F1C86630A19E4643ECCD0F6BA017AC21592A0DFF9EEAC9F7F
  2629132522939C976744DD05E09CECD628D81A1D2C125FC1ADD6DFC05CC608736D
  B50C4ABD5D6702E01BA1A8CF6C8CFC4EB9D0674D9FF73C49FACCEB050CB2AA9
  B3AF0EB7D38A68ED9F1914EAD067E500FE98AF9140E01794DCED517896D4801F8
  CF3C2687DF23F9F01CEC7111362E90BF,5B42D1D0FBB8135D

The command looks like this:

```
<138#2#3212369DD9B8111F1AB432959ACF2513303745FA6AD847AB488B04AE1F5A
416496667A2507908763B3A46DD72143D72DCD4F1D5311F94A5A672566568BBCC1A
E0D822FF4638F293DD5BC30166490DD1EFE1BEB859FA1E8909EDD4BD36E0AFCC17E
9726DE1367E561EE1E19DF8FF97A5D4D8631F45506D981700F64476BC83407#1pRG
E000,17067CDFB29AA51323D92EB9BE05741AC0E03C2BE2A94FE3E431E1CFBD4A2C
AF479BE2D780B09813A4BAC27DE6955CA1D9958334E1474F2E3650775BF0116E6C0
7EC4891F781D7DE4BAAD806FE6B3CDA78F23BF90A4BE3D87196E9AA31521A81E390
8AAF43C152C94AB5814EF466E73BD8F4746A60A7547D7B6940A027B5B1C11C25847
```

```
1A576C7E4A717544F9A31472FD547F7905904A785FADB60AB223E22C50E5D344D2B
2669B09D114C895F26AA0B3A8FC59DEDC7909A349EAD41D38D9BB3CA6010455A930
B88784C3E681624CC6B1168A9526C7A98B55C5DBEE51C4AFE5AA9085FA85F5F797E
135BE772080140C81C58CFEABC853E09E33FDD1C3CD0099B9D16AE04D43C4A67EA3
13459D980F95A15D9517E143ABEDA2C4E72662C144C7F1317D7478BBC3DA2CBE8C6
DFE200BD507224C3426C65A76B9A5492C6EB398633CE2767FA3FDC9ED6B82513ECD
2BA9FBCA047BE28B700F9B0E15F862A3E4E1B52F20C9229E6E71BE220CCB27B110C
4775C66E8D910EDFAAABF88AA99342F9B135AD40A2F972BCEF810B38AFF1F189D29
4032069F0F4DDD5C0D34B1B588AC124B9077B1D3BA09EF3B6B5008EFAEA0722CD7D
23A40708DCB6BCF2716E8A1A83BBC858EABE2C9D0AD14269276C0C01797F1C86630
A19E4643ECCD0F6BA017AC21592A0DFF9EEAC9F7F2629132522939C976744DD05E0
9CECD628D81A1D2C125FC1ADD6DFC05CC608736DB50C4ABD5D6702E01BA1A8CF6C8
CFC4EB9D0674D9FF73C49FACCEB050CB2AA9B3AF0EB7D38A68ED9F1914EAD067E50
0FE98AF9140E01794DCED517896D4801F8CF3C2687DF23F9F01CEC7111362E90BF,
5B42D1D0FBB8135D#>
```

The Network Security Processor returns the following response:

```
<238#353434323938343434343434343434323D313331323130313132333435 1D42
3534343239383434343434343434343325E5443342F5E31333132313031313233343
52020313233202020##>
```

DES DUKPT Encrypt/Decrypt Data (Command 388)

Command 388 uses the Derived Unique Key Per Transaction (DUKPT) algorithm, a base derivation key, and a key serial number to generate the current key. A one-way function is applied to the current key to generate a session data key. This generated session data key is then used to either encrypt or decrypt data. Cipher Block Chaining (CBC) and Electronic Code Book (ECB) modes of 3DES are supported. The clear or encrypted data must be provided as ASCII hexadecimal characters. Binary data is not supported.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

Command

```
<388#Operation#Mode#Header,EMFK.E(BDK),MAC#Key Serial Number#[IV]#
Data#>
```

Response

```
<488#Data#Ending IV#Base Derivation Key Check Digits#
Data Key Check Digits#>[CRLF]
```

Calling Parameters

`388`

Field 0, the command identifier.

`Operation`

Field 1, indicates the operation to be performed on the data. This field contains one letter, either E to indicate encryption, or D to indicate decryption.

`Mode`

Field 2, the 3DES mode used to encrypt or decrypt the data. The following table indicates the modes that the Network Security Processor supports.

Value	Mode
0	Electronic Code Book (ECB)
1	Cipher Block Chaining (CBC)

Header,$E_{MFK.E}$(BDK),MAC

Field 3, the Base Derivation Key (BDK) encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The base derivation key must be a 2key-3DES (double-length) key or a 3key-3DES (triple-length) key. A replicated single-length key is not allowed. The following headers are supported in this command: 1dDNE000 or 1dDNN000.

`Key Serial Number`

Field 4, the key serial number used to generate the session data key. This field contains a 10 - 20 hexadecimal digit value.

`[IV]`

Field 5, the Initialization Vector.

This field must be empty when the mode is ECB (field 2 contains the number 0).

When the mode is CBC (field 2 contains the number 1), this field must contain 16 hexadecimal digits.

If the amount of data to be encrypted or decrypted exceeds the 4096 hexadecimal digit limit, the data must be split into segments. Each segment is sent in a separate command 388. All commands after the first command in the chain must contain the ending IV which was returned in field 2 of the response to the previous command 388.

`Data`

Field 6, the input data, clear-text or encrypted.

This field must contain clear-text data when field 1 contains the letter E. The length of clear-text data to be encrypted must be within the range of 2 - 4096 hexadecimal digits. If the length of the clear-text input data is not a multiple of 16 the Network Security Processor will right-pad the data with zeros such that the resulting length will be a multiple of 16.

This field must contain encrypted data when field 1 contains the letter D. The length of the encrypted data to be decrypted must be within the range of 16 - 4096 hexadecimal digits and be a multiple of 16.

If the amount of data to be encrypted or decrypted exceeds the 4096 hexadecimal digit limit, the data must be split into segments. Each segment is sent in a separate command 388. When encrypting data, the length of all segments except the last segment must be a multiple of 16, the last segment can be any length as long as it is not greater than 4096, this prevents the Network Security Processor from padding the intermediate segment data with zeros. When decrypting data, the length of all data segments must be a multiple of 16.

Table 5-16 Command 388: 3DES DUKPT Encrypt/Decrypt Data

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	388
1	Operation	1	D, E
2	Mode	1	0, 1
3	Header,$E_{MFK.E}$(BDK),MAC*	74	printable ASCII
4	Key Serial Number	10 - 20	0 - 9, A - F
5	IV	0, 16	0 - 9, A - F
6	Data	2 - 4096	0 - 9, A - F

* Can be a volatile table location.

Responding Parameters

`488`

Field 0, the response identifier.

`Data`

Field 1, the encrypted or clear-text data.

This field will contain clear-text data when field 1 of the command contains the letter D. The length of the clear-text data will be in the range of 16 - 4096 hexadecimal digits. It is the responsibility of the host application to validate/remove any padding.

This field will contain encrypted data when field 1 of the command contains the letter E. The length of the encrypted data will be in the range of 16 - 4096 hexadecimal digits.

`Ending IV`

Field 2, the ending initialization vector must be used as the starting IV for the next block of data if the amount of data to be encrypted or decrypted is greater than 4096 hexadecimal digits and the 3DES mode is CBC. This field contains a 16 hexadecimal digit value which is the last 16 hexadecimal digits of response field 1. It is included merely for convenience. If the mode is ECB this field is not relevant.

`Base Derivation Key Check Digits`

Field 3, check digits; that is, the first four hexadecimal digits that result from encrypting zeros using the Base Derivation Key. If option 88 is enabled, this field will contain six check digits.

`Data Key Check Digits`

Field 4, check digits; that is, the first four hexadecimal digits that result from encrypting zeros using the generated session data key. If option 88 is enabled, this field will contain six check digits.

Table 5-17 Response 488: 3DES DUKPT Encrypt/Decrypt Data

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	488
1	Data	2 - 4096	0 - 9, A - F
2	Ending IV	16	0 - 9, A - F
3	BDK Check Digits	4 or 6	0 - 9, A - F
4	Data Key Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- If you are encrypting or decrypting large amounts of data using the 3DES CBC mode, specify the ending Initialization Vector, returned in the response for the first block of data, to be the starting Initialization Vector in the encryption or decryption for the next block of data. Be sure to specify the same Base Derivation Key, Key Serial Number and mode for each data block.
- Generate the AKB for the Base Derivation Key.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Encryption

- Operation: E
- Mode: Cipher block chaining (1)
- Base Derivation Key: 0123456789ABCDEF FEDCBA9876543210,
  check digits 08D7
  The Base Derivation Key in AKB format:
  1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,342946FE
  884AA8B2
- Key Serial Number: FFFF9876543210E00001
- IV: 0000000000000000

- Data: 4E6F772069732074495D0F96DCF42F40

The command looks like this:

```
<388#E#1#1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,
342946FE884AA8B2#9876543210E00001#0000000000000000#4E6F772069732074
495D0F96DCF42F40#>
```

The Network Security Processor returns the following response:

```
<488#98EFA6D1AAC43A805A0B7F205A8808E1#5A0B7F205A8808E1#08D7#156B#>
```

Decryption

- Operation: D
- Mode: Cipher block chaining (1)
- Base Derivation Key: 0123456789ABCDEF FEDCBA9876543210,
  check digits 08D7
  The Base Derivation Key in AKB format:
  1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,342946FE
  884AA8B2
- Key Serial Number: FFFF9876543210E00001
- IV: 0000000000000000
- Data: 98EFA6D1AAC43A805A0B7F205A8808E1

The command looks like this:

```
<388#D#1#1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,
342946FE884AA8B2#9876543210E00001#0000000000000000#98EFA6D1AAC43A80
5A0B7F205A8808E1#>
```

The Network Security Processor returns the following response:

```
<488#4E6F772069732074495D0F96DCF42F40#495D0F96DCF42F40#08D7#156B#>
```

Encrypt/Decrypt Data using AES (Command 390)

Command 390 uses the Advanced Encryption Standard algorithm to either encrypt or decrypt data. The Cipher Block Chaining (CBC) mode is supported. Key sizes of 128-, 192-, and 256-bits are allowed.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<390#Operation#Mode#Header,EMFK.E(KD),MAC#[IV]#Data Type#Length#
Data#Check Digit Method#>
```

Response

```
<490#Operation#Mode#Check Digits#IV#Header,EMFK.E(Ending IV),MAC#
Data Type#Length#Data#>[CRLF]
```

Calling Parameters

`390`

Field 0, the command identifier.

`Operation`

Field 1, the operation to be performed on the data. The following table lists the operation that the Network Security Processor can perform and the value to enter in this field for each operation.

Value	Operation
D	Decryption
E	Encryption

`Mode`

Field 2, the AES mode used to encrypt or decrypt the data. Only Cipher Block Chaining (CBC) is supported. This field must contain the value 1.

Header,$E_{MFK.E}$(KD),MAC

Field 3, the Data Key (KD) encrypted under the MFK. This field contains a 58, 74 or 90 byte value which is required to support a 128-, 192- or 256-bit key. Only Data Keys with the following headers are supported: 1DJNE000, 1DJNN000, 1DJEE000, 1DJEN000, 1DJDE000, and 1DJDN000.

`[IV]`

Field 4, the Initialization Vector (IV) encrypted under the MFK.

If the operation is Encryption, this field can be:

- Empty, in which case the Network Security Processor will generate a random IV.
- The letter “D”, which instructs the Network Security Processor to use a null IV (binary zeros).
- A cleartext 32 hexadecimal character IV.
- A 58 byte value of the IV encrypted under the MFK. The header for the encrypted IV must be either 1IJNE000 or 1IJNN000.

If the operation is Decryption, this field can be:

- The letter “D”, which instructs the Network Security Processor to use a null IV (binary zeros).
- A cleartext 32 hexadecimal character IV.
- A 58 byte value of the IV encrypted under the MFK. The header for the encrypted IV must be either 1IJNE000 or 1IJNN000.

`Data Type`

Field 5, the data type. The following table indicates the type of data that the Network Security Processor supports and the value to enter in this field for each type.

Value	**Data Type**
U	Unpacked ASCII-hexadecimal
B	Binary

See Data formats for more information on data types and cautions on using Binary data.

`Length`

Field 6, the data’s length. This command will encrypt or decrypt up to 4096 bytes of data when using type ‘B’. For type ‘U’, the command will process between 2 and 4096 characters of data (the length must be divisible by 2). For encryption, if the input length is not a multiple of the AES block size (32 characters for Data type U or 16 bytes for type B), the Network Security Processor will right pad the data with zeros. For decryption, the input length must be a multiple of the block size.

`Data`

Field 7, the input data, encrypted or in clear-text format. The length of this field must match what was specified in field 6. If the amount of data to be encrypted or decrypted exceeds the 4096 byte limit, the data must be split into segments. Each segment is sent in a separate command. All commands after the first command in the chain must contain the ending IV which was returned in field 2 of the response.

`Check Digit Method`

Field 8, the method used to calculate the check digits of the Data Key. The following table indicates the method that the Network Security Processor supports and the value to enter in this field for each method.

Value	Method
C	CMAC$_{KD}$(Block of zeros)
H	SHA-256 (KD)

Table 5-18 Command 390: Encrypt/Decrypt Data using AES

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	390
1	Operation	1	D, E
2	Mode	1	1
3	Header,E$_{MFK.E}$(KD),MAC	58, 74, 90	printable ASCII
4	[IV]	0, 32, 58	printable ASCII
5	Data Type	1	B or U
6	Length	1 - 4	0 - 9
7	Data	2 - 4096	Binary = any Unpacked ASCII = 0 - 9, A - F
8	Check Digit Method	1	C or H

Responding Parameters

`490`

Field 0, the response identifier.

`Operation`

Field 1, the operation performed on the data. This field will contain the same value as field 1 of the command.

`Mode`

Field 2, the mode of operation. This field will contain the same value as field 2 of the command.

`Check Digits`

Field 3, the check digits of the Data Key. The check digits are generated using the algorithm specified in field 8 of the command. If the check digit algorithm is C (CMAC), this field will contain 10 hexadecimal characters. If the check digit algorithm is H (SHA-256), this field will contain 64 hexadecimal characters.

`IV`

Field 4, the Initialization Vector (IV) used in the encryption or decryption operation. The contents of this field depends on the contents of field 4 of the command.

Field 4 of the command	**Response field 4 value**
Empty	Header,$E_{MFK.E}$(Random IV),MAC
D	D
Cleartext IV	Cleartext IV
Encrypted IV	Encrypted IV

Header,$E_{MFK.E}$(Ending IV),MAC

Field 5, the ending IV encrypted under the MFK. This field contains a 58 byte value. The ending IV must be used as the starting IV for the next block of data if the amount of data to be encrypted or decrypted is greater than 4096 hexadecimal digits.

`Data Type`

Field 6, the data type. This field will contain the same value as field 5 of the command.

`Length`

Field 7, the length of the data in field 8 of the response. The minimum length is 16 and the maximum length is 4096. For encryption, this value may be longer than the input data due to padding. For decryption, the output data length is always the same as the input length; the Network Security Processor does not remove any padding bytes.

`Data`

Field 8, the cleartext or encrypted output data. The length of this field is defined by the value in response field 7.

Table 5-19 Response 490: Encrypt/Decrypt Data using AES

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Response identifier	3	490
1	Operation	1	D, E

Table 5-19 Response 490: Encrypt/Decrypt Data using AESsn(continued)

Field	Contents	Length (bytes)	Legal Characters
2	Mode	1	1
3	KD Check Digits	10 or 64	0 - 9, A - F
4	IV	1, 32 or 58	0 - 9, A - F
5	Header,$E_{MFK.E}$(Ending IV),MAC	58	Printable ASCII
6	Data Type	1	B, U
7	Length	1 - 4	0 - 9
8	Data	16 - 4096	Binary = any Unpacked ASCII = 0 - 9, A - F

Usage Notes

- Before using this command, generate the AKB for the Data Key.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Encryption

- Operation: E
- Mode: Cipher block chaining (1)
- Data Key (128 bits): 0123456789ABCDEFFEDCBA9876543210
  The Data Key in AKB format:
  1DJNE000,454EB64571576C185EA9D338840CD76F,D669BF4E9D039D94
- IV: D - binary zeros
- Type: Unpacked ASCII hexadecimal
- Length: 16
- Data: 1234567890123456
- Check Digit Method: H - SHA-256$_{(KD)}$

The command looks like this:

```
<390#E#1#1DJNE000,454EB64571576C185EA9D338840CD76F,D669BF4E9D039D94
#D#U#16#1234567890123456#H#>
```

The Network Security Processor returns the following response:

```
<490#E#1#411D3F1D2390FF3F482AC8DF4E730780BB081A192F283D2F373138FD10
1DC8FE#D#1IJNE000,C4AA24218243CF0DE3611C294C0B4FB9,573A3C7BA67D7684
#U#32#F31B20110CACF9CCA4B6A2BD632FE257#>
```

Decryption

- Operation: D
- Mode: Cipher block chaining (1)
- Data Key (128 bits): 0123456789ABCDEFFEDCBA9876543210
  The Data Key in AKB format:
  1DJNE000,454EB64571576C185EA9D338840CD76F,D669BF4E9D039D94
- IV: D - binary zeros
- Type: Unpacked ASCII hexadecimal
- Length: 32
- Data: F31B20110CACF9CCA4B6A2BD632FE257
- Check Digit Method: H - SHA-256$_{(KD)}$

The command looks like this:

```
<390#D#1#1DJNE000,454EB64571576C185EA9D338840CD76F,D669BF4E9D039D94
#D#U#32#F31B20110CACF9CCA4B6A2BD632FE257#H#>
```

The Network Security Processor returns the following response:

```
<490#D#1#411D3F1D2390FF3F482AC8DF4E730780BB081A192F283D2F373138FD10
1DC8FE#D#1IJNE000,C4AA24218243CF0DE3611C294C0B4FB9,573A3C7BA67D7684
#U#32#12345678901234560000000000000000#>
```

6

Authenticating Transaction Data

Data authentication is the process of verifying transmitted data to be sure that it has not been altered during transmission. Authentication thus ensures data integrity. This section outlines the tasks involved in authenticating data.

Federal information Processing Standard 113, ANSI X9.19, and ISO 9797 provide detailed information on Message Authentication.

To skip this introduction, go to Table 6-1 on page 6-4 for a list of commands.

About Data Authentication

A Message Authentication Code (MAC) is used to validate that data has not been altered. The node sending the data, generates a MAC by applying a special, predefined algorithm and a data authentication key to a block of data, the result is the MAC. The data and MAC are sent to the receiving node. The receiving node applies the same algorithm and key to compute a MAC for the data it receives. If the computed MAC matches the received MAC, the data has not been altered during transmission.

The Network Security Processor can authenticate an unlimited amount of data. If you will be sending or verifying more than 4096 bytes, you must also send the data in more than one batch. Sending data in one or multiple batches is explained in Authentication in Batches.

Data Authentication Tasks

Authenticating data typically involves the following tasks:

- Generating the MAC to be transmitted with the data.
- Verifying the MAC at the receiving end.

Whether you are generating or verifying a MAC, the steps involved are the same: generate a KMAC key, and specify its cryptogram as a parameter in the appropriate MAC generate or MAC verify command. The Network Security Processor response contains either the MAC or a verification flag. If you are sending or authenticating a large volume of data – necessitating the use of the MAC continuation commands – the Network Security Processor does not return a MAC or verification flag until you finish sending or verifying data.

Data authentication can be expressed as the following function:

Authenticated data = ƒ(data, IV)

In other words, authentication is a function of data and an Initialization Vector. The following section explains how Initialization Vectors are used in data authentication.

Using Initialization Vectors

Initialization vectors are used in data authentication when more than 4096 bytes of data are being authenticated. An Initialization Vector of binary zeros is used when fewer than 4096 bytes of data are being authenticated.

The following examples illustrate two ways that the Network Security Processor can be used to send or authenticate the data string, “This is an idea.” and the role that Initialization Vectors play in each case.

Authentication All at Once

The first way the string can be sent or authenticated is all at once. Thus, the starting string is, “This is an idea.” The starting Initialization Vector is the Network Security Processor default IV, which is all zeros.

ƒ(This is an idea.)

If you are generating a MAC, the result is a MAC and the ending Initialization Vector. If you are verifying a MAC, the Network Security Processor returns a verification flag and the ending Initialization Vector.

Generating: (MAC, E_{IV} = X)

Verifying: (Flag – Y or N, E_{IV} = X)

This method of authenticating data is sufficient when you are using Command 98 to send messages that contain fewer than 4096 bytes of data.

Authentication in Batches

The second way the string can be sent or authenticated is in batches. In this example, the starting string is, “This is” and the starting Initialization Vector is the Network Security Processor default IV, which is all zeros.

ƒ(This is)

The result is the Initialization Vector, Z. The Network Security Processor only returns a MAC or verification flag when it has finished receiving or authenticating all of your data.

(E_{IV} = Z)

To send or authenticate the rest of the string, you supply the remainder of the sentence and Z, the Initialization Vector obtained when you authenticated the first part of the string.

f(an idea., E_{IV} = Z)

If you are on the sending node – thus, generating a MAC, the result is a MAC and the ending Initialization Vector, X. If you are verifying a MAC, the Network Security Processor returns a verification flag and the ending Initialization Vector.

Generating: (MAC, E_{IV} = X)

Verifying: (Flag -- Y or N, E_{IV} = X)

This method of sending or authenticating data – in batches – must be used when you are using Command 98 to send messages that contain more than 4096 bytes of data.

note When authenticating data in batches, the length of the data authenticated in each batch – except for the last batch – must be a multiple of eight bytes. The length of the last batch of data authenticated is not restricted.

Verification in VISA UKPT Networks

VISA UKPT (Unique Key Per Transaction) key management uses MACs, but implements them a little differently from the process just described. Specifically, VISA UKPT requires three authentication codes: MAC one, MAC two, and MAC three. MAC one authenticates the transaction data received from the PIN pad. The host calculates and returns MAC two if the transaction is approved. If the transaction is denied, the host returns MAC three. See Verify and Generate MAC for VISA UKPT (Command 5C) for the command syntax.

Data Authentication Commands

The rest of this section contains the command and response syntax for the Network Security Processor data authentication commands.

Quick Reference

The following identifies each command by number, name, and purpose. While Table 6-1 organizes the message authentication commands by category, the commands themselves are presented in numerical order.

Table 6-1 Data Authentication Commands

Command	Name	Purpose
MAC generation commands		
59	Generate MAC and Encrypt or Translate Data	Generates a Message Authentication Code and can either encrypt or translate data.
98	Generate Message Authentication Code	Generates a Message Authentication Code.
305	Generate MAC using CMAC-TDES	Generates a CMAC using the TDES algorithm.
386	Generate DUKPT MAC	Generates a Message Authentication Code using a derived key.
39B	Generate MAC using HMAC	Generates a Message Authentication Code using either the HMAC-SHA1 or HMAC-SHA256 algorithm.
MAC Translation command		
58	MAC Translate	Verifies a Message Authentication Code, and then generates a Message Authentication Code using a different key.
MAC verification commands		
5C	Verify & Generate MAC for VISA UKPT	Verifies a Message Authentication Code and generates an approval or denial Message Authentication Code.
5F	Verify MAC and Decrypt PIN	Verifies a Message Authentication Code and decrypts a PIN.
99	Verify Message Authentication Code	Verifies a Message Authentication Code.
9B	Verify ACR Response MAC	Verifies a Message Authentication Code from an Atalla Challenge Response Unit.
304	Verify CMAC using TDES	Verify a CMAC using the TDES algorithm.
348	Verify DUKPT MAC	Verifies a Message Authentication Code using a derived key.
39C	Verify MAC using HMAC	Verifies a Message Authentication Code using either the HMAC-SHA1 or HMAC-SHA256 algorithm.
APACS 40 commands		
324	Receive APACS 40 Request Message	Receives an APACS 40 Request Message
325	Generate APACS 40 Response Message	Generates an APACS 40 Response Message
326	Verify APACS 40 Confirmation Message	Verifies an APACS 40 Confirmation Message

MAC Translate (Command 58)

Command 58 translates a Message Authentication Code from one key to another key. It first verifies an incoming MAC using the incoming KMAC key designated as KMAC-I, and if successful, generates a MAC using the outgoing KMAC key designated as (KMAC-O.)

This command can also be used to either only verify a MAC, or only generate a MAC. If this command will only be used to verify a MAC, command fields 6, 7, 8 and 9 must be empty. If this command will only be used to generate a MAC, command fields 1, 2, 3, 4, and 5 must be empty.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

This command is not enabled in the Network Security Processor’s default security policy. To use this command, you must enable it in the Network Security Processor’s security policy.

Command

```
<58#[Header,EMFK.E(KMAC-I),MAC]#[MAC-I Length]#
[Header,EMFK.E(IV-I),MAC]#Reserved#[MAC-I]#
[Header,EMFK.E(KMAC-O),MAC]#[MAC-O Length]#
[Header,EMFK.E(IV-O),MAC]#Reserved#Data Type#Data Length#Data#>
```

Response

```
<68#[MAC Length-I]#[Verification Flag or
Header,EMFK.E(Ending IV-I),MAC]#[KMAC-I Check Digits]#
[MAC Length-O]#[MAC or Header,EMFK.E(Ending IV-O),MAC]#
[KMAC-O Check Digits]#>[CRLF]
```

Calling Parameters

`58`

Field 0, the command identifier.

`[Header,EMFK.E(KMAC-I),MAC]`

Field 1, the incoming KMAC key (KMAC-I) encrypted under the MFK. This field contains a 74 byte value, a volatile table location, or is empty. The following headers are supported: 1MDNE000, 1MDNN000, 1MDVE000, and 1MDVN000.

If this field is empty, fields 2, 3, 4 and 5 must also be empty.

`[MAC-I Length]`

Field 2, the size of the MAC to be verified. The following table indicates the possible MAC sizes and the values to enter in this field.

Value	MAC Size
0	More data expected, no MAC verified
1	32 bits
2	48 bits
3	64 bits

A 32 bit MAC is expressed as eight hexadecimal digits (0-9, A-F) and displayed as two groups of four digits, separated by a space.
A 48 bit or 64 bit MAC is expressed as three or four groups of four hexadecimal digits, separated by a space.

This field can contain a 1 byte decimal value or is empty.

If this field contains a 0 (zero), field 7 must also contain a 0 (zero). If this field is empty, fields 1, 3, 4 and 5 must be empty.

`[Header,`$E_{MFK.E}$`(IV-I),MAC]`

Field 3, the incoming Initialization Vector (IV-I), used in the verification of the MAC, encrypted under the MFK.

If this command contains the first block of multiple blocks of data, this field must be empty.

If this command contains data subsequent to the first block in a multiple block series (that is, it contains continuation data), this field should contain the ending Initialization Vector from the previously sent data block. This field must be empty if any of the fields 1, 2, 4 and 5 are empty.

This field contains a 74 byte value, or is empty. The following headers are supported: 1IDNE000 and 1IDNN000.

`Reserved`

Field 4, this field is reserved, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

`[MAC-I]`

Field 5, the incoming MAC to be verified. This field must empty if more data is coming in subsequent commands.

A 32 bit MAC is expressed as eight hexadecimal digits, it is displayed as two groups of four digits, separated by a space.

A 48 bit or 64 bit MAC is expressed as three or four groups of four hexadecimal digits, separated by a space.

Field 5 should be empty, or its length should be 9 bytes (8 characters plus 1 space), 14 bytes (12 characters plus 2 spaces), or 19 bytes (16 characters plus three spaces).

If field 2 contains a zero, this field must be empty. If this field is empty, fields 1, 2, 3 and 4 must also be empty.

`[Header,E`$_{MFK.E}$`(KMAC-O),MAC]`

Field 6, the outgoing KMAC Key (KMAC-O) encrypted under the MFK. This key is used to generate the outgoing MAC. If this field is empty, fields 7, 8 and 9 must also be empty.

This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1MDNE000, 1MDNN000, 1MDGE000, and 1MDGN000.

`[MAC-O Length]`

Field 7, the length of the outgoing MAC. The following table indicates the possible outgoing MAC lengths and the value to enter in this field.

Value	Outgoing MAC length
0	More data expected, no MAC returned
1	32 bits
2	48 bits
3	64 bits

A 32 bit MAC is expressed as eight hexadecimal digits (0-9, A-F) and displayed as two groups of four digits, separated by a space.
A 48 bit or 64 bit MAC is expressed as three or four groups of four hexadecimal digits, separated by a space.

If field 2 contains 0, this field also must be 0. If field 7 is empty, fields 6, 8 and 9 must be empty.

`[Header,E`$_{MFK.E}$`(IV-O),MAC]`

Field 8, the outgoing Initialization Vector (IV-O) encrypted under the MFK. This IV-O is used in the generation of the outgoing MAC

If this command contains the first block of multiple blocks of data, this field must be empty.

If this command contains data subsequent to the first block in a multiple block series (that is, it contains continuation data), this field contains the ending Initialization Vector from the previously sent data block.

This field contains a 74 byte value. The following headers are supported: 1IDNE000 and 1IDNN000. If field 3 is empty, this field also must be empty. This field must be empty if fields 6, 7 or 8 are empty.

`Reserved`

Field 9, this field is reserved, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

`Data Type`

Field 10, the data type. The following table indicates the type of data that the Network Security Processor supports.

Value	Data Type
U	Unpacked ASCII hexadecimal
B	Binary

`Data Length`

Field 11, the data length. This command authenticates up to 4096 bytes of data.

If more data is being sent in the next command, the data length must be a multiple of eight for binary data, and a multiple of 16 for Unpacked ASCII data (batch authentication is indicated when field 2 is set to 0, see Authentication in Batches for additional information).

If data sent is not batched, the Network Security Processor will pad the data field will be binary zeros to a multiple of eight.

This field contains a 1 to 4 byte decimal value.

`Data`

Field 12, the input data. This field can be from 1 to 4096 bytes long. If the data is in unpacked ASCII hexadecimal format, this field can contain the numbers 0 to 9 and characters A to F.

Table 6-2 Command 58: MAC Translate

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	58
1	[Header, $E_{MFK.E}$(KMAC-I), MAC]*	0, 74	printable ASCII
2	[MAC-I Length]	0, 1	0 - 3
3	[Header,$E_{MFK.E}$(IV-I),MAC]	0, 74	printable ASCII
4	Reserved	0 - 2	3, 19
5	[MAC-I]	0, 9, 14, 19	0 - 9, A - F, “ ”
6	[Header,$E_{MFK.E}$(KMAC-O),MAC]*	0, 74	printable ASCII
7	[MAC-O Length]	0, 1	0 - 3
8	[Header,$E_{MFK.E}$(IV-O),MAC]	0, 74	printable ASCII

Table 6-2 Command 58: MAC Translate (continued)

Field	Contents	Length (bytes)	Legal Characters
9	Reserved	0 - 2	3, 18
10	Data Type	1	U, B
11	Data Length	1 - 4	0 - 9
12	Data	1 - 4096	0 - 9, A - F if unpacked ASCII

* Can be a volatile table location.

Responding Parameters

`68`

Field 0, the response identifier.

`MAC Length-I`

Field 1, the length of the incoming MAC.

If this field is set to 0, more data is expected, and field 2 of the response will contain the ending Initialization Vector.

If this field is set to 1, 2, or 3, field 2 will contain the MAC verification flag.

`Verification Flag or Header,`$E_{MFK.E}$`(Ending IV-I),MAC`

Field 2, either verification flag, or if field 1 of the response is 0, the ending Initialization Vector encrypted under the MFK.

If your use of this command results in the generation of an ending Initialization Vector in this field, use it as the starting Initialization Vector in the subsequent MAC command to continue generating MACs.

If your use of this command results in a MAC verification flag, this field will return Y if the MAC is verified, or N if the MAC is not verified.

This field will be empty if fields 1, 2, 3, 4 and 5 in the command are empty (that is, this command will only generate a MAC).

`KMAC-I Check Digits`

Field 3, check digits, the first four digits that result from encrypting zeros using the incoming KMAC key. If option 88 is enabled, this field will contain six check digits.

This field will be empty if fields 1, 2, 3, 4 and 5 in the command are empty.

`MAC Length-O`

Field 4, the length of the generated outgoing MAC.

If this field is set to 0, more data is expected, and field 2 will contain the ending Initialization Vector.

If this field is set to 1, 2, or 3, field 2 will contain the MAC.

`MAC or Header,`$E_{MFK.E}$`(Ending IV-O),MAC`

Field 5, either verification flag, or if field 1 of the response is 0, the Initialization Vector used in the MAC generation process.

If your use of this command results in the generation of an ending Initialization Vector in this field, use it as the starting Initialization Vector in the subsequent MAC command to continue the generation of the MAC. Otherwise, this field will have generated MAC if the incoming MAC is verified, or it will be empty.

This field is contains either a zero byte, 9 byte, 14 byte, 16 byte or 19 byte hexadecimal value, as well as spaces.

If the verification flag (field 2) is N, this field is empty. This field will be empty if fields 6, 7, 8 and 9 in the command are empty.

`KMAC-O Check Digits`

Field 6, check digits, the first four digits that result from encrypting zeros using the outgoing KMAC key. If option 88 is enabled, this field will contain six check digits.

If the verification flag (field 2) is N, this field will be empty.

This field will be empty if fields 6, 7, 8 and 9 in the command are empty.

Table 6-3 Response 68: MAC Translate

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	68
1	MAC Length-I	0, 1	0 - 3
2	Verification flag or Header,$E_{MFK.E}$(Ending IV-I),MAC	0, 1, 74	0 - 9, A - F, Y, N
3	KMAC-I Check Digits	0, 4 or 6	0 - 9, A - F
4	MAC Length-O	0, 1	0 - 3
5	MAC or Header,$E_{MFK.E}$(Ending IV-O),MAC	0, 9, 14, 19, 74	0 - 9, A - F, “ ”
6	KMAC-O Check Digits	0, 4 or 6	0 - 9, A - F

Usage Notes

- Generate the incoming and outgoing KMAC keys.
- If fields 1, 2, 3, 4 and 5 in this command are empty, this command will only generate a MAC.
- If fields 6, 7, 8 and 9 are empty, this command will only verify a MAC.
- This command can contain a binary data field. The Network Security Processor's and the host's serial interface must both be configured to 8 data bits, 1 stop bit, and no parity to correctly process binary data.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Generate a MAC

- Clear-text outgoing KMAC key (KMAC-O): FEDC BA98 7654 3210
  The outgoing KMAC key (KMAC-O) in AKB format:
  1MDNE000,EEC942756FEE2710A60A4410F2A12215153C05BC7A4A2B9B,96A475A13
  7989605
- MAC length: 64 bits (3)
- Clear-text outgoing Initialization Vector (IV-O): 2558 8552 2558 8552
  The outgoing Initialization Vector (IV-O) in AKB format:
  1IDNE000,F5034A71376938873CE42E8CBEC1657C117F76D1FC937DFC,D6EAA159C1F
  DC638
- Data type: Unpacked ASCII hexadecimal (U)
- Data length: 8 bytes
- Data: 01AB8D89

The command looks like this:

```
<58######1MDNE000,EEC942756FEE2710A60A4410F2A12215153C05BC7A4A2B9B,
96A475A137989605#3#1IDNE000,CE6518F12DABFE5BFBD04D67D95ACABFD765750
4C0173D0D,0153727BB483E4FB##U#8#01AB8D89#>
```

The Network Security Processor returns the following response:

```
<68####3#0299 23CE A64A D1B0#A68C#>
```

Verify a MAC with data in Unpacked ASCII format

- Clear-text incoming KMAC key (KMAC-I): FEDC BA98 7654 3210
  The incoming KMAC key (KMAC-I) in AKB format:
  1MDNE000,EEC942756FEE2710A60A4410F2A12215153C05BC7A4A2B9B,96A475A13
  7989605
- MAC length: 64 bits (3)
- Clear-text outgoing Initialization Vector (IV-O): 2558 8552 2558 8552
  The outgoing Initialization Vector (IV-O) in AKB format:
  1IDNE000,F5034A71376938873CE42E8CBEC1657C117F76D1FC937DFC,D6EAA159C1F
  DC638
- MAC: 78FA FA86 68CF 1FC7
- Data type: Unpacked ASCII hexadecimal (U)
- Data length: 6 bytes
- Data: 303430

The command looks like this:

```
<58#1MDNE000,EEC942756FEE2710A60A4410F2A12215153C05BC7A4A2B9B,96A47
5A137989605#3#1IDNE000,CE6518F12DABFE5BFBD04D67D95ACABFD7657504C017
3D0D,0153727BB483E4FB##78FA FA86 68CF 1FC7#####U#6#303430#>
```

The Network Security Processor returns the following response:

```
<68#3#Y#A68C####>
```

Generate MAC and Encrypt or Translate Data (Command 59)

Command 59 generates a MAC and encrypts or translates data. This command supports ECB and CBC modes of DES.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

This command is not enabled in the Network Security Processor's default security policy. To use this command, enable it in Network Security Processor's security policy.

Multiple Mode Command

Command – ECB-Mode Encryption

```
<59#0#Header,$E_{MFK.E}$(KMAC),MAC#MAC Data##Header,$E_{MFK.E}$($KD_O$),MAC#
Clear data#>
```

Command – CBC-Mode Encryption

```
<59#0#Header,$E_{MFK.E}$(KMAC),MAC#MAC Data##Header,$E_{MFK.E}$($KD_O$),MAC###
[Header,$E_{MFK.E}$($IV_O$),MAC]#Data Type#Length#Clear data#>
```

Command – ECB-Mode Translation

```
<59#0#Header,$E_{MFK.E}$(KMAC),MAC#MAC Data#Header,$E_{MFK.E}$($KD_I$),MAC#
Header,$E_{MFK.E}$($KD_O$),MAC#Encrypted data#>
```

Command – CBC-Mode Translation

```
<59#0#Header,$E_{MFK.E}$(KMAC),MAC#MAC Data#Header,$E_{MFK.E}$($KD_I$),MAC#
Header,$E_{MFK.E}$($KD_O$),MAC##[Header,$E_{MFK.E}$($IV_I$),MAC]#
[Header,$E_{MFK.E}$($IV_O$),MAC]#Data Type#Length#Encrypted data#>
```

Multiple Mode Response

Response – ECB-Mode

```
<69#MAC#KMAC Check Digits#Encrypted Data#[Incoming KD Check Digits]#
Outgoing KD Check Digits#>[CRLF]
```

Response – CBC-Mode

```
<69#MAC#KMAC Check Digits##[$KD_I$ Check Digits]#$KD_O$ Check Digits#
[Header,$E_{MFK.E}$($IV_I$),MAC]#Header,$E_{MFK.E}$($IV_O$),MAC#Data Type#Length#
Encrypted data#>[CRLF]
```

Calling Parameters – ECB-Mode Encryption

`59`

Field 0, the command identifier.

`0`

Field 1, the data continuation flag, must be set to 0.

Header,$E_{MFK.E}$(KMAC),MAC

Field 2, the KMAC key encrypted under the MFK. This key is used to generate the MAC. This field contains either a 74 byte value or a volatile table location. The following headers are supported: 1MDNE000, 1MDNN000, 1MDGE000, and 1MDGN000.

`MAC Data`

Field 3, data to be authenticated. This field can be from one to 240 bytes long and can contain the characters A to Z, the numbers 0 to 9, and “, ”, “.”, and “ ”.

`Reserved`

Field 4, this field must be empty.

Header,$E_{MFK.E}$(KD_O),MAC

Field 5, the outgoing Data Key encrypted under the MFK. This key is used to encrypt the data contained in field 6.

This field contains a 74 byte value or a volatile table location. The following headers are supported: 1DDNE000, 1DDNN000, 1DDEE000, and 1DDEN000.

`Clear Data`

Field 6, the clear data to be encrypted using the outgoing Data Key using the Electronic Code Book (ECB) method of DES.

Data less than 16 characters must be right padded with zeros such that the data length is 16 hexadecimal characters.

Table 6-4 Command 59: ECB-Mode Encryption

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	59
1	Data continuation flag	1	0
2	Header,$E_{MFK.E}$(KMAC),MAC*	74	printable ASCII
3	MAC Data	1 - 240	0 - 9, A - Z, , . “ ”

Table 6-4 Command 59: ECB-Mode Encryption (continued)

Field	Contents	Length (bytes)	Legal Characters
4	Reserved	0	
5	Header,$E_{MFK.E}$(KD_O),MAC*	74	printable ASCII
6	Clear Data	16	0 - 9, A - F

* Can be a volatile table location.

Calling Parameters – CBC-Mode Encryption

`59`

Field 0, the command identifier.

`0`

Field 1, the data continuation flag, must be set to zero.

Header,$E_{MFK.E}$(KMAC),MAC

Field 2, the KMAC key encrypted under the MFK. This key is used to generate the MAC. This field contains either a 74 byte value or a volatile table location. The following headers are supported: 1MDNE000, 1MDNN000, 1MDGE000, and 1MDGN000.

`MAC Data`

Field 3, data to be authenticated. This field can be from one to 240 bytes long and can contain the characters A to Z, the numbers 0 to 9, and “,”, “.”, and “ ”.

`Reserved`

Field 4, a reserved field, it must be empty.

Header,$E_{MFK.E}$(KD_O),MAC

Field 5, the outgoing Data Key encrypted under the MFK. This key is used to encrypt the data contained in field 11. This field contains either a 74 byte value, or a volatile table location. The following headers are supported: 1DDNE000, 1DDNN000, 1DDEE000, and 1DDEN000.

`Reserved`

Field 6, a reserved field, it must be empty.

`Reserved`

Field 7, a reserved field, it must be empty.

`[Header,`$E_{MFK.E}(IV_O)$`,MAC]`

Field 8, the Initialization Vector encrypted under the MFK. This Initialization Vector is used in the outgoing CBC data encryption process. If this field is empty, the default Initialization Vector of all zeros will be used. This field contains a 74 byte value or is empty. The following headers are supported: 1IDNE000 and 1IDNN000.

`Data Type`

Field 9, the type of the data in field 11:

Value	Data Type
U	Unpacked ASCII hexadecimal
B	Binary

`Length`

Field 10, the length of the data in field 11. This field contains a 1 to 4 byte decimal value.

`Clear Data`

Field 11, the clear data to be encrypted using the outgoing data encryption key using the Cipher Block Chaining (CBC) method of encryption.

This field is from one to 3500 bytes long if the data is in binary format. If the data is in unpacked ASCII hexadecimal format, this field is two to 3500 bytes long and must be a multiple of 2.

The Network Security Processor will pad the data with binary zeros to achieve a value that is an eight byte multiple (binary data) or a 16 byte multiple (unpacked ASCII data).

Table 6-5 Command 59: CBC-Mode Encryption

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	59
1	Data continuation flag	1	0
2	Header,$E_{MFK.E}$(KMAC),MAC*	74	printable ASCII
3	MAC Data	1 - 240	0 - 9, A - Z, , . “ ”
4	Reserved	0	
5	Header,$E_{MFK.E}(KD_O)$,MAC*	74	printable ASCII
6	Reserved	0	
7	Reserved	0	

Table 6-5 Command 59: CBC-Mode Encryption (continued)

Field	Contents	Length (bytes)	Legal Characters
8	Header,$E_{MFK.E}(IV_O)$,MAC	0, 74	printable ASCII
9	Data type	1	U, B
10	Length	1 - 4	0 - 9
11	Clear Data	1 - 3500 or 2 - 3500	0 - 0, A - F (if unpacked ASCII hexadecimal

* Can be a volatile table location.

Calling Parameters – ECB-Mode Translation

`59`

Field 0, the command identifier.

`0`

Field 1, the data continuation flag, must be set to 0.

Header,$E_{MFK.E}$(KMAC),MAC

Field 2, the KMAC key encrypted under the MFK. This key is used to generate the MAC. This field contains either a 74 byte value or a volatile table location. The following headers are supported: 1MDNE000, 1MDNN000, 1MDGE000, and 1MDGN000.

`MAC Data`

Field 3, data to be authenticated. This field can be from one to 240 bytes long and can contain the characters A to Z, the numbers 0 to 9, and “,”, “.”, and “ ”.

Header,$E_{MFK.E}(KD_I)$,MAC

Field 4, the incoming Data Key encrypted under the MFK. This key is used to decrypt the data in field 6. This field contains a 74 byte value or a volatile table location. The following headers are supported: 1DDNE000, 1DDNN000, 1DDDE000, and 1DDDN000.

Header,$E_{MFK.E}(KD_O)$,MAC

Field 5, the outgoing Data Key encrypted under the MFK. This key is used to encrypt the data being translated. This field contains a 74 byte value or a volatile table location. The following headers are supported: 1DDNE000, 1DDNN000, 1DDEE000, and 1DDEN000.

`Encrypted Data`

Field 6, the data to be translated. This field contains a 16 byte hexadecimal value that can contain the numbers 0 to 9 and the characters A to F.

Table 6-6 Command 59: ECB-Mode Translation

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	59
1	Data continuation flag	1	0
2	Header,$E_{MFK.E}$(KMAC),MAC*	74	printable ASCII
3	MAC Data	1 - 240	0 - 9, A - Z, , . “ ”
4	Header,$E_{MFK.E}$(KD_I),MAC*	74	printable ASCII
5	Header,$E_{MFK.E}$(KD_O),MAC*	74	printable ASCII
6	Encrypted Data	16	0 - 9, A - F

* Can be a volatile table location.

Calling Parameters – CBC-Mode Translation

`59`

Field 0, the command identifier.

`0`

Field 1, the data continuation flag, must be set to 0.

Header,$E_{MFK.E}$(KMAC),MAC

Field 2, the KMAC Key encrypted under the MFK. This key is used to generate the MAC. This field contains a 74 byte value or a volatile table location. The following headers are supported: 1MDNE000, 1MDGE000.

`MAC Data`

Field 3, data to be authenticated. This field can be from one to 240 bytes long and can contain the characters A to Z, the numbers 0 to 9, and “,”, “.”, and “ ”.

Header,$E_{MFK.E}$(KD_I),MAC

Field 4, the incoming Data Key encrypted under the MFK. This key will be used to decrypt the data in field 6. This field contains a 74 byte value or a volatile table location. The following headers are supported: 1DDNE000, 1DDNE000, 1DDDE000, and 1DDDN000.

`Reserved`

Field 6, reserved field, it must be empty.

`[Header,`$E_{MFK.E}$`(`IV_I`),MAC]`

Field 7, the incoming Initialization Vector (IV-I) encrypted under the MFK. This Initialization Vector is used during decryption. If this field is empty, the default Initialization Vector of all zeros is used. This field contains a 74 byte value, or is empty. The following headers are supported: 1IDNE000 and 1IDNN000.

`[Header,`$E_{MFK.E}$`(`IV_O`),MAC]`

Field 8, the outgoing Initialization Vector (IVO) encrypted under the MFK. This Initialization Vector is used during re-encryption. If this field is empty, the default Initialization Vector of all zeros is used. This field contains a 74 byte value or is empty. The following headers are supported: 1IDNE000 and 1IDNN000.

`Data Type`

Field 9, the data type. This field contains a one byte value, either U or B. The following table indicates the type of data that the Network Security Processor supports and the value to enter in this field.

Value	Data Type
U	Unpacked ASCII hexadecimal
B	Binary

`Length`

Field 10, the length of the data in field 11.

The data is from eight to 3496 bytes long if the data is in binary format. If the data is in unpacked ASCII hexadecimal format, the length is from 16 to 3488 bytes long. This field contains a 1 to 4 byte decimal value.

`Data`

Field 11, the data to be translated, that is decrypted with the incoming data encryption key, and re-encrypted using the outgoing data encryption key.

This field is from eight to 3496 bytes long if the data is in binary format. If the data is in unpacked ASCII hexadecimal format, field is from 16 to 3488 bytes long.

When using this command, be sure to pad the data with zeros, if necessary, to achieve a value that is an eight byte multiple (binary) or a 16 byte multiple (ASCII).

Table 6-7 Command 59: CBC-Mode Translation

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	2	59
1	Data continuation flag	1	0
2	Header,$E_{MFK.E}$(KMAC),MAC*	74	printable ASCII
3	MAC Data	1 - 240	0 - 9, A - Z, , . " "
4	Header,$E_{MFK.E}$(KD_I),MAC*	74	printable ASCII
5	Header,$E_{MFK.E}$(KD_O),MAC*	74	printable ASCII
6	Reserved	0	
7	Header,$E_{MFK.E}$(IV_I),MAC	0, 74	printable ASCII
8	Header,$E_{MFK.E}$(IV_O),MAC	0, 74	printable ASCII
9	Data type	1	U, B
10	Length	1 - 4	0 - 9
11	Encrypted Data	8 - 3496 or 16 - 3488	0 - 9, A - F

* Can be a volatile table location.

Responding Parameters – ECB-Mode

`69`

Field 0, the response identifier.

`MAC`

Field 1, the 32 bit, generated MAC. This field contains an 8 byte hexadecimal value.

`KMAC Check Digits`

Field 2, the KMAC key check digits, the first four digits that result from encrypting zeros using the KMAC key. If option 88 is enabled, this field will contain six check digits.

`Encrypted Data`

Field 3, the encrypted or translated data. This field contains a 16 byte hexadecimal value.

`[Incoming KD Check Digits]`

Field 4, a variable field, depending on the nature of the command sent. If the command sent was translation, this field contains the incoming data key check digits, the first four digits that result from encrypting zeros using the incoming Data Key. If option 88 is enabled, this field will contain six check digits. If the command sent was encryption, this field is empty.

`Outgoing KD Check Digits`

Field 5, the outgoing data encryption key check digits, the first four digits that result from encrypting zeros using the outgoing Data Key. If option 88 is enabled, this field will contain six check digits.

Table 6-8 Response 69: ECB-Mode

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	69
1	MAC	8	0 - 9, A - F
2	KMAC Check Digits	4 or 6	0 - 9, A - F
3	Encrypted data	16	0 - 9, A - F
4	Incoming KD Check Digits	0, 4	0 - 9, A - F
5	Outgoing KD Check Digits	4 or 6	0 - 9, A - F

Responding Parameters – CBC-Mode

`69`

Field 0, the response identifier.

`MAC`

Field 1, the 32 bit generated MAC. This field contains an 8 byte hexadecimal value.

`KMAC Check Digits`

Field 2, the KMAC key check digits, the first four digits that result from encrypting zeros using the message authentication key. If option 88 is enabled, this field will contain six check digits.

`Reserved`

Field 3, a reserved field, it will be empty.

`[Incoming KD Check Digits]`

Field 4, a variable field, depending on the nature of the command sent. If the command sent was translation, this field contains the incoming data encryption key's check digits.

Check digits are the first four digits that result from encrypting zeros using the encryption key. If option 88 is enabled, this field will contain six check digits. If the command sent was encryption, this field is empty.

`Outgoing KD Check Digits`

Field 5, the outgoing Data Key check digits, the first four digits that result from encrypting zeros using the outgoing Data Key. If option 88 is enabled, this field will contain six check digits.

[Header,$E_{MFK.E}$(IV$_I$),MAC]

Field 6, a variable field, depending on the nature of the command sent.

If the command sent was translation, this field contains the ending Initialization Vector encrypted by the MFK. If the command sent was encryption, this field is empty.

Header,$E_{MFK.E}$(IV$_O$),MAC

Field 7, the ending Initialization Vector encrypted under the MFK. This IV results from encrypting the text or re-encrypting the text that is being translated. This field contains a 74 byte value.

`Data Type`

Field 8, this field contains a one byte value, either U for Unpacked ASCII hexadecimal, or B for binary.

`Data Length`

Field 9, the length of the data in field 10. For encryption, the length of the data returned in this field may be longer than the data sent. This field contains a to 4 byte decimal value.

`Encrypted Data`

Field 10, the encrypted or translated text. This field is from 8 to 3504 bytes long if the data is in binary format. If the data is in unpacked ASCII hexadecimal format, this field is from 16 to 3504 bytes long.

Table 6-9 Response 69: CBC-Mode

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	69
1	MAC	8	0 - 9, A - F
2	KMAC Check Digits	4 or 6	0 - 9, A - F
3	Reserved	0	
4	Incoming KD Check Digits	0, 4 or 6	0 - 9, A - F

Table 6-9 Response 69: CBC-Mode (continued)

Field	Contents	Length (bytes)	Legal Characters
5	Outgoing KD Check Digits	4 or 6	0 - 9, A - F
6	Header,$E_{MFK.E}(IV_I)$,MAC	0, 74	printable ASCII
7	Header,$E_{MFK.E}(IV_O)$,MAC	74	printable ASCII
8	Data type	1	U, B
9	Length	1 - 4	0 - 9
10	Encrypted data	8 - 3504 or 16 - 3504	0 - 9, A - F (if unpacked ASCII hexadecimal)

Usage Notes

- Generate the KMAC key cryptogram.
- Generate the incoming and outgoing data keys and IVs.
- This command can contain a binary data field. The Network Security Processor's and the host's serial interface must both be configured to 8 data bits, 1 stop bit, and no parity to correctly process binary data.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Generate a MAC and Encrypt Data Using ECB-Mode

- Clear-text KMAC key (KMAC): 8FF4 98F1 B661 5151
  The KMAC key (KMAC) in AKB format:
  1MDNE000,E210B2B1F67CE7F0F22434D315EA9361BE734B0079844FAF,381990A89
  D696C75
- Data to be authenticated:
  A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4Y5Z6A1B2C3
  D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X
- Clear-text outgoing Data Key: 3F78 1D6A B654 AEAD
  The outgoing Data Key in AKB format:
  1DDNE000,91F4020191D1B20AD50EC3976174DDAACDD4D5A212EACFAF,DF083221
  CA81678C
- Clear data to be encrypted: 1234567890ABCDEF

The command looks like this:

```
<59#0#1MDNE000,E210B2B1F67CE7F0F22434D315EA9361BE734B0079844FAF,381
990A89D696C75#A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4Y5Z6A
1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X##1DDNE000,91F4020191
D1B20AD50EC3976174DDAACDD4D5A212EACFAF,DF083221CA81678C#1234567890A
BCDEF#>
```

The Network Security Processor returns the following response:

```
<69#4316C2C1#1DE3#7CAA1966B0EFFA55##430D#>
```

Generate a MAC and Encrypt Data Using CBC-Mode using the default IV

- Clear-text KMAC key (KMAC): 8FF4 98F1 B661 5151
  The KMAC key (KMAC) in AKB format:
  1MDNE000,E210B2B1F67CE7F0F22434D315EA9361BE734B0079844FAF,381990A89
  D696C75
- Data to be authenticated:
  A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4Y5Z6A1B2C3
  D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X
- Clear-text outgoing Data Key: 3F78 1D6A B654 AEAD
  The outgoing Data Key in AKB format:
  1DDNE000,91F4020191D1B20AD50EC3976174DDAACDD4D5A212EACFAF,DF083221
  CA81678C
- Data type: Unpacked ASCII hexadecimal (U)
- Length: 16 bytes
- Clear data: 1234567890ABCDEF

The command looks like this:

```
<59#0#1MDNE000,E210B2B1F67CE7F0F22434D315EA9361BE734B0079844FAF,381
990A89D696C75#A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4Y5Z6A
1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X##1DDNE000,91F4020191
D1B20AD50EC3976174DDAACDD4D5A212EACFAF,DF083221CA81678C####U#16#123
4567890ABCDEF#>
```

The Network Security Processor returns the following response:

```
<69#4316C2C1#1DE3###430D##1IDNE000,34075897FEAAE260C64D0DEB41A1D9EF
03CAD29844B6C542,F3FBA0D9519D5484#U#16#7CAA1966B0EFFA55#>
```

Generate a MAC and Translate Data Using ECB-Mode

The following example illustrates translating data using ECB mode, based on the following input:

- Clear-text KMAC key (KMAC): 8FF4 98F1 B661 5151
  The KMAC key (KMAC) in AKB format:
  1MDNE000,E210B2B1F67CE7F0F22434D315EA9361BE734B0079844FAF,381990A89
  D696C75

- Data to be authenticated:
  1234567890ABCDEFGHIJ1234567890ABCDEFGHIJ1234567890ABCDEFGH
  IJ1234567890ABCDEFGHIJ1234567890ABCDEFGHIJ1234567890ABCDEFGH
  IJ1234567

- Clear-text incoming Data Key: D9E5 7FE9 8F83 322A
  The incoming Data Key in AKB format:
  1DDNE000,4631610968924203B476069817CD8AD935EA7A87F1F4A61E,5FC259758C
  F718FF

- Clear-text outgoing Data Key: 4029 BFE6 3720 0E98
  The outgoing Data Key in AKB format:
  1DDNE000,50F9EE4CC2FE3515816D13475A4B78EA906E2C2630911ABA,A5D35CC95
  7D63AD3

- Encrypted data to be translated: 413E C8B2 0165 E59A

The command looks like this:

```
<59#0#1MDNE000,E210B2B1F67CE7F0F22434D315EA9361BE734B0079844FAF,381
990A89D696C75#1234567890ABCDEFGHIJ1234567890ABCDEFGHIJ1234567890ABC
DEFGHIJ1234567890ABCDEFGHIJ1234567890ABCDEFGHIJ1234567890ABCDEFGHIJ
1234567#1DDNE000,4631610968924203B476069817CD8AD935EA7A87F1F4A61E,5
FC259758CF718FF#1DDNE000,50F9EE4CC2FE3515816D13475A4B78EA906E2C2630
911ABA,A5D35CC957D63AD3#413EC8B20165E59A#>
```

The Network Security Processor returns the following response:

```
<69#3B3F91F7#1DE3#2190B17248E002CA#90B2#B607#>
```

Verify and Generate MAC for VISA UKPT (Command 5C)

Command 5C verifies a MAC and generates an approval or denial MAC.

This command, by default, will generate a 1key-3DES (single-length) session key. Use option A2 to control the length of the generated session key.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

This command is enabled in the Network Security Processor’s default security policy.

Command

```
<5C#Header,EMFK.E(Derivation Key),MAC#Current Key Serial Number#
Data#MAC-1#[Session Key Length#]>
```

Response

```
<6C#Verification Flag#MACs#>[CRLF]
```

Calling Parameters

`5C`

Field 0, the command identifier.

`Header,E`$_{MFK.E}$`(Derivation Key),MAC`

Field 1, the Derivation Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1dDNE000 and 1dDNN000.

`Key Serial Number`

Field 2, the current Key Serial Number. This field contains a 10 to 20 byte hexadecimal value.

`Data`

Field 3, the data used to generate the MACs. This field contains a 16 or 32 byte hexadecimal value.

`MAC-1`

Field 4, the MAC to be verified. This field contains an eight byte hexadecimal value.

`[Session Key Length#]`

Field 5, this field is required only if option A2 is set to “B”. For all other cases, this field is optional. If this field exists, it should contain “S” if a 1key-3DES (single-length) session key is to be generated. It should contain “D” if a 2key-3DES (double-length) session key is to be generated. If option A2 is set to “D”, this field cannot contain the value “S”, and if option A2 is set to “S”, this field cannot contain the value “D”.

Table 6-10 Command 5C: Verify and Generate MAC for VISA UKPT

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	5C
1	Header,$E_{MFK.E}$(Derivation Key),MAC*	74	printable ASCII
2	Key Serial Number	10 - 20	0 - 9, A - F
3	Data	16, 32	0 - 9, A - F
4	MAC-1	8	0 - 9, A - F
5	[Session Key Length#]	0,1	S or D

* Can be a volatile table location.

Responding Parameters

`6C`

Field 0, the response identifier.

`Verification Flag`

Field 1, the MAC verification flag. This field returns Y if the MAC is verified or N if the MAC is not verified.

`MACs`

Field 2, contains the MACs to return to the PIN pad. This field returns MAC-2 and MAC-3 if MAC one is verified, otherwise, it returns 00000000 and MAC three. This field contains a 16 byte hexadecimal value.

Table 6-11 Response 6C: Verify and Generate MAC for VISA UKPT

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	6C
1	Verification flag	1	Y, N
2	MACs	16	0 - 9, A - F

Usage Notes

Before using Command 5C, generate the Base Derivation Key, and set option A2 as necessary.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

1key-3DES (single-length) session key is used to verify MAC-1 and generate an approval or denial MAC in return

- Option A2 is set to “S” in the Network Security Processor security policy, and option 6C is enabled.
- Clear-text Base Derivation Key: 1334 1334 1334 1334
  The Base Derivation Key in AKB format:
  1dDNE000,449DD7F6420AD2AD710102B444C180F37D0E78F5EFC186E7,181753679
  9D56B5E
- Key Serial Number: 9876 5432 10E0 0001
- Data: 1234 1234 5678 5678
- MAC #1: 6268 14A7

The command looks like this:

```
<5C#1dDNE000,449DD7F6420AD2AD710102B444C180F37D0E78F5EFC186E7,18175
36799D56B5E#9876543210E00001#1234123456785678#626814A7#>
```

The Network Security Processor returns the following response:

```
<6C#Y#0C266C371DEABF85#>
```

where MAC #2 = `0C266C37`, and MAC #3 = `1DEABF85`

This example shows the syntax when option A2 is set to “B” in the Network Security Processor security policy, and option 6C is enabled.

```
<5C#1dDNE000,449DD7F6420AD2AD710102B444C180F37D0E78F5EFC186E7,18175
36799D56B5E#9876543210E00001#1234123456785678#626814A7#S#>
```

2key-3DES (double-length) session key is used to verify MAC-1 and generate an approval or denial MAC in return

- Option A2 is set to “D” in the Network Security Processor security policy.
- Clear-text Base Derivation Key: 0123 4567 89AB CDEF FEDC BA98 7654 3210
  The Base Derivation Key in AKB format:
  1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,342946FE
  884AA8B2
- Key Serial Number: 9876 5432 10E0 0001

- Data: 1234 1234 5678 5678
- MAC #1: 7E37 D982

The command looks like this:

```
<5C#1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,34294
6FE884AA8B2#9876543210E00001#1234123456785678#7E37D982#>
```

The Network Security Processor returns the following response:

```
<6C#Y#4D3AA91B0A0E7E12#>
```

where MAC #2 = `4D3AA91B`, and MAC #3 = `0A0E7E12`

This example shows the syntax when option A2 is set to “B” in the Network Security Processor security policy.

```
<5C#1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,34294
6FE884AA8B2#9876543210E00001#1234123456785678#7E37D982#D#>
```

The Network Security Processor returns the following response:

```
<6C#Y#4D3AA91B0A0E7E12#>
```

where MAC #2 = `4D3AA91B`, and MAC #3 = `0A0E7E12`

Verify MAC and Decrypt PIN (Command 5F)

Command 5F verifies a MAC, and if successful, decrypts an encrypted PIN Block.

This command is not enabled in the Network Security Processor’s default security policy. You must purchase this command in the form of a command 105, and enable it in the Network Security Processor’s security policy.

Command

```
<5F#Header,EMFK.E(KC),MAC#MAC Length#Header,EMFK.E(IV),MAC#MAC#
EKC(PIN Block)#Data Type#Data Length#Data#>
```

Response

```
<6F#Verification Flag or Header,EMFK.E(Ending IV),MAC#
Decrypted PIN Block#KC Check Digits#>
```

Calling Parameters

`5F`

Field 0, the command identifier.

`Header,E`$_{MFK.E}$`(KC),MAC`

Field 1, the Data Key used for two purposes: first to verify the MAC, and if successful, to decrypt the encrypted PIN block. This field contains a 74 byte value. The following headers are supported: 1cDNE00B and 1cDNN00B.

`MAC Length`

Field 2, the size of the MAC to be verified. If field 2 is set to 0, fields 4 and 5 must be empty.

Value	MAC Size
0	More data expected, no MAC verified
1	32 bits
2	48 bits
3	64 bits

A 32 bit MAC is expressed as eight hexadecimal digits (0-9, A - F) and displayed as two groups of four digits, separated by a space. A 48 bit or 64 bit MAC is expressed as three or four groups of four hexadecimal digits, separated by a space.

`[Header,E`$_{MFK.E}$`(IV),MAC]`

Field 3, the Initialization Vector encrypted under the MFK. This Initialization Vector is used in the verification of a MAC.

If this command contains the first block of multiple blocks of data, this field must be empty. If this command contains data subsequent to the first block in a multi-block series (that is, it contains continuation data), this field contains the ending Initialization Vector from the previously sent data block. This field contains a 74 byte value, or is empty. The following headers are supported: 1IDNE000 and 1IDNN000.

`MAC`

Field 4, the MAC to be verified. This field will contain the MAC when there is no more data in a subsequent command. A 32 bit MAC is expressed as eight hexadecimal digits (0-9, A - F) and displayed as two groups of four digits, separated by a space. A 48 bit or 64 bit MAC is expressed as three or four groups of four hexadecimal digits, separated by a space. This field will be empty if field 2 contains a 0.

E_{KC}`(PIN Block)`

Field 5, the incoming PIN Block encrypted under the communications key (KC). This field contains a 16 byte hexadecimal value or is empty.

`Data Type`

Field 6, the data type. The following table indicates the type of data that the Network Security Processor supports.

Value	**Data Type**
U	Unpacked ASCII hexadecimal
B	Binary

`Data Length`

Field 7, the data length.

This command will authenticate up to 4096 bytes of data.

If more data is being sent in the next command – indicated by field 2 being set to 0 – the data length must be multiple of eight. See Authentication in Batches for additional information.

If data sent is not batched, the Network Security Processor will right pad the data with zeros, such that, its length will be a multiple of eight.

This field contains a 1 to 4 byte decimal value.

`Data`

Field 8, the data to be authenticated. This field can be from 1 to 4096 bytes long. If the data is in unpacked ASCII hexadecimal format, this field can contain the numbers 0 to 9 and the characters A to F.

Table 6-12 Command 5F: Verify MAC and Decrypt PIN

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	5F
1	Header,$E_{MFK.E}$(KC),MAC*	74	printable ASCII
2	MAC Length	1	0 - 3
3	[Header,$E_{MFK.E}$(IV),MAC]	0, 74	printable ASCII
4	MAC	0, 9, 14, 19	0 - 9, A - F, " "
5	E_{KC}(PIN Block)	0, 16	0 - 9, A - F
6	Data Type	1	U, B
7	Data Length	1 - 4	0 - 9
8	Data to be authenticated	1 - 4096	0 - 9, A - F if unpacked ASCII

* Can be a volatile table location.

Responding Parameters

`6F`

Field 0, the response identifier.

`Verification Flag or Header,`$E_{MFK.E}$`(Ending IV),MAC`

Field 1, the ending Initialization Vector if command field 2 is set to 0, or the MAC verification flag if command field 2 is not set to 0.

If your use of this command results in the generation of an ending Initialization Vector in this field, use it as the starting Initialization Vector in subsequent MAC command to continue generating MACs.

If your use of this command results in a MAC verification flag, this field will return Y if the MAC is verified, or N if the MAC is not verified.

`Decrypted PIN Block`

Field 2, the decrypted PIN block. Field 2 is empty if MAC is not verified or command field 2 is set to 0.

`KC Check Digits`

Field 3, check digits, the first four digits that result from encrypting zeros using the Communications Key. If option 88 is enabled, this field will contain six check digits.

Table 6-13 Response 6F: Verify MAC and Decrypt PIN

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	6F
1	Verification Flag or Header,$E_{MFK.E}$(Ending IV),MAC	1, 74	Y, N, 0 - 9, A - F, “,”
2	Decrypted PIN Block	0, 16	0 - 9, A - F
3	KD Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- Generate the communications key.
- This command can contain a binary data field.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify a MAC and decrypt the outer layer of a PIN Block

- Clear-text Communications Key: FEDC BA98 7654 3210
  The Communications Key in AKB format:
  1cDNE00B,DA52B7C6DE41613913B5D5A0F0E530443D738DEE08938C25,C9128885B944B7E2
- MAC length: 48 bits (2)
- Clear-text Initialization Vector: 2558 8552 2558 8552
  The Initialization Vector in AKB format:
  1IDNE000,CE6518F12DABFE5BFBD04D67D95ACABFD7657504C0173D0D,0153727BB483E4FB
- MAC (48 bits): 78FA FA86 68CF
- Clear-text PIN block: 1234 0000 0000 0000
  The PIN block encrypted under the Communications Key: A931 0B88 55BC 6881
- Data type: Binary (B)
- Data length: 3 bytes

- Data to be authenticated: 040

The command:

```
<5F#1cDNE00B,DA52B7C6DE41613913B5D5A0F0E530443D738DEE08938C25,C9128
885B944B7E2#2#1IDNE000,CE6518F12DABFE5BFBD04D67D95ACABFD7657504C017
3D0D,0153727BB483E4FB#
78FA FA86 68CF#A9310B8855BC6881#B#3#040#>
```

The Network Security Processor returns the following response:

```
<6F#Y#1234000000000000#A68C#>
```

Generate MAC (Command 98)

Command 98 generates a MAC using Cipher Block Chaining.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

This command has a high security exposure, it is not enabled in the Network Security Processor's default security policy. To use this command, enable it in the Network Security Processor's security policy.

The following types of MACs can be generated.

- ISO 9797-1 Algorithm 1 - uses all three blocks of the KMAC key for all blocks of data.
- ISO 9797-1 Algorithm 3 - uses DES-CBC with the first block of the KMAC key for all data blocks except the last block (8 bytes). The last data block is processed using 3DES-CBC with all three key blocks of the KMAC key.

Command

```
<98#Header,EMFK.E(KMAC),MAC#MAC Length#MAC Type#
[Header,EMFK.E(IV),MAC]#Data Type#Length#Data#[Reserved#]>
```

Response

```
<A8#MAC Length#MAC or Header,EMFK.E(Ending IV),MAC#
KMAC Check Digits#>[CRLF]
```

Calling Parameters

`98`

Field 0, the command identifier.

Header,$E_{MFK.E}$(KMAC),MAC

Field 1, the KMAC Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1MDNE000, 1MDNN000, 1MDGE000, and 1MDGN000.

`MAC Length`

Field 2, the length of the MAC to be returned. The following table indicates the possible returned MAC lengths.

Value	Returned MAC Size
0	More data expected, no MAC returned
1	32 bits
2	48 bits
3	64 bits

A 32 bit MAC is expressed as eight hexadecimal digits (0-9, A - F) and displayed as two groups of four digits, separated by a space. A 48- or 64-bit MAC is expressed as three or four groups of four hexadecimal digits, separated by a space.

`MAC Type`

Field 3, the algorithm used to generate the MAC. The following table indicates the supported MAC types.

Value	MAC Type
Empty, or 1-6	ISO - 9797-1 Algorithm 1 - 1key or 2key-3DES Cipher block chaining.
7	ISO 9797-1 Algorithm 3 - Only the last data block is processed using 3DES, all previous blocks are processed using single DES.

[Header,$E_{MFK.E}$(IV),MAC]

Field 4, the Initialization Vector encrypted under the MFK.

If this command contains the first block of multiple blocks of data, or if you are authenticating only one block of data, this field must be empty. The Network Security Processor will use an Initialization Vector of all zeros.

If this command contains data subsequent to the first block in a multi-block series (that is, it contains continuation data), this field should contain the ending Initialization Vector from the previously sent data block. This field is either empty, or contains a 74 byte value. The following headers are supported: 1IDNE000 and 1IDNN000.

`Data Type`

Field 5, the data type. The following table indicates the type of data that the Network Security Processor supports.

Value	Data Type
U	Unpacked ASCII hexadecimal
B	Binary

See Data formats for more information.

`Length`

Field 6, the data's length. This command will authenticate up to 4096 bytes of data. If more data is being sent in the next command – indicated by field 2 being set to 0 – the data length must be a multiple of eight for binary data, or a multiple of 16 for Unpacked ASCII data. If no more data is being sent, the Network Security Processor will right-pad the data with binary zeros (nulls, 0x00) such that the resulting data length will be a multiple of eight. This field contains a 1 to 4 byte decimal value.

`Data`

Field 7, the input data. This field can be from one to 4096 bytes long. If the data is in unpacked ASCII hexadecimal format, this field can contain the numbers 0 to 9 and the characters A to F.

`[Reserved#]`

Field 8, this field is reserved, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

Table 6-14 Command 98: Generate MAC

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	2	98
1	Header,$E_{MFK.E}$(KMAC),MAC*	74	printable ASCII
2	Response length	1	0 - 3
3	MAC Type	0, 1	Empty, or 1-7
4	Header,$E_{MFK.E}$(IV),MAC	0, 74	printable ASCII
5	Data type	1	U, B
6	Length	1 - 4	0 - 9
7	Data	1 - 4096	0 -9, A - F (if unpacked ASCII)
8	[Reserved]	0, 1, 2	0 - 9

* Can be a volatile table location.

Responding Parameters

`A8`

Field 0, the response identifier.

`MAC Length`

Field 1, the length of the MAC. This field will contain the value specified in field 2 of the command.

`MAC or Header,`$E_{MFK.E}$`(Ending IV),MAC`

Field 2, if field 1 is set to zero this field will contain the ending Initialization Vector encrypted under the MFK. If field 1 is not set to 0, this field will contain the MAC.

If your use of this command results in the generation of an ending Initialization Vector in this field, use it as the starting Initialization Vector in the subsequent MAC command to continue generating MACs. This field contains a 9, 14, 16, 19, or 74 byte value as well as spaces (that is, “ ”).

`KMAC Check Digits`

Field 3, check digits, the first four digits that result from encrypting zeros using the KMAC key. If option 88 is enabled, this field will contain six check digits.

`[MAC Type#]`

Field 4, this field is only present if the MAC type is either 6 or 7.

Table 6-15 Response A8: Generate MAC

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	A8
1	MAC length	1	0 - 3
2	MAC or Header,$E_{MFK.E}$(Ending IV),MAC	9, 14, 16, 19, 74	0 - 9, A - F, “ ”, “,” or printable ASCII
3	KMAC Check Digits	4 or 6	0 - 9, A - F
4*	MAC Type	1	6 or 7

* Only present if MAC Type is 6 or 7

Usage Notes

- Generate the KMAC key.
- This command can contain a binary data field.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Generate a 3DES CBC MAC using the Default IV

- Clear-text KMAC key: FEDC BA98 7654 3210 0123 4567 89AB CDEF
  The KMAC key in AKB format:
  1MDNE000,EEC942756FEE2710B52EBE9EA6C349192369EFDC1D1D0ACE,7BB0728F8BAD0432
- MAC length: 32 bits (1)
- Data type: Unpacked ASCII hexadecimal (U)
- Data length: 6 bytes
- Data: 303430

The command looks like this:

```
<98#1MDNE000,EEC942756FEE2710B52EBE9EA6C349192369EFDC1D1D0ACE,7BB07
28F8BAD0432#1#6##U#6#303430#>
```

The Network Security Processor returns the following response:

```
<A8#1#AFA3 9CEF#7B83#6#>
```

Generate an ISO 9797-1 Algorithm 3 - MAC

- Clear-text KMAC key: FEDC BA98 7654 3210 0123 4567 89AB CDEF
  The KMAC key in AKB format:
  1MDNE000,EEC942756FEE2710B52EBE9EA6C349192369EFDC1D1D0ACE,7BB0728F8BAD0432
- MAC length: 32 bits (1)
- Data type: Unpacked ASCII hexadecimal (U)
- Data length: 18 bytes
- Data: 303430303430303430

The command looks like this:

```
<98#1MDNE000,EEC942756FEE2710B52EBE9EA6C349192369EFDC1D1D0ACE,7BB07
28F8BAD0432#1#7##U#18#303430303430303430#>
```

The Network Security Processor returns the following response:

```
<A8#1#B21A E4A4#7B83#7#>
```

Verify MAC (Command 99)

Command 99 verifies a MAC using Cipher Block Chaining. It is enabled in the Network Security Processor's default security policy.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

The following types of MACs can be verified.

- ISO 9797-1 Algorithm 1 - - uses all three blocks of the KMAC key for all blocks of data.
- ISO 9797-1 Algorithm 3 - - uses DES-CBC with the first block of the KMAC key for all data blocks except the last block (8 bytes). The last data block is processed using 3DES-CBC with all three key blocks of the KMAC key.

Command

```
<99#Header,EMFK.E(KMAC),MAC#MAC Type#[Header,EMFK.E(IV),MAC]#
MAC Length#Data Type#Data Length#Data#[MAC]#[Reserved#]>
```

Response

```
<A9#MAC Length#Verification Flag or Header,EMFK.E(Ending IV),MAC#
KMAC Check Digits#>[CRLF]
```

Calling Parameters

`99`

Field 0, the command identifier.

`Header,EMFK.E(KMAC),MAC`

Field 1, the KMAC key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1MDNE000, 1MDNN000, 1MDVE000, and 1MDVN000.

`[MAC Type]`

Field 2, the algorithm used to verify the MAC. The following table indicates the supported MAC types.

Value	MAC Type
Empty, or 1-6	ISO - 9797-1 Algorithm 1 - 1key or 2key-3DES Cipher block chaining.
7	ISO 9797-1 Algorithm 3 - Only the last data block is processed using 3DES, all previous blocks are processed using single DES.

`[Header,`$E_{MFK.E}$`(IV),MAC]`

Field 3, the Initialization Vector encrypted under the MFK.

If this command contains the first block of multiple blocks of data, or if you are authenticating only one block of data, this field must be empty. The Network Security Processor will use an Initialization Vector of all zeros.

If this command contains data subsequent to the first block in a multi-block series (that is, it contains continuation data), this field should contain the ending Initialization Vector from the previously sent data block. This field is either empty, or contains a 74 byte value. The following headers are supported: 1IDNE000 and 1IDNN000.

`MAC Length`

Field 4, the size of the MAC to be verified. The following table indicates the possible MAC sizes.

Value	MAC Size
0	More data expected, no MAC verified
1	32 bits
2	48 bits
3	64 bits

A 32 bit MAC is expressed as eight hexadecimal digits (0-9, A - F) and displayed as two groups of four digits, separated by a space. A 48- or 64-bit MAC is expressed as three or four groups of four hexadecimal digits, separated by a space.

`Data Type`

Field 5, the data type. The following table indicates the type of data that the Network Security Processor supports.

Value	Data Type
U	Unpacked ASCII hexadecimal
B	Binary

`Data Length`

Field 6, the data's length. This command will authenticate up to 4096 bytes of data. If more data is being sent in the next command – indicated by field 4 being set to zero – then the data length must be a multiple of eight for binary data, and a multiple of 16 for Unpacked ASCII data. If no more data is being sent, the Network Security Processor will right-pad the data with binary zeros (nulls, 0x00) such that the resulting data length will be a multiple of eight. This field contains a 1 to 4 byte decimal value.

`Data`

Field 7, the data to be authenticated. If the data is in unpacked ASCII hexadecimal format, this field can contain the numbers 0 to 9 and the characters A to F.

`[MAC]`

Field 8, the MAC to be verified when no more data is expected. A 32 bit MAC is expressed as eight hexadecimal digits. It is expressed as two groups of four digits, separated by a space. A 48- or 64-bit MAC is expressed as three or four groups of four hexadecimal digits, separated by a space.

This field must be empty if more data will be sent in a subsequent command.

`[Reserved#]`

Field 9, this field is reserved, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

Table 6-16 Command 99: Verify MAC

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	2	99
1	Header,$E_{MFK.E}$(KMAC),MAC*	74	printable ASCII
2	[MAC Type]	0, 1	1 - 7
3	[Header,$E_{MFK.E}$(IV),MAC]	0, 74	printable ASCII
4	MAC length	1	0 - 3
5	Data type	1	U, B
6	Data length	1 - 4	0 - 9
7	Data to be authenticated	1 - 4096	0 - 9, A - F (if unpacked ASCII)
8	[MAC]	0, 9, 14, 19	0 - 9, A - F
9	[Reserved]	0, 1, 2	0 - 9

* Can be a volatile table location.

Responding Parameters

`A9`

Field 0, the response identifier.

`MAC Length`

Field 1, the length of the MAC.

`Verification Flag or Header,`$E_{MFK.E}$`(Ending IV),MAC`

Field 2, if field 1 is set to 0, this field will contain the ending Initialization Vector. If field 1 is not set to 0, this field will contain the MAC verification flag. This field contains a 74 byte value, or “Y” or “N”.

If your use of this command results in the generation of an ending Initialization Vector in this field, use it as the starting Initialization Vector in subsequent MAC command to continue generating MACs.

If your use of this command results in a MAC verification flag, this field will return Y if the MAC is verified, or N if the MAC is not verified.

`KMAC Check Digits`

Field 3, check digits, the first four digits that result from encrypting zeros using the KMAC key. If option 88 is enabled, this field will contain six check digits.

`[MAC Type#]`

Field 4, the MAC Type. This field exists only if the MAC Type is 6 or 7.

Table 6-17 Response A9: Verify MAC

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	A9
1	MAC length	1	0 - 3
2	Verification flag or Header,$E_{MFK.E}$(Ending IV),MAC	1, 74	0 - 9, A - F, Y, N, “,” or printable ASCII
3	KMAC Check Digits	4 or 6	0 - 9, A - F
4*	[MAC Type#]	1	6, 7

* This field exists only when the MAC Type is 6 or 7.

Usage Notes

- Generate the KMAC key.
- This command can contain a binary data field.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify a 3DES CBC MAC using the Default IV

- Clear-text KMAC key: FEDC BA98 7654 3210 0123 4567 89AB CDEF
  The KMAC key in AKB format:
  1MDNE000,EEC942756FEE2710B52EBE9EA6C349192369EFDC1D1D0ACE,7BB0728F
  8BAD0432
- MAC Type : 6
- MAC length: 1
- Data type: Unpacked (U)
- Data length: 6
- Data: 303430
- MAC: AFA3 9CEF

The command looks like this:

```
<99#1MDNE000,EEC942756FEE2710B52EBE9EA6C349192369EFDC1D1D0ACE,7BB0728
F8BAD0432#6##1#U#6#303430#AFA3 9CEF#>
```

The Network Security Processor returns the following response:

```
<A9#1#Y#7B83#6#>
```

Verify an ISO 9797-1 Algorithm 3 - MAC using the Default IV

- Clear-text KMAC key: FEDC BA98 7654 3210 0123 4567 89AB CDEF
  The KMAC key in AKB format:
  1MDNE000,EEC942756FEE2710B52EBE9EA6C349192369EFDC1D1D0ACE,7BB0728F
  8BAD0432
- MAC Type : 7
- MAC length: 1
- Data type: Unpacked (U)
- Data length: 18
- Data: 3034 3030 3430 3034 30
- MAC: B21A E4A4

The command looks like this:

```
<99#1MDNE000,EEC942756FEE2710B52EBE9EA6C349192369EFDC1D1D0ACE,7BB07
```

```
28F8BAD0432#7##1#U#18#303430303430303430#B21A E4A4#>
```

The Network Security Processor returns the following response:

```
<A9#1#Y#7B83#7#>
```

Verify ACR Response MAC (Command 9B)

Command 9B verifies the response of the challenge number in both normal and auto mode for the Atalla Challenge Response (ACR) token. This command is enabled in the Network Security Processor's default security policy.

This command can accept a 1key-3DES (single-length) key only if 6C is enabled.

Command

```
<9B#Header,EMFK.E(KMACR),MAC#Challenge Number#MAC#Response Format#
[Mode#]>
```

Response

```
<AB#Verification Flag#Residual MAC#>[CRLF]
```

Calling Parameters

`9B`

Field 0, the command identifier.

`Header,EMFK.E(KMACR),MAC`

Field 1, the KMACR Key encrypted under the MFK. This field contains a 74 byte value or a volatile table location. The following headers are supported: 1MDNE00C, 1MDNN00C, 1MDVE00C and 1MDVN00C.

`Challenge Number`

Field 2, the data, typically the challenge number that was used to compute the MAC.

This field can be four to 128 bytes long and can contain the numbers 0 to 9.

In auto mode, this field will be contain 8 characters for initialization, or 9 characters if it contains the previous response with a single challenge number.

`MAC`

Field 3, the MAC response to be verified. This field contains a 4 to 8 byte hexadecimal value.

Response Format

Field 4, the response format: H, D, or D*nnnnnnnnnnnnnnnn*:

Entering H in this field, indicates that the response will be a hexadecimal value.

Entering D in this field, indicates that the default decimalization table will be used to construct the response. The default decimalization table is: 0123456789222333.

To change the contents of the decimalization table, enter D*nnnnnnnnnnnnnnnn* where *nnnnnnnnnnnnnnnn* contains the numbers you want in the table. The response will be a decimal value.

```
[Mode]
```

Field 5, the mode flag. This field is optional.

Normal mode value is either 0, or empty.

Auto mode value is 1.

Table 6-18 Command 9B: Verify Response MAC

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	2	9B
1	Header,$E_{MFK.E}$(KMAC$_R$),MAC*	74	printable ASCII
2	Challenge number Normal Mode Auto Mode	 4 - 128 8, 9	 0 - 9 0 - 9, A - F
3	MAC	4 - 8	0 - 9, A - F
4	Response format	1, 17	H, D, D*nnnnnnnnnnnnnnnn*
5	[Mode]	0,1	0, 1

* Can be a volatile table location.

Responding Parameters

```
AB
```

Field 0, the response identifier.

```
Verification Flag
```

Field 1, the verification flag. This field returns Y if the MAC is verified, otherwise, it returns N.

```
Residual MAC
```

Field 2, a residue MAC, the last 32 bits of the MAC result. This field contains an eight byte hexadecimal value.

If the MAC is not verified, this field returns XXXXXXXX.

A 16 byte number if the MAC is verified in auto mode, the first 32 bits plus Residue MAC (last 32 bits). The MAC must be saved in Host DB for the next verification routine (Auto Mode keeps track of previous MAC results in continuous calculation). This response is not decimalized.

Table 6-19 Response AB: Verify Response MAC

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	AB
1	Verification flag	1	Y, N
2	Residual MAC	8, 16	0 - 9, A - F, X

Usage Notes

Before using Command 9B, generate the KMAC key.

ACR Token Auto Mode (types 2, 3, 6 and 7)

The ACR Token has six system fields.

The field lengths are 8 characters.

These fields are utilized in auto mode.

Up to six users can be assigned an ACR token.

ACR Token Auto Mode steps:

User enters the PIN followed by predefined single digit user system number (0-5). Then the user enters a predefined or single digit challenge number selected by the host, which is used to generate the Response.

The Response is generated from MAC processed data, 8 characters from system field 1, the single digit challenge number and 7 zeros.

The left 8 digits of this result are used as a Response or stored in the selected system field for the next operation.

Overview of Initialization of System Fields

When the Host Application needs to initialize or re-synchronize the auto mode operation, the host generates 8 digits for the challenge number and requires the user to enter the 8 digit challenge number instead of single digit.

The result of this 8 digit challenge (the response) is in turn stored in the system field.

The host saves the left 8 digits of result (from field 2 of response “AB”) in the data base, and then this value is used as part of the challenge for the next operation.

ACR Sample Flow

Normal Mode.

This is a sample flow for normal mode operation for the ACR:

1. The user enters their PIN into the ACR.
2. The system prompts with a challenge number, typically 4 to 8 digits (created from Command 93).
3. The user enters the challenge number into the ACR.
4. The ACR responds with a Response MAC.
5. The user enters this Response into the system.
6. The system verifies that this Response MAC is correct and allows the user to continue logging on (Command 9B).

To calculate the Response, generate a MAC on the ASCII representation of the challenge number.

Auto (Single Digit) Mode

This is a sample flow for INITIALIZING or RESYNCHRONIZATION of the ACR in single digit mode:

1. The user enters their PIN into the ACR (and optionally a system field number, the default system field number is 0)
2. The system prompts with an 8-digit challenge number (from Command 93). The user enters the challenge number into the ACR.
3. The ACR responds with a Response MAC. This response MAC is also stored in the selected system field for future use.
4. The user enters this Response into the system.
5. The system verifies that this Response MAC is correct and allows the user to continue logging on.

To calculate the Response, generate a MAC on the ASCII representation of the challenge number.

This is a sample flow for standard operation of the ACR in single digit mode:

1. The user enters their PIN into the ACR (and optionally a system field number, the default system field number is 0)
2. The user enters a predefined single digit challenge number into the ACR.

3. The ACR responds with a Response MAC which is calculated from the stored system field value and the entered single digit challenge. This response MAC is also stored in the selected system field for future use.

4. The user enters this Response into the system.

5. The system verifies that this Response MAC is correct and allows the user to continue logging on.

To calculate the Response, generate a MAC on the packed representation of the first 8 characters of the saved response (the response from the previous challenge/response session). Concatenate this with the single digit challenge and zero filled to 64 bits.

Examples

These detailed types show an example of Verifying with Hexadecimal MAC or Decimal MAC using either the Default Table or Custom Table.

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify a Hexadecimal MAC

This example illustrates verifying a hexadecimal MAC, based on the following input:

- Clear-text KMAC key: 69EA 0A4E 73CF F9F0
  The KMAC key in AKB format:
  1MDNE00C,0E75B459229A68B735D1C1BB222038DA98E89DE7EF11DBC2,9BDFAFA0DA31C91E

- Challenge number: 1487

- MAC response to be verified: 2B4F AB1A

- Response format: Hexadecimal (H)

The command looks like this:

```
<9B#1MDNE00C,0E75B459229A68B735D1C1BB222038DA98E89DE7EF11DBC2,9BDFA
FA0DA31C91E#1487#2B4FAB1A#H#>
```

The Network Security Processor returns the following response:

```
<AB#Y#3EAE165F#>
```

Verify a Decimal MAC Using a Customized Table

- Clear-text KMAC key: 69EA 0A4E 73CF F9F0
  The KMAC key in AKB format:
  1MDNE00C,0E75B459229A68B735D1C1BB222038DA98E89DE7EF11DBC2,9BDFAFA0DA31C91E
- Challenge number: 1618 5
- MAC response to be verified: 1111 1111
- Response format: Decimal (D) using the decimalization table: 1111 1111 1111 1111

The command looks like this:

```
<9B#1MDNE00C,0E75B459229A68B735D1C1BB222038DA98E89DE7EF11DBC2,9BDFA
FA0DA31C91E#16185#11111111#D1111111111111111#>
```

The Network Security Processor returns the following response:

```
<AB#Y#4DC6D4DE#>
```

Verify CMAC using TDES (Command 304)

Command 304 verifies a Cipher-based Message Authentication Code (CMAC) using the triple Data Encryption Algorithm (TDEA) in accordance with *NIST Special Publication 800-38B*.

note To be consistent with the remainder of this manual, the acronym 3DES is used in place of TDEA.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<304#Header,EMFK.E(KCMAC),MAC#Algorithm#Data Type#Data Length#Data#
CMAC#>
```

Response

```
<404#Verification Flag#KCMAC Check Digits#>
```

Calling Parameters

`304`

Field 0, the command identifier.

`Header,`$E_{MFK.E}$`(KCMAC),MAC`

Field 1, the CMAC-TDES key encrypted under the MFK. This key can be a 2key-3DES (double-length) key. If the MFK is a 3key-3DES (triple-length) key, this key can be either a double-length key or a triple-length key. The following headers are supported: 1BDNE000, 1BDNN000, 1BDGE000, and 1BDGN000.

`Algorithm`

Field 2, the algorithm used to generate the CMAC. This field must contain the number 1 which indicates 3DES.

`Data Type`

Field 3, the data type. This field must contain the letter "U" for Unpacked ASCII hexadecimal data.

`Data Length`

Field 4, the length of the data in field 5. This field must contain an even number in the range of 2 through 4096.

`Data`

Field 5, the data to be processed. This field must contain an even number of hexadecimal characters. The length of this field must be equal to the value specified in field 4.

`CMAC`

Field 5, the CMAC to be verified. If this field contains less than 16 hexadecimal characters, it must contain the leftmost hexadecimal characters of the CMAC result. The minimum length of this field is 4 hexadecimal characters, the maximum length is 16 hexadecimal characters. This field must contain an even number of hexadecimal characters.

Table 6-20 Command 304: Verify CMAC using TDES

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	304
1	Header,$E_{MFK.E}$(KMAC),MAC	74	printable ASCII
2	Algorithm	1	1
3	Data Type	1	U
4	Data Length	1 - 4	0 - 9
5	Data	2 - 4096	0 - 9, A - F
6	CMAC	4 - 16	0 - 9, A - F

Responding Parameters

`404`

Field 0, the response identifier.

`Verification Flag`

Field 1, this field will contain the letter “Y” if the CMAC verified. It will return the letter “N” if the CMAC did not verify.

`KCMAC Check Digits`

Field 2, the first four digits that result from encrypting zeros using the KCMAC key. If option 88 is enabled, this field will contain six check digits.

Table 6-21 Response 404: Verify CMAC using TDES

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	404

Table 6-21 Response 404: Verify CMAC using TDES (continued)

Field	Contents	Length (bytes)	Legal Characters
1	Verification Flag	1	Y or N
2	KCMAC Check Digits	4, 6	0 - 9, A - F

Usage Notes

- Generate the AKB for the KCMAC key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text triple-length KCMAC Key:
  8AA83BF8CBDA10620BC1BF19FBB6CD58BC313D4A371CA8B5, check digits C8CC.
- The KCMAC key in AKB format:
  1BDGE000,BAEBD923AA1215A9B4B1D6BE65EF332880FF4B4A937DE973,E72B8D32DEBD8D1E
- Algorithm: 1
- Data Type: U
- Data Length: 64
- Data: 6BC1BEE22E409F96E93D7E117393172AAE2D8A571E03AC9C9EB76FAC45AF8E51

The command looks like this:

```
<304#1BDGE000,BAEBD923AA1215A9B4B1D6BE65EF332880FF4B4A937DE973,E72B
8D32DEBD8D1E#1#U#64#6BC1BEE22E409F96E93D7E117393172AAE2D8A571E03AC9
C9EB76FAC45AF8E51#33E6B1092400EAE5#>
```

The Network Security Processor returns the following response:

```
<404#Y#C8CC#>
```

Generate CMAC using TDES (Command 305)

Command 305 generates a Cipher-based Message Authentication Code (CMAC) using the triple Data Encryption Algorithm (TDEA) in accordance with *NIST Special Publication 800-38B*.

note To be consistent with the remainder of this manual, the acronym 3DES is used in place of TDEA.

This command is not enabled in the Network Security Processor's default security policy. To use this command, you must add it to the Network Security Processor's security policy.

Command

<305#Header,$E_{MFK.E}$(KCMAC),MAC#Algorithm#Data Type#Data Length#Data#>

Response

```
<405#CMAC#KCMAC Check Digits#>
```

Calling Parameters

`305`

Field 0, the command identifier.

`Header,`$E_{MFK.E}$`(KCMAC),MAC`

Field 1, the CMAC-TDES key encrypted under the MFK. This key can be a 2key-3DES (double-length) key. If the MFK is a 3key-3DES (triple-length) key, this key can be either a double-length key or a triple-length key. The following headers are supported: 1BDNE000, 1BDNN000, 1BDGE000, and 1BDGN000.

`Algorithm`

Field 2, the algorithm used to generate the CMAC. This field must contain the number 1 which indicates 3DES.

`Data Type`

Field 3, the data type. This field must contain the letter "U" for Unpacked ASCII hexadecimal data.

`Data Length`

Field 4, the length of the data in field 5. This field must contain an even number in the range of 2 through 4096.

`Data`

Field 5, the data to be processed. This field must contain an even number of hexadecimal characters. The length of this field must be equal to the value specified in field 4.

Table 6-22 Command 305: Generate CMAC using TDES

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	305
1	Header,$E_{MFK.E}$(KMAC),MAC	74	printable ASCII
2	Algorithm	1	1
3	Data Type	1	U
4	Data Length	1-4	0 - 9
5	Data	2 - 4096	0 - 9, A - F

Responding Parameters

`405`

Field 0, the response identifier.

`CMAC`

Field 1, the generated CMAC. This field will contain 16 hexadecimal characters.

`KCMAC Check Digits`

Field 2, the first four digits that result from encrypting zeros using the KCMAC key. If option 88 is enabled, this field will contain six check digits.

Table 6-23 Response 405: Generate CMAC using TDES

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	405
1	CMAC	16	0 - 9, A - F
2	KCMAC Check Digits	4, 6	0 - 9, A - F

Usage Notes

- Generate the AKB for the KCMAC key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text triple-length KCMAC Key:
  8AA83BF8CBDA10620BC1BF19FBB6CD58BC313D4A371CA8B5, check digits C8CC
- The KCMAC key in AKB format:
  1BDGE000,BAEBD923AA1215A9B4B1D6BE65EF332880FF4B4A937DE973,E72B8D32 DEBD8D1E
- Algorithm: 1
- Data Type: U
- Data Length: 64
- Data: 6BC1BEE22E409F96E93D7E117393172AAE2D8A571E03AC9C9EB76FA C45AF8E51

The command looks like this:

```
<305#1BDGE000,BAEBD923AA1215A9B4B1D6BE65EF332880FF4B4A937DE973,E72B
8D32DEBD8D1E#1#U#64#6BC1BEE22E409F96E93D7E117393172AAE2D8A571E03AC9
C9EB76FAC45AF8E51#>
```

The Network Security Processor returns the following response:

```
<405#33E6B1092400EAE5#C8CC#>
```

Receive APACS 40 Request Message (Command 324)

Command 324 is used to receive an APACS 40 request message. The Network Security Processor verifies the MAC on the message and returns the encrypted residue for generating future messages.

If the transaction message contains an encrypted PIN block that the Acquirer will be processing, the Network Security Processor will generate the transaction PIN key. The encrypted PIN block can subsequently be processed by the Network Security Processor using the transaction PIN key in PIN translation or PIN verification commands.

If the transaction message contains a tunneled PIN block, the PIN block is encrypted such that only the issuer can decrypt it. The Network Security Processor will generate a key named KEYVAL. The KEYVAL is provided in the response in AKB format, it must be sent securely to the issuer to process the PIN.

This command is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<324#Message Type#Header,EMFK.E(Terminal Register Key),MAC#AB#[CD]#
Message#>
```

Response

```
<424#Verification Indicator#[Header,EMFK.E(Residue),MAC]#
[PIN Key or KEYVAL]#Terminal Register Key Check Digits#>[CRLF]
```

Calling Parameters

`324`

Field 0, the command identifier.

`Message Type`

Field 1, the type of APACS 40 message. This field can contain one of these numeric values:

Value	Message Type
1	Administration Request or Request message without PIN.
3	Request message with PIN sent to the Acquirer.
4	Request message with PIN tunneled to the Issuer.

`Header,E`$_{MFK.E}$`(Terminal Key Register),MAC`

Field 2, the current Terminal Key Register value encrypted under the Master File Key. This key can be a 2key-3DES (double-length) key. If option 6C is enabled, this key can also be a 1key-3DES (single-length) key. This field must contain 74 characters. The following headers are supported: 1dKNE000 and 1dKNN000.

`AB`

Field 3, the A and B values as defined in section 10.5, Derivation Of cryptographic keys in the *APACS STANDARD 70 - BOOK 5.* This field must contain 16 hexadecimal characters.

`[CD]`

Field 4, the C and D values as defined in section 10.5, Derivation Of cryptographic keys in the *APACS STANDARD 70 - BOOK 5.* This field must be empty when the message type (field 1) contains the value 1 or 4. This field must contain 16 hexadecimal characters when the message type (field 1) contains the value 3.

`Message`

Field 5, the message to verify. This field must contain hexadecimal characters. The message must contain an even number of characters. The minimum length is 16 characters and the maximum length is 4096 characters.

Table 6-24 Command 324: Receive APACS 40 Request Message

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	3	324
1	Message Type	1	1, 3, 4
2	Header,E$_{MFK.E}$(Terminal Key Register),MAC	74	printable ASCII
3	AB	16	0 - 9, A - F
4	[CD]	0, 16	0 - 9, A - F
5	Message	16 - 4096	0 - 9, A - F

Responding Parameters

`424`

Field 0, the response identifier.

`Verification Indicator`

Field 1, verification indicator. This field will contain either the letter Y which indicates that the message verified or the letter N which indicates that the message did not verify. If the message did not verify response fields 2 and 3 will be empty.

`[Header,E`$_{MFK.E}$`(Residue),MAC]`

Field 2, the residue from the response MAC, it is right padded with 128 bits of random data and then encrypted under the Master File Key. This field is present only if the verification indicator (field 1) contains the letter Y. If present, this field will contain 74 characters. The AKB header will be 1IKNE000.

`[Header,E`$_{MFK.E}$`(PIN Key),MAC]`

Field 3, the derived PIN Key encrypted under the Master File Key. The header will be 1PUNE000. This key is used by the Acquirer to translate the encrypted PIN block sent in the message. This field is present only when the Message Type (field 1 of the command) contains the number 3 and the verification indicator (field 1 of the response) contains the letter Y. If present, this field will contain 74 characters.

OR

`[Header,E`$_{MFK.E}$`(KEYVAL),MAC]`

Field 3, the derived KEYVAL encrypted under the Master File Key. The header will be 1PKNE000. This key is used by the Issuer to translate the encrypted tunneled PIN block. This field is present only when the Message Type (field 1 of the command) contains the number 4 and the verification indicator (field 1 of the response) contains the letter Y. If present, this field will contain 74 characters.

`Terminal Key Register Check Digits`

Field 4, the check digits of the current Terminal Key Register value supplied in field 2 of the command. Check digits are the leftmost 4 digits of the result of encrypting all zeros with the Terminal Key Register. If option 88 is enabled, this field will contain six check digits.

Table 6-25 Response 424: Receive APACS 40 Request Message

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Response identifier	3	424
1	Verification Indicator	1	Y or N
2	[Header,E$_{MFK.E}$(Residue),MAC]	0, 74	printable ASCII
3	[Header,E$_{MFK.E}$(PIN Key),MAC] OR [Header,E$_{MFK.E}$(KEYVAL),MAC]	0, 74	printable ASCII
4	Terminal Key Register Check Digits	4, 6	0 - 9, A - F

Usage Notes

- Generate the AKB of the current Terminal Key Register value. This command does not support a 3key-3DES (triple-length) Terminal Key Register.

- The length of the Terminal Key Register determines the MAC algorithm.

Terminal Key Register Length	MAC algorithm
16 (single-length)	ISO - 9797-1 Algorithm 1
32 (double-length)	ISO - 9797-1 Algorithm 3

- The 16 character cleartext residue is right padded with 16 hexadecimal randomly generated characters prior to being formatted into the AKB returned in field 2 of the response.

- The KEYVAL returned in field 3 of the response must be sent to the Issuer. It is not adjusted to odd parity prior to encryption under the Master File Key. You can use command 1A to export this key.

- The PIN Key returned in field 3 of the response can be used by the Acquirer to process the encrypted PIN block. It is adjusted to odd parity prior to encryption under the Master File Key.

- This command does not translate or verify a PIN block. It verifies the message data and depending on the message type, generates either a PIN encryption key for the Acquirer or a KEYVAL for the Issuer.

- The Network Security Processor supports tunneling PINs to an external issuer, however it does not support the commands required by the issuer to process tunneled PINs.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Message Type = 1 (Request Message without PIN)

- Terminal Key Register = 0123456789ABCDEF FEDCBA9876543210
  in AKB format =
  1dKNE000,1ED75F0B0B1CFCC96F9A16828D43FE08A1D2DE47F6F999B2,63EF7649B15A090
  B

- AB = 0123456789ABCDEF

- Message = FEDCBA98765432100123456789ABCDEF5384AFC7

The command looks like this:

```
<324#1#1dKNE000,1ED75F0B0B1CFCC96F9A16828D43FE08A1D2DE47F6F999B2,63
EF7649B15A090B#0123456789ABCDEF##FEDCBA98765432100123456789ABCDEF53
84AFC7#>
```

The Network Security Processor response will be similar to this:

```
<424#Y#1IKNE000,BDB4D9DF9DD7133DE21F27A61ECE09F1AB110CA03AD591BE,52
08B2B954A2A1D1##08D7#>
```

Message Type = 3 (Request message with PIN sent to Acquirer)

- Terminal Key Register = 0123456789ABCDEF FEDCBA9876543210
  in AKB format =
  1dKNE000,1ED75F0B0B1CFCC96F9A16828D43FE08A1D2DE47F6F999B2,63EF7649B15A090B
- AB = 0123456789ABCDEF
- CD = 12345678ABCDEF12
- Message = FEDCBA98765432100123456789ABCDEF5384AFC7

The command looks like this:

```
<324#3#1dKNE000,1ED75F0B0B1CFCC96F9A16828D43FE08A1D2DE47F6F999B2,63
EF7649B15A090B#0123456789ABCDEF#12345678ABCDEF12#FEDCBA987654321001
23456789ABCDEF5384AFC7#>
```

The Network Security Processor response will be similar to this:

```
<424#Y#1IKNE000,BDB4D9DF9DD7133D4A18F5D0EE2CB9755797D3E4EA96F9AA,2F
F18B54B283C8A3#1PUNE000,E0426FB2CD9B4ABA6E75E836C593FFE262D198EC153
28232,36BED505BDECEC2A#08D7#>
```

Message Type = 4 (Request Message with PIN tunneled to Issuer)

- Terminal Key Register = 0123456789ABCDEF FEDCBA9876543210
  in AKB format =
  1dKNE000,1ED75F0B0B1CFCC96F9A16828D43FE08A1D2DE47F6F999B2,63EF7649B15A090B
- AB = 0123456789ABCDEF
- Message = FEDCBA98765432100123456789ABCDEF5384AFC7

The command looks like this:

```
<324#4#1dKNE000,1ED75F0B0B1CFCC96F9A16828D43FE08A1D2DE47F6F999B2,63
EF7649B15A090B#0123456789ABCDEF##FEDCBA98765432100123456789ABCDEF53
84AFC7#>
```

The Network Security Processor response will be similar to this:

```
<424#Y#1IKNE000,BDB4D9DF9DD7133DF217648D6B183379DE819DFFB187C20C,F4
0EA78FDD93CA8D#1PKNE000,FC522C7986AD993FEC7E871F44D9C3564F978B1B700
6C005,BB569C7AF7D67E3F#08D7#>
```

Generate APACS 40 Response Message (Command 325)

Command 325 is used to generate the APACS 40 response message. The Network Security Processor returns the encrypted residue for generating future messages, and also the Terminal Key Register that will be used in the next transaction.

This command supports a cleartext as well as an encrypted authorisation parameter.

This command is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

<325#Message Type#Header,$E_{MFK.E}$(Terminal Key Register),MAC#AB#
[Header,$E_{MFK.E}$(Auth Parameter Key),MAC]#Auth Param Data#
Header,$E_{MFK.E}$(Req MAC Residue),MAC#Message#>

Response

<425#MAC#Header,$E_{MFK.E}$(Resp MAC Residue),MAC#
Header,$E_{MFK.E}$(Terminal Key Register Next),MAC#
Terminal Key Register Check Digits#>[CRLF]

Calling Parameters

`325`

Field 0, the command identifier.

`Message Type`

Field 1, the type of APACS 40 message. This field can contain one of these numeric values:

Value	Message Type
1	Response message with a cleartext authorisation parameter.
2	Response message with an issuer encrypted authorisation parameter.

Header,$E_{MFK.E}$(Terminal Key Register),MAC

Field 2, the current Terminal Key Register value encrypted under the Master File Key. This key can be a 2key-3DES (double-length) key. If option 6C is enabled, this key can also be a 1key-3DES (single-length) key. This field must contain 74 characters. The following headers are supported: 1dKNE000 and 1dKNN000.

`AB`

Field 3, the A and B values as defined in section 10.5, Derivation Of cryptographic keys in the *APACS STANDARD 70 - BOOK 5*. This field must contain 16 hexadecimal characters.

`[Header,`$E_{MFK.E}$`(Auth Parameter Key),MAC]`

Field 4, the key used to encrypt the authorisation parameter encrypted under the Master File Key. This key can be a 2key-3DES (double-length) key. If option 6C is enabled this key can also be a 1key-3DES (single-length) key. The length of this key must be the same as the length of the Terminal Key Register supplied in field 2.

This field must contain 74 characters when the message type (field 1) contains the number 2, otherwise it must be empty. The header value must be 1DKNE000.

`Auth Parameter Data`

Field 5, the authorisation parameter data, which will be used to generate the response message. If the message type (field 1) contains the number 1, this field contains cleartext data. If the message type (field 1) contains the number 2, this field contains data that was encrypted by the issuer using their Authorisation Parameter Key. This field must contain 16 hexadecimal characters.

`Header,`$E_{MFK.E}$`(Req MAC Residue),MAC`

Field 6, the residue from the request MAC encrypted under the Master File Key. This field must contain 74 characters. The header value must be 1IKNE000.

`Message`

Field 7, the message data to be sent as the response message. This field must contain hexadecimal characters. The message must contain an even number of characters. The minimum length is 16 characters and the maximum length is 4096 characters.

Table 6-26 Command 325: Generate APACS 40 Response Message

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	325
1	Message Type	1	1 or 2
2	Header,$E_{MFK.E}$(Terminal Key Register),MAC	74	printable ASCII
3	AB	16	0 - 9, A - F
4	Header,$E_{MFK.E}$(Auth Parameter Key),MAC	74	printable ASCII
5	Authorisation Parameter Data	16	0 - 9, A - F
6	Header,$E_{MFK.E}$(Req MAC Residue),MAC	74	printable ASCII
7	Message	16 - 4096	0 - 9, A - F

Responding Parameters

`425`

Field 0, the response identifier.

`MAC`

Field 1, the Message Authentication Code (MAC). This field will contain the MAC of the response message. This field will contain 8 hexadecimal characters.

Header,$E_{MFK.E}$(Resp MAC Residue),MAC

Field 2, the 16 character cleartext residue from the response MAC is right-padded with 16 hexadecimal randomly generated characters prior to being formatted into the AKB and encrypted under the Master File Key. This field will contain 74 characters. The AKB header will be 1IKNE000.

Header,$E_{MFK.E}$(Terminal Key Register Next),MAC

Field 3, the Terminal Key Register, that will be used in the terminal's next transaction, encrypted under the Master File Key. This field will contain 74 characters. The AKB header will be 1dKNE000. The length of this key will be the same length as the Terminal Key Register (supplied in field 2 of the command).

`Terminal Key Register Check Digits`

Field 4, the check digits of the current Terminal Key Register value supplied in field 2 of the command. Check digits are the leftmost 4 digits of the result of encrypting all zeros with the Terminal Key Register. If option 88 is enabled, this field will contain six check digits.

Table 6-27 Response 425: Generate APACS 40 Response Message

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	424
1	MAC	8	0 - 9, A - F
2	[Header,$E_{MFK.E}$(Resp MAC Residue),MAC]	74	printable ASCII
3	[Header,$E_{MFK.E}$(Terminal Key Register Next),MAC]	74	printable ASCII
4	Terminal Key Register IN Check Digits	4, 6	0 - 9, A - F

Usage Notes

- Use the same current Terminal Key Register value to process all three message types (request - command 324, response - command 325, and optional confirmation - command 326) for a specific transaction from a specific terminal. This command does not support a 3key-3DES (triple-length) Terminal Key Register value.

- If necessary, generate the AKB of the Authorisation Parameter Key.
- The length of the Terminal Key Register determines the MAC algorithm.

Terminal Key Register Length	MAC algorithm
16 (single-length)	ISO - 9797-1 Algorithm 1
32 (double-length)	ISO - 9797-1 Algorithm 3

- Use the encrypted request MAC residue returned in field 2 of the response to command 324 as field 6 of this command.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Message Type = 1 (Response message with a cleartext authorisation parameter)

- Terminal Key Register = 0123456789ABCDEF FEDCBA9876543210
  in AKB format =
  1dKNE000,1ED75F0B0B1CFCC96F9A16828D43FE08A1D2DE47F6F999B2,63EF7649B15A090B

- AB = 0123456789ABCDEF

- Authorisation Parameter = 1234123456785678

- Request MAC Residue in AKB format =
  1IKNE000,BDB4D9DF9DD7133DD19C7768737D809A1448CC79CB98402C,8906FCB565DBE318

- Message = FEDCBA98765432100123456789ABCDEF

The command looks like this:

```
<325#1#1dKNE000,1ED75F0B0B1CFCC96F9A16828D43FE08A1D2DE47F6F999B2,63
EF7649B15A090B#0123456789ABCDEF##1234123456785678#1IKNE000,BDB4D9DF
9DD7133DD19C7768737D809A1448CC79CB98402C,8906FCB565DBE318#FEDCBA987
65432100123456789ABCDEF#>
```

The Network Security Processor response will be similar to this:

```
<425#BAB25A08#1IKNE000,E201C693EF0CAD4E371FBA6B3F3E68A0783EF45E24C9
6034,2E49E96797972EE1#1dKNE000,E4971092AB6DEABEEEF2CBB8F4D085AAB8BF
21376B71D34F,C73BFE1CCA7C08D5#08D7#>
```

Message Type = 2 (Response message with issuer encrypted auth param)

- Terminal Key Register = 0123456789ABCDEF FEDCBA9876543210
  in AKB format =
  1dKNE000,1ED75F0B0B1CFCC96F9A16828D43FE08A1D2DE47F6F999B2,63EF7649B15A090B
- AB = 0123456789ABCDEF
- Authorisation Parameter = 1234123456785678
- Encrypted Authorisation Parameter = CB585D1530F2479B
- Request MAC Residue in AKB format =
  1IKNE000,BDB4D9DF9DD7133DD19C7768737D809A1448CC79CB98402C,8906FCB565DBE318
- Message = FEDCBA98765432100123456789ABCDEF

The command looks like this:

```
<325#2#1dKNE000,1ED75F0B0B1CFCC96F9A16828D43FE08A1D2DE47F6F999B2,63
EF7649B15A090B#0123456789ABCDEF#1DKNE000,A7872E5AA0CD526980A81268B6
F0B5E9817EF5328D1690A3,E0D8B9DC4F6FE8D6#CB585D1530F2479B#1IKNE000,B
DB4D9DF9DD7133DD19C7768737D809A1448CC79CB98402C,8906FCB565DBE318#FE
DCBA98765432100123456789ABCDEF#>
```

The Network Security Processor response will be similar to this:

```
<425#A96C6135#1IKNE000,CB45510C8AB286BC13DE2F40A0B844B9663A6099CA80
B643,75B9294F3A77D445#1dKNE000,447DBF9751EA9911379A489178B582DCB238
DD7DB40C1F7C,2C69187B65685BD6#08D7#>
```

Verify APACS 40 Confirmation Message (Command 326)

Command 326 is used to verify the APACS 40 confirmation message. The Network Security Processor uses the Terminal Key Register, AB data, MAC residue, and message data to generate a MAC. It then compares the generated MAC to the MAC included in the confirmation message. If they are identical the confirmation message is verified.

This command is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<326#Message Type#Header,EMFK.E(Terminal Key Register),MAC#AB#
Header,EMFK.E(Resp MAC Residue),MAC#Message#>
```

Response

```
<426#Verification Indicator#Terminal Key Register IN Check Digits#>
[CRLF]
```

Calling Parameters

`326`

Field 0, the command identifier.

`Message Type`

Field 1, the confirmation APAC40 message type. This field must contain the number 1.

Header,$E_{MFK.E}$(Terminal Key Register),MAC

Field 2, the current Terminal Key Register value encrypted under the Master File Key. This key can be a 2key-3DES (double-length) key. If option 6C is enabled this key can also be a 1key-3DES (single-length) key. This field must contain 74 characters. The following headers are supported: 1dKNE000 and 1dKNN000.

`AB`

Field 3, the A and B values as defined in section 10.5, Derivation Of cryptographic keys in the *APACS STANDARD 70 - BOOK 5*. This field must contain 16 hexadecimal characters.

Header,$E_{MFK.E}$(Resp MAC Residue),MAC

Field 4, the residue from the response MAC key encrypted under the Master File Key. This field must contain 74 characters. The header must be 1IKNE000.

`Message`

Field 5, the confirmation message data to be verified. This field must contain hexadecimal characters. The message must contain an even number of characters. The minimum length is 16 characters and the maximum length is 4096 characters.

Table 6-28 Command 326: Verify APACS 40 Confirmation Message

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	326
1	Message Type	1	1
2	Header,$E_{MFK.E}$(Terminal Key Register),MAC	74	printable ASCII
3	AB	16	0 - 9, A - F
4	Header,$E_{MFK.E}$(Resp MAC Residue),MAC	74	printable ASCII
5	Message	16 - 4096	0 - 9, A - F

Responding Parameters

`426`

Field 0, the response identifier.

`Verification Indicator`

Field 1, verification indicator. This field will contain either the letter Y which indicates that the confirmation message verified or the letter N which indicates that the confirmation message did not verify.

`Terminal Key Register Check Digits`

Field 2, the check digits of the current Terminal Key Register value supplied in field 2 of the command. Check digits are the leftmost 4 digits of the result of encrypting all zeros with the Terminal Key Register. If option 88 is enabled, this field will contain six check digits.

Table 6-29 Response 426: Verify APACS 40 Confirmation Message

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	426
1	Verification Indicator	1	Y or N
2	Terminal Key Register Check Digits	4, 6	0 - 9, A - F

Usage Notes

- Use the same current Terminal Key Register value to process all three message types (request - command 324, response - command 325, and optional confirmation - command 326) for a specific transaction from a specific terminal. This command does not support a 3key-3DES (triple-length) Terminal Key Register.
- Use the encrypted response MAC residue returned in field 2 of the response to command 325 as field 4 of this command.
- The length of the Terminal Key Register determines the MAC algorithm.

Terminal Key Register Length	MAC algorithm
16 (single-length)	ISO - 9797-1 Algorithm 1
32 (double-length)	ISO - 9797-1 Algorithm 3

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Message Type = 1 (Verify confirmation message)

- Terminal Key Register = 0123456789ABCDEF FEDCBA9876543210
  in AKB format =
  1dKNE000,1ED75F0B0B1CFCC96F9A16828D43FE08A1D2DE47F6F999B2,63EF7649B15A090
  B
- AB = 0123456789ABCDEF
- Response MAC Residue in AKB format =
  1IKNE000,44CA29FB2FD8C70670C7F2B1C0D2DA4164CAFC5133EC8342,28AC6EA7F
  7E68056
- Message = FEDCBA98765432100123456789ABCDEF5384AFC7

The command looks like this:

```
<326#1#1dKNE000,1ED75F0B0B1CFCC96F9A16828D43FE08A1D2DE47F6F999B2,63
EF7649B15A090B#0123456789ABCDEF#1IKNE000,44CA29FB2FD8C70670C7F2B1C0
D2DA4164CAFC5133EC8342,28AC6EA7F7E68056#FEDCBA98765432100123456789A
BCDEF30E5229C#>
```

The Network Security Processor response will be:

```
<426#Y#08D7#>
```

Verify DUKPT MAC (Command 348)

Command 348 derives a Message Authentication session key using the Base Derivation Key and the key serial number. The session key is used to verify a MAC.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<348#Header,EMFK.E(Base Derivation Key),MAC#Key Serial Number#
[Header,EMFK.E(IV),MAC]#Data Continuation#[MAC Type]#Data Length#
MAC Data#[MAC]#Session Key Length#>
```

Response

```
<448#Data Continuation#Verification Flag or Intermediate IV#
Base Derivation Key Check Digits#KMAC Check Digits#>
```

Calling Parameters

`348`

Field 0, the command identifier.

Header,$E_{MFK.E}$(Base Derivation Key),MAC

Field 1, the Base Derivation Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. If option 6C is enabled, the AKB can contain a 1key-3DES (single-length) key. The following headers are supported: 1dDNE000 and 1dDNN000.

`Key Serial Number`

Field 2, the 10 to 20 digit Key Serial Number (KSN) from the PIN entry device. This field contains a 10 to 20 hexadecimal digit value.

[Header,$E_{MFK.E}$(IV),MAC]

Field 3, the Initialization Vector (IV) encrypted under the MFK. If this command contains the first block of multiple blocks of data, or if you are authenticating only one block of data, this field must be empty. The Network Security Processor will use an Initialization Vector of all zeros. If this command contains data subsequent to the first block in a multi-block series (that is, it contains continuation data), this field should contain the intermediate Initialization Vector from the previously sent data block. This field is either empty, or contains a 74 byte value. The following headers are supported: 1IDNE000 and 1IDNN000.

`Data Continuation`

Field 4, If field 7 contains all the data to be used to verify the MAC, set this field to 1. If the amount of data to be used in the MAC verification process exceeds 4096 ASCII hexadecimal characters, multiple commands are required to process the MAC. If this is the case, set this field to 0 for all commands except the command that contains the final block of data, when processing the last block of data set this field to 1.

The value of this field can be either:

- 0 - More data is coming in a subsequent command
- 1 - This command contains all the data, or contains the last block of data

`[MAC Type]`

Field 5, the type of MAC to be calculated. The possible values for this field are:

Value	MAC Type
Empty, or 1-6	ISO - 9797-1 Algorithm 1 - 1key or 2key-3DES Cipher block chaining
7	ISO 9797-1 Algorithm 3 - Only the last data block is processed using 3DES, all previous blocks are processed using single DES
V	Visa DUKPT (old style) as generated by command 5C

`MAC Data Length`

Field 6, the number of bytes of data supplied in field 7. The minimum data length is 2, the maximum data length is 4096.

`MAC Data`

Field 7, the data in ASCII hexadecimal format that was used to generate the MAC. This field contains a 2 - 4096 hexadecimal character value. This field must contain an even number of hexadecimal characters.

`[MAC]`

Field 8, the MAC to be verified. This field must contain eight hexadecimal digits (32 bits), or must be empty if the Data Continuation flag (field 4) contains a 0 (zero).

`Session Key Length`

Field 9, the length of the generated incoming PIN Encryption and MAC session keys. The value of this field can be either:

- S - generate a 1key-3DES (single length) session key
- D - generate a 2key-3DES (double-length) session key

If the Base Derivation Key, provided in field 1, is a 1key-3DES (single-length) key, this field must contain the letter S.

Table 6-30 Command 348: Verify DUKPT MAC

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	348
1	Header,$E_{MFK.E}$(Base Derivation Key),MAC	74	printable ASCII
2	Key Serial Number	10 - 20	0 - 9, A - F
3	[Header,$E_{MFK.E}$(IV),MAC]	0, 74	printable ASCII
4	Data Continuation	1	0, 1
5	[MAC Type]	0, 1	empty, 1 - 7, V
6	MAC Data Length	1 - 4	2 - 4096
7	MAC Data	2 - 4096	0 - 9, A - F
8	[MAC]	0, 8	empty, 0 - 9, A - F
9	Session Key Length	1	S or D

Responding Parameters

`448`

Field 0, the response identifier.

`Data Continuation`

Field 1, the value specified in field 4 of the command.

`Verification Flag or Intermediate IV`

Field 2, This field contains one of following values:

- Y – This value will be present only if field 4 of the command contains a 1, and the MAC verified.
- N – This value will be present only if field 4 of the command contains a 1, and the MAC did not verify.
- Intermediate IV - This value will be present only if field 4 of the command contains a zero. If present, this field will contain the AKB of the intermediate IV.

`Base Derivation Key Check Digits`

Field 3, check digits of the base derivation key. Check digits are the first six digits that result from encrypting zeros using the base derivation key.

```
KMAC Check Digits
```

Field 4, check digits of the generated Message Authentication Key (KMAC). Check digits are the first six digits that result from encrypting zeros using the KMAC.

Table 6-31 Response 448: Verify DUKPT MAC

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	448
1	Data Continuation	1	0, 1
2	Verification Flag or IV	1, 74	Y, N, or printable ASCII
3	Base Derivation Key Check Digits	6	0 - 9, A - F
4	KMAC Check Digits	6	0 - 9, A - F

Usage Notes

- Generate the AKB for the base derivation key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text Base Derivation Key: 0123456789ABCDEF FEDCBA9876543210
  The Base Derivation Key in AKB format:
  1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,342946FE884AA8B2
- Key serial number: 9876543210E00012
- The command contains all the data: there is no IV
- The MAC Type: ISO 9797-1 Algorithm 1 - 3DES CBC
- The MAC data: 0123456789ABCEF
- The MAC to be verified: 6FCEDEBD
- The DUKPT session key length: 2key-3DES (double-length)

The command looks like this:

```
<348#1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,3429
46FE884AA8B2#9876543210E00012##1##16#0123456789ABCDEF#6FCEDEBD#D#>
```

The Network Security Processor returns the following response:

```
<448#1#Y#08D7B4#B97051#>
```

Generate DUKPT MAC (Command 386)

Command 386 derives a Message Authentication session key using the Base Derivation Key and the key serial number. The session key is used to generate a MAC.

This command is not enabled in the Network Security Processor’s default security policy. To use this command, you must add it to the Network Security Processor’s security policy.

Command

```
<386#Header,EMFK.E(Base Derivation Key),MAC#Key Serial Number#
[Header,EMFK.E(IV),MAC]#Data Continuation#[MAC Type]#Data Length#
MAC Data#Session Key Length#>
```

Response

```
<486#Data Continuation#MAC or Intermediate IV#
Base Derivation Key Check Digits#KMAC Check Digits#>[CRLF]
```

Calling Parameters

`386`

Field 0, the command identifier.

Header,$E_{MFK.E}$(Base Derivation Key),MAC

Field 1, the Base Derivation Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. If option 6C is enabled, the AKB can contain a 1key-3DES (single-length) key. The following headers are supported: 1dDNE000 and 1dDNN000.

`Key Serial Number`

Field 2, the 10 to 20 digit Key Serial Number (KSN) from the PIN entry device. This field contains a 10 to 20 hexadecimal digit value.

[Header,$E_{MFK.E}$(IV),MAC]

Field 3, the Initialization Vector (IV) encrypted under the MFK. If this command contains the first block of multiple blocks of data, or if you are authenticating only one block of data, this field must be empty. The Network Security Processor will use an Initialization Vector of all zeros. If this command contains data subsequent to the first block in a multi-block series (that is, it contains continuation data), this field should contain the intermediate Initialization Vector from the previously sent data block. This field is either empty, or contains a 74 byte value. The following headers are supported: 1IDNE000 and 1IDNN000.

`Data Continuation`

Field 4, If field 7 contains all the data to be used to generate the MAC, set this field to 1. If the amount of data to be used in the MAC generation process exceeds 4096 ASCII hexadecimal characters, multiple commands are required to process the MAC. If this is the case, set this field to 0 for all commands except the command that contains the final block of data, when processing the last block of data set this field to 1.

The value of this field can be either:

- 0 - More data is coming in a subsequent command
- 1 - This command contains all the data, or contains the last block of data

`[MAC Type]`

Field 5, the type of MAC to be calculated. The possible values for this field are:

Value	**MAC Type**
Empty, or 1-6	ISO - 9797-1 Algorithm 1 - 1key or 2key-3DES Cipher block chaining
7	ISO 9797-1 Algorithm 3 - Only the last data block is processed using 3DES, all previous blocks are processed using single DES
V, VL or VR	Visa DUKPT (old style) as generated by command 5C. The Network Security Processor will return the left half of the MAC if this field contains “VL”, it will return the right half of the MAC if this field contains “VR”.

`MAC Data Length`

Field 6, the number of bytes of data supplied in field 7. The minimum data length is 2, the maximum data length is 4096.

`MAC Data`

Field 7, the data in ASCII hexadecimal format that was used to generate the MAC. This field contains a 2 - 4096 hexadecimal character value. This field must contain an even number of hexadecimal characters.

`Session Key Length`

Field 8, the length of the generated incoming PIN Encryption and MAC session keys. The value of this field can be either:

- S - generate a 1key-3DES (single length) session key
- D - generate a 2key-3DES (double-length) session key

If the Base Derivation Key, provided in field 1, is a 1key-3DES (single-length) key, this field must contain the letter S.

Table 6-32 Command 386: Generate DUKPT MAC

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	386
1	Header,$E_{MFK.E}$(Base Derivation Key),MAC	74	printable ASCII
2	Key Serial Number	10 - 20	0 - 9, A - F
3	[Header,$E_{MFK.E}$(IV),MAC]	0, 74	printable ASCII
4	Data Continuation	1	0, 1
5	[MAC Type]	0, 1	empty, 1 - 7, V
6	MAC Data Length	1 - 4	2 - 4096
7	MAC Data	2 - 4096	0 - 9, A - F
8	Session Key Length	1	S or D

Responding Parameters

`486`

Field 0, the response identifier.

`Data Continuation`

Field 1, the value specified in field 4 of the command.

`MAC or Intermediate IV`

Field 2, If field 4 of the command contains a 1, this field contains the 32 bit MAC represented as 8 hexadecimal characters. If field 4 of the command contains a 0 (zero) this field will contain the AKB of the intermediate Initialization Vector.

`Base Derivation Key Check Digits`

Field 3, check digits of the base derivation key. Check digits are the first six digits that result from encrypting zeros using the base derivation key.

`KMAC Check Digits`

Field 4, check digits of the generated Message Authentication Key (KMAC). Check digits are the first six digits that result from encrypting zeros using the KMAC.

Table 6-33 Response 486: Generate DUKPT MAC

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	486
1	Data Continuation	1	0, 1
2	MAC or Intermediate IV	8 or 74	0 - 9, A - F, or or printable ASCII
3	Base Derivation Key Check Digits	6	0 - 9, A - F
4	KMAC Check Digits	6	0 - 9, A - F

Usage Notes

- Generate the AKB for the base derivation key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text Base Derivation Key: 0123456789ABCDEF FEDCBA9876543210
  The Base Derivation Key in AKB format:
  1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,342946FE884AA8B2
- Key serial number: 9876543210E00012
- The command contains all the data: there is no Initialization Vector
- The MAC Type: ISO 9797-1 Algorithm 1 - 3DES CBC
- The MAC data: 0123456789ABCEF
- The DUKPT session key length: 2key-3DES (double-length)

The command looks like this:

```
<386#1dDNE000,791AC3DAFF7D8502293D9D241D9BB9A80806FA3825F670E9,3429
46FE884AA8B2#9876543210E00012##1##16#0123456789ABCDEF#
D#>
```

The Network Security Processor returns the following response:

```
<486#1#6FCEDEBD#08D7B4#B97051#>
```

Generate MAC using HMAC (Command 39B)

Command 39B generates a message authentication code (MAC) using the HMAC algorithm in accordance with RFC2104.

You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<39B#Header,EMFK.E(KMAC),MAC#[Header,EMFK.E(KMAC-part2),MAC]#
Algorithm#Data#>
```

Response

```
<49B#MAC#KMAC Check Digits#[KMAC-part2 Check Digits]#>[CRLF]
```

Calling Parameters

`39B`

Field 0, the command identifier.

`Header,EMFK.E(KMAC),MAC`

Field 1, the 80-256 bit KMAC key encrypted under the MFK. It will be used, by the algorithm specified in field 3, to generate the MAC. If the KMAC key is longer than 256 bits, this field must contain the leftmost 256 bits of the KMAC key and field 2 must be an AKB that contains the remainder of the KMAC key. This field contains a 74, 90, or 106 byte value. For HMAC-SHA1, the allowed headers are: 1MHNE000, 1MHNN000, 1MHGE000, and 1MHGN000. For HMAC-SHA256, the allowed headers are: 1M2NE000, 1M2NN000, 1M2GE000, and 1M2GN000.

`[Header,EMFK.E(KMAC-part2),MAC]`

Field 2, this field must contain an AKB only when the KMAC key is longer than 256-bits. This AKB contains the remainder of the KMAC key and will be a 74, 90, or 106 byte value. The AKB header must be the same as the one supplied in field 1. If the size of the KMAC key is 256 bits or less this field must be empty.

`Algorithm`

Field 3, the algorithm used to generate the MAC. This field must contain one of these values:

Value	**Algorithm**
H	HMAC-SHA1
2	HMAC-SHA256

`Data`

Field 4, the data to be MACed. The data must be in unpacked ASCII-hexadecimal format. This field must contain a minimum of 2 and a maximum of 4096 hexadecimal characters, and the length of this field must be a multiple of 2.

Table 6-34 Command 39B: Generate MAC using HMAC

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	39B
1	Header,$E_{MFK.E}$(KMAC),MAC	74, 90, or 106	printable ASCII
2	[Header,$E_{MFK.E}$(KMAC-part2),MAC]	0, 74, 90, or 106	printable ASCII
3	Algorithm	1	H or 2
4	Data	2 - 4096	0 - 9, A - F

Responding Parameters

`49B`

Field 0, the response identifier.

`MAC`

Field 1, the generated MAC. The length of this field is based on field 3 of the command. This field will contain a 40 hexadecimal character MAC when the algorithm is HMAC-SHA1. It will contain a 64 hexadecimal character MAC when the algorithm is HMAC-SHA256.

`KMAC Check Digits`

Field 2, The KMAC key check digits are the leftmost 14 digits that result from hashing the KMAC key using the SHA-256 algorithm. If the KMAC key is larger than 256 bits, this field will contain the check digits of the leftmost 256 bits of the KMAC key.

`[KMAC-part2 Check Digits]`

Field 3, this field will contain check digits only if there is an AKB in field 2 of the command, otherwise it will be empty. The check digits are the leftmost 14 digits that result from hashing the remainder of the KMAC key using the SHA-256 algorithm.

Table 6-35 Response 49B: Generate MAC using HMAC

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	49B
1	MAC	40, 64	0 - 9, A - F

Table 6-35 Response 49B: Generate MAC using HMAC (continued)

Field	Contents	Length (bytes)	Legal Characters
2	KMAC Check Digits	14	0 - 9, A - F
3	[KMAC-part2 Check Digits]	0, 14	0 - 9, A - F

Usage Notes

- The maximum size of the KMAC key is 512 bits. If the KMAC key is greater than 256 bits, you must supply AKBs in fields 1 and 2.
- Use command 39A to generate the KMAC key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text 256-bit KMAC Key:
  0123456789ABCDEFFEDCBA98765432101234123412341234567856785678 5678
- The KMAC Key in AKB format:
  1MHNE000,F05883DE1887118B2888AB6686B62D386333B63DBEC25CBD69B3609A6 60703AB051AF43F9DF29584,58643450A27C2CCF
- Algorithm: HMAC-SHA1
- Data: 0123456789ABCDEF
- Data in unpacked-ASCII hexadecimal format: 30313233343536373839414243444546

The command looks like this:

```
<39B#1MHNE000,F05883DE1887118B2888AB6686B62D386333B63DBEC25CBD69B36
09A660703AB051AF43F9DF29584,58643450A27C2CCF##H#3031323334353637383
9414243444546#>
```

The Network Security Processor returns the following response:

```
<49B#2599F753A6C1F185A5C8EBA3954908F959BBFEDD#79267CA84AF6B3##>
```

Verify MAC using HMAC (Command 39C)

Command 39C verifies a HMAC-SHA1 or HMAC-SHA256 message authentication code in accordance with RFC2104.

You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<39C#Header,EMFK.E(KMAC),MAC#[Header,EMFK.E(KMAC-part2),MAC]#
Algorithm#Data#MAC#>
```

Response

```
<49C#Flag#KMAC Check Digits#[KMAC-part2 Check Digits]#>[CRLF]
```

Calling Parameters

`39C`

Field 0, the command identifier.

`Header,`$E_{MFK.E}$`(KMAC),MAC`

Field 1, the 80-256 bit KMAC key encrypted under the MFK. It will be used, by the algorithm specified in field 3, to verify the MAC. If the size of the KMAC key is greater than 256 bits, this field must contain the leftmost 256 bits of the KMAC key and field 2 must be an AKB that contains the remainder of the KMAC. This field contains a 74, 90, or 106 byte value. For HMAC-SHA1, the allowed headers are: 1MHNE000, 1MHNN000, 1MHVE000, and 1MHVN000. For HMAC-SHA256, the allowed headers are: 1M2NE000, 1M2NN000, 1M2VE000, and 1M2VN000.

`[Header,`$E_{MFK.E}$`(KMAC-part2),MAC]`

Field 2, this field must contain an AKB only when the size of the KMAC key is greater than 256-bits. This AKB contains the remainder of the KMAC key and will be a 74, 90, or 106 byte value. The AKB header must be the same as the one supplied in field 1. If the size of the KMAC key is 256 bits or less, this field must be empty.

`Algorithm`

Field 3, the algorithm used to generate the MAC. This field must contain one of these values:

Value	Algorithm
H	HMAC-SHA1
2	HMAC-SHA256

`Data`

Field 4, the data that was used to generate the MAC. The data must be in unpacked ASCII-hexadecimal format. This field must contain a minimum of 2 and a maximum of 4096 hexadecimal characters, and the length of this field must be a multiple of 2.

`MAC`

Field 5, the MAC to be verified. This field must contain a minimum of 10 hexadecimal characters, and the length of this field must be a multiple of 2. The maximum length of a HMAC-SHA1 MAC is 40 hexadecimal characters. The maximum length of a HMAC-SHA256 MAC is 64 hexadecimal characters.

Table 6-36 Command 39C: Verify MAC using HMAC

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	39C
1	Header,$E_{MFK.E}$(KMAC),MAC	74, 90, or 106	printable ASCII
2	[Header,$E_{MFK.E}$(KMAC-part2),MAC]	0, 74, 90, or 106	printable ASCII
3	Algorithm	1	H or 2
4	Data	2 - 4096	0 - 9, A - F
5	MAC	10 - 64	0 - 9, A - F

Responding Parameters

`49C`

Field 0, the response identifier.

`Flag`

Field 1, the verification flag. This field will contain the MAC verification result.

Value	Meaning
Y	The MAC verified.
N	The MAC did not verify.

`KMAC Check Digits`

Field 2, The KMAC key check digits are the first 14 digits that result from hashing the KMAC key using the SHA-256 algorithm. If the size of the KMAC key is greater than 256 bits, this field will contain the check digits of the leftmost 256 bits of the KMAC key.

`[KMAC-part2 Check Digits]`

Field 3, this field will contain check digits only if there is an AKB in field 2 of the command, otherwise it will be empty. The check digits are the first 14 digits that result from hashing the remainder of the KMAC key using the SHA-256 algorithm.

Table 6-37 Response 49C: Verify MAC using HMAC

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	49C
1	Flag	1	Y or N
2	KMAC Check Digits	14	0 - 9, A - F
3	[KMAC-part2 Check Digits]	0, 14	0 - 9, A - F

Usage Notes

- The maximum size of the KMAC key can be 512 bits. If the KMAC key is greater than 256 bits, you must supply AKBs in fields 1 and 2.
- Use command 39A to generate the KMAC key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text 256-bit KMAC Key:
  0123456789ABCDEFFEDCBA987654321012341234123412345678567856785678
- The KMAC Key in AKB format:
  1MHNE000,F05883DE1887118B2888AB6686B62D386333B63DBEC25CBD69B3609A660703AB051AF43F9DF29584,58643450A27C2CCF
- Algorithm: HMAC-SHA1
- Data: 0123456789ABCDEF
- Data in unpacked-ASCII hexadecimal format: 30313233343536373839414243444546
- MAC: 2599F753A6C1F185A5C8EBA3954908F959BBFEDD

The command looks like this:

```
<39C#1MHNE000,F05883DE1887118B2888AB6686B62D386333B63DBEC25CBD69B36
09A660703AB051AF43F9DF29584,58643450A27C2CCF##H#3031323334353637383
9414243444546#2599F753A6C1F185A5C8EBA3954908F959BBFEDD#>
```

The Network Security Processor returns the following response:

```
<49C#Y#79267CA84AF6B3##>
```

7

Authorizing VISA, MasterCard, American Express and Discover Cards

VISA and MasterCard magnetic stripe card transactions are authorized using the same algorithm. When the algorithm is used for VISA transactions, it is called Card Verification Value (CVV). When the algorithm is used for MasterCard transactions, it is called Card Validation Code. American Express uses a different algorithm called a Card Security Code (CSC).

MasterCard PayPass transactions are protected using a value called the CVC3, which is generated by the PayPass chip for each transaction. VISA uses a dynamic Card Verification Value (dCVV) generated by the smart card to protect contactless smart card transactions.

This section explains the purpose of CVVs, dCVVs, CVCs, CVC3s, and CSCs, and describes the commands that are used to implement support for CVV/CVC/CSCs.

To skip this introduction, go to Table 7-1 on page 7-2 for a list of commands.

About CVVs, CVCs, and CSCs

VISA Card Verification Values (CVVs), MasterCard Card Validation Codes (CVCs), and American Express Card Security Code (CSCs) are check-values that confirm the validity of a magnetic stripe. Confirming the magnetic stripe's validity protects against the production of counterfeit cards that have account numbers which have been generated in sequential order based on a valid account number.

The CVV/CVC algorithm takes as its input the primary account number, expiration date, and service code. These values are on the magnetic stripe's first two tracks. The input is operated on by a pair of keys, that are contained in a single Atalla Key Block. The result – the CVV/CVC – is added to the card's magnetic stripe. For calculating the encoded CVC1, use the primary account number, card expiration date and the service code. For calculating the indent CVC2, use the primary account number, card expiration date, and "zero fill" the service code.

A static CVC3 uses the same algorithm as CVC1 and CVC2, the data inputs are the primary account number, card expiration date, and a service code value of 502. A static CVC3 can be generated or verified using commands 5D and 5E, respectively.

A dynamic CVC3 uses a different algorithm, the data inputs are the primary account number, card expiration date, service code, unpredictable number, and application transaction counter, value. Use command 359 to verify a dynamic CVC3.

note For specific applications Visa refers to the CVV by other similar names. The Cardholder Authentication Verification Value (CAVV) uses the same algorithm and data values as those used to generate and verify a CVV. The Integrated Card Verification Value (iCVV) also uses the CVV algorithm with a service code of ‘999’.

The CSC algorithm takes as its input the primary account number and expiration date. These values are on the magnetic stripe’s first two tracks. The input is operated on by a key pair contained in a single Atalla Key Block. The result – the CSC– is added to the card’s magnetic stripe.

The Discover algorithm is unique to Discover smart cards. Use command 35F to verify a Discover dynamic CVV.

CVV, dCVV, CVC, CVC3, and CSC Commands

The remainder of this section contains the command and response syntax for the VISA CVV, MasterCard CVC, and American Express CSC commands.

Quick Reference

Table 7-1 identifies each command by number, name, and purpose.

Table 7-1 CVV, dCVV, CVC, CVC3 and CSC Commands

Command	Name	Purpose
5D	Generate CVV/CVC	Generates a Card Verification Value/Card Validation Code
5E	Verify CVV/CVC	Verifies a Card Verification Value/Card Verification Code
357	Verify dCVV	Verifies a VISA dynamic Card Verification Value
359	Verify dynamic CVC3	Verifies a MasterCard dynamic Card Validation Code 3
35A	Verify CSC	Verifies a Card Security Codes
35B	Generate CSC	Generates Card Security Codes
35F	Verify DCVV	Verifies a Discover Dynamic Card Verification Value
36A	Verify AMEX Expresspay - Magstripe	Verifies an AMEX Expresspay value using the Magstripe mode

Generate CVV/CVC (Command 5D)

Command 5D generates a Visa Card Verification Value (CVV) or a MasterCard Card Validation Code (CVC). Visa and MasterCard use the same algorithm to generate their CVV or CVC value. Whenever the terms Card Verification Value or CVV are used in this manual, they also refer to Card Validation Code and CVC.

This command is not enabled in the Network Security Processor's default security policy.

Command

```
<5D#Algorithm#Header,EMFK.E(KCVV),MAC#Reserved#Data#>
```

Response

```
<6D#CVV#Check Digits##>[CRLF]
```

Calling Parameters

`5D`

Field 0, the command identifier.

`Algorithm`

Field 1, the algorithm identifier. This field may contain either 2 or 3. The standard algorithm for CVV is 3. Algorithm 2 is no longer recommended. A three alphanumeric character CVV is returned in field 1 of the response when the algorithm identifier is set to 2. An eight digit CVV is returned in field 1 of the response when the algorithm identifier is set to 3.

`Header,E`$_{MFK.E}$`(KCVV),MAC`

Field 2, the Card Verification Value key **pair** encrypted under the MFK. This field must contain a 2key-3DES (double-length) key, or if option 6C is enabled, the key can be a replicated single-length key. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1CDNE000, 1CDNN000, 1CDGE000, and 1CDGN000.

`Reserved`

Field 3, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor.

`Data`

Field 4, the data used to generate the Card Verification Value. The data in this field should be the primary account number, the card expiration date, and the service code. This field contains a 1 to 32 byte decimal value.

Table 7-2 Command 5D: Generate CVV/CVC

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	5D
1	Algorithm	1	2, 3
2	Header,$E_{MFK.E}$(KCVV),MAC*	74	printable ASCII
3	Reserved	0, 74	Empty, printable ASCII
4	Data	1 - 32	0 - 9

* Can be a volatile table location.

Responding Parameters

`6D`

Field 0, the response identifier.

`CVV`

Field 1, the generated Card Verification Value. When the algorithm identifier (specified in field one of the command) is two, this field will contain three alphanumeric characters. When the algorithm identifier is three, this field will contain 8 decimal digits.

`Check Digits`

Field 2, the Card Verification Value key check digits; the first four digits that result from encrypting zeros using the Card Verification Value key. If option 88 is enabled, this field will contain the first six digits of the result.

Table 7-3 Response 6D: Generate CVV/CVC

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	6D
1	CVV	varies	0 - 9, A - Z
2	Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- Generate the Card Verification Value Key pair.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.

See 3key-3DES Key (Triple-Length) for component values.

- Clear-text Card Verification Value key pair:
  0123 4567 89AB CDEF FEDC BA98 7654 3210.
  The key pair in AKB format:
  1CDNE000,4F7EFE3F44984427CF46B823CE4BDE1839E35E6F46EB2814,26A5545D383FA701

- The data used to generate the Card Verification Value is 4123 4567 8901 2345 8701 101 This value includes the following information:

- Primary account number: 4123 4567 8901 2345

- Card expiration date: 8701

- Service code: 101

The command looks like this:

```
<5D#3#1CDNE000,4F7EFE3F44984427CF46B823CE4BDE1839E35E6F46EB2814,26A
5545D383FA701##41234567890123458701101#>
```

The Network Security Processor returns the following response:

```
<6D#56149820#08D7##>
```

Verify CVV/CVC (Command 5E)

Command 5E verifies a Visa Card Verification Value (CVV) or a MasterCard Card Validation Code (CVC). Visa and MasterCard use the same algorithm to verify their CVV or CVC value. Whenever the terms Card Verification Value and CVV are used in this manual, they also refer to Card Validation Code and CVC.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<5E#Algorithm#Header,EMFK.E(KCVV),MAC#Reserved#Data#CVV#>
```

Response

```
<6E#Verification Flag#Check Digits##>[CRLF]
```

Calling Parameters

`5E`

Field 0, the command identifier.

`Algorithm`

Field 1, the algorithm identifier. This field may contain either 2 or 3. The standard algorithm is 3. Algorithm 2 is no longer recommended.

`Header,E`$_{MFK.E}$`(KCVV),MAC`

Field 2, the Card Verification Value key **pair** encrypted under the MFK. This field must contain a 2key-3DES (double-length) key. When option 6C is enabled, this key can be a 1key-3DES (single-length) key. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1CDNE000, 1CDNN000, 1CDVE000, and 1CDVN000.

`Reserved`

Field 3, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor.

`Data`

Field 4, the data used to verify the Card Verification Value. The data in this field should be the primary account number, the card expiration date, and the service code. This field contains a 1 to 32 byte decimal value.

`CVV`

Field 5, the Card Verification Value to be verified. When the algorithm identifier (field 1) is set to two, this field must contain a three alphanumeric character value. When the algorithm identifier (field 1) is set to three and option 4D is enabled, this field must contain a 3 to 8 digit value.

Table 7-4 Command 5E: Verify CVV/CVC

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	5E
1	Algorithm identifier	1	2, 3
2	Header,$E_{MFK.E}$(KCVV),MAC*	74	printable ASCII
3	Reserved	0, 74	Empty, printable ASCII
4	Data	1 - 32	0 - 9
5	CVV	varies	0 - 9, A - Z

* Can be a volatile table location.

Responding Parameters

`6E`

Field 0, the response identifier.

`Verification Flag`

Field 1, the verification flag. This field returns Y if the CVV verified, otherwise, it returns N.

`Check Digits`

Field 2, the Card Verification Value key pair check digits; the first four digits that result from encrypting zeros using the Card Verification Value Key. If option 88 is enabled this field will contain the first six digits of the result.

`Reserved`

Field 3, an empty field.

Table 7-5 Response 6E: Verify CVV/CVC

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	6E
1	Verification flag	1	Y, N
2	Check Digits	4 or 6	0 - 9, A - F
3	Reserved	0	

Usage Notes

- Generate the Card Verification Value key pair.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text Card Verification Value key pair:
  0123 4567 89AB CDEF FEDC BA98 7654 3210
  The key pair in AKB format:
  1CDNE000,4F7EFE3F44984427CF46B823CE4BDE1839E35E6F46EB2814,26A5545D3
  83FA701

- The data used to generate the Card Verification Value is 4123 4567 8901 2345 8701 101 This value includes the following information:

- Primary account number: 4123 4567 8901 2345

- Card expiration date: 8701

- Service code: 101

- The Card Verification Value to be verified: 56149820

The command looks like this:

```
<5E#3#1CDNE000,4F7EFE3F44984427CF46B823CE4BDE1839E35E6F46EB2814,26A
5545D383FA701##41234567890123458701101#56149820#>
```

The Network Security Processor returns the following response:

```
<6E#Y#08D7##>
```

Verify dCVV and dCVV2 (Command 357)

Command 357 verifies a VISA dynamic Card Verification Value generated by a contactless smart card. This command is enabled in the Network Security Processor's default security policy.

In version 1.60 and above, support for dCVV2 has been added.

Command

```
<357#Header,E_MFK.E(IMK_CVV),MAC#PAN#PAN Sequence Number#
Expiration Date#Service Code#ATC or TBN#dCVV or dCVV2#>
```

Response

```
<457#Verification Flag#UDK Check Digits#>[CRLF]
```

Calling Parameters

`357`

Field 0, the command identifier.

`Header,E_MFK.E(IMK_CVV),MAC`

Field 1, the 2key-3DES Issuer Master Key for CVV encrypted by the MFK. This field contains a 74 byte value. The following headers are supported: 1mENE000 and 1mENN000.

`PAN`

Field 2, Primary Account Number. This field contains a 3 through 19 digit value. When processing a dCVV2 the minimum lenght of the PAN is 8.

`PAN Sequence Number`

Field 3, the two digit sequence number which is appended to the PAN.

`Expiration Date`

Field 4, the four digit expiration date.

`Service Code`

Field 5, the three digit service code. When processing a dCVV2 this field must contain 888.

`ATC or TBN`

Field 6, the three or four digit Application Transaction Counter. When processing a dCVV2, this field contains the 8 decimal digit Time Based Number (TBN) calculated according to the Visa dCVV2 technical specification.

`dCVV or dCVV2`

Field 7, the three digit dynamic Card Verification Value to be verified.

Table 7-6 Command 357: Verify dCVV

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	357
1	Header,$E_{MFK.E}(IMK_{CVV})$,MAC	74	Printable ASCII
2	PAN	3 - 19, or 8 - 19	0 - 9
3	PAN Sequence Number	2	0 - 9
4	Expiration Date	4	0 - 9
5	Service Code	3	0 - 9
6	ATC or TBN	3 - 4, or 8	0 - 9
7	dCVV or dCVV2	3	0 - 9

Responding Parameters

`457`

Field 0, the response identifier.

`Verification Flag`

Field 1, the verification flag. This field returns Y if the dCVV or dCVV2 is verified, otherwise, it returns N.

`UDK Check Digits`

Field 2, the unique derived key check digits; the first four digits that result from encrypting zeros using the unique derived key. If option 88 is enabled, this field will contain the first six digits of the result.

Table 7-7 Response 457: Verify dCVV

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	457
1	Verification flag	1	Y, N
2	UDK Check Digits	4 or 6	0 - 9, A - F

Usage Notes

Before using Command 357 generate the Issuer Master Key.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify dCVV

- Clear-text Issuer Master Key is:
  0123 4567 89AB CDEF FEDC BA98 7654 3210.
  1mENE000,8AB6708833051A7258342A74919328D5BCAADF60322ABF10,A05A674C696D25AA

- Primary account number: 0123456789

- PAN sequence number: 00

- Card expiration date: 1204

- Service code: 555

- ATC: 666

- The dCVV to be authenticated: 505

The command looks like this:

```
<357#1mENE000,8AB6708833051A7258342A74919328D5BCAADF60322ABF10,A05A
674C696D25AA#0123456789#00#1204#555#666#505#>
```

The Network Security Processor returns the following response:

```
<457#Y#677E#>
```

Verify dCVV2

This example uses the same data values as the previous example, except:

- Service code: 888

- TBN: 12345678

- The dCVV2 to be authenticated: 946

The command looks like this:

```
<357#1mENE000,8AB6708833051A7258342A74919328D5BCAADF60322ABF10,A05A
```

```
674C696D25AA#0123456789#00#1204#888#12345678#946#>
```

The Network Security Processor returns the following response:

- `<457#Y#677E#>`

Verify dynamic CVC3 (Command 359)

Command 359 verifies a MasterCard dynamic Card Verification Code 3 (CVC3) generated by a PayPass smart card. This command is enabled in the Network Security Processor's default security policy.

In version 1.40 and above, support for PIN-CVC3 has been added, see [Type#].

Command

```
<359#Header,EMFK.E(IMKCVC3),MAC#PAN#[PAN Sequence Number]#
Track 1/2 Data#Unpredictable Number#ATC#dynamic CVC3#[Type#]>
```

Response

```
<459#Verification Flag#UDK Check Digits#>[CRLF]
```

Calling Parameters

`359`

Field 0, the command identifier.

`Header,EMFK.E(IMKCVC3),MAC`

Field 1, the 2key-3DES Issuer Master Key for dynamic CVC3 encrypted using the MFK. This field contains a 74 byte value. The following headers are supported: 1mENE000 and 1mENN000.

`PAN`

Field 2, Primary Account Number. This field is also used to indicate the Master Key derivation method. If this field contains the letter "B" followed by 17 to 19 decimal digits, method B will be used, otherwise method A will be used.

`[PAN Sequence Number]`

Field 3, the application PAN sequence number. This field is optional. If present, this field contains a 2 hexadecimal character value. If not present, the default value of a 00 will be used.

`Track 1/2 Data`

Field 4, the track 1 or track 2 data used to generate the dynamic CVC3. Track 1 data must be supplied as the hexadecimal representation of ASCII characters. For example, the number '5' is converted to 0x35, the letter 'A' is converted to 0x41. Track 2 data must be supplied as hexadecimal characters. If the track 2 data length is not an even number append a hexadecimal 'F'. The maximum length of this field is 160 hexadecimal characters. The length of this field must be a multiple of 16. The track data must be padded per these steps:

1. If the track length is a multiple of 16, add these 16 pad digits 8000000000000000, and then go to step 4. If not, go to step 2.
2. If the track length is not a multiple of 16, add these two pad digits ‘80’, and then go to step 3.
3. If the padded track data is a multiple of 16 go to step 4. If not, it is right-filled with hexadecimal zeros until it is a multiple of 16. Go to step 4.
4. The padding is complete.

`Unpredictable Number`

Field 5, the 8 digit unpredictable number.

`ATC`

Field 6, the 4 hexadecimal digit Application Transaction Counter.

`dynamic CVC3`

Field 7, the three to five digit dynamic Card Validation Code 3 value to be verified.

`[Type#]`

Field 8, this field is optional. When present, it defines the type of dynamic Card Validation Code 3 value to be verified. If this field is not present or contains the letter “S”, the NSP uses the standard CVC3 algorithm to verify the value supplied in field 7. When this field contains the letter “P”, the NSP uses the PIN-CVC3 algorithm to verify the value supplied in field 7.

Table 7-8 Command 359: Verify dynamic CVC3

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	3	359
1	Header,$E_{MFK.E}(IMK_{CVC3})$,MAC	74	Printable ASCII
2	PAN	1 - 20	0 - 9, B
3	[PAN Sequence Number]	0, 2	0 - 9, A - F
4	Track 1/2 Data	16 - 160	0 - 9, A - F
5	Unpredictable Number	8	0 - 9
6	ATC	4	0 - 9, A - F
7	dynamic CVC3	3 - 5	0 - 9
8	[Type#]	0,1	S or P

Responding Parameters

`459`

Field 0, the response identifier.

`Verification Flag`

Field 1, the verification flag. This field returns Y if the dynamic CVC3 is verified, otherwise, it returns N.

`UDK Check Digits`

Field 2, the unique derived key check digits; the first four digits that result from encrypting zeros using the unique derived key A. If option 88 is enabled, this field will contain the first six digits of the result.

Table 7-9 Response 459: Verify dynamic CVC3

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	459
1	Verification flag	1	Y, N
2	UDK Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- Before using Command 359 generate the Issuer Master Key.
- When field 8 of the command contains the letter P, the rightmost two bytes of the MAC is exclusive or'd (XOR) with the value 0x9559 to produce the initialization vector. See MasterCard's documentation for more information about PIN-CVC3.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text Issuer Master Key is:
  0123456789987654 3210012345678998, check digits 48F2
  1mENE000,3CF871A47137424D36A2F93C3EDBB42286077443D44717B9,5A231A765B CFBB2E

Track 1 example

- Primary account number: 5413123456784808
- PAN sequence number: 00

- Track 1 data: B5413123456784808^SUPPLIED/
NOT^0906101330003330002222200011110

  Track 1 data in hexadecimal: 42 35 34 31 33 31 32 33 34 35 36 37 38 34 38 30 38 5E 53 55 50 50 4C 49 45 44 2F 4E 4F 54 5E 30 39 30 36 31 30 31 33 33 30 30 30 33 33 33 30 30 30 32 32 32 32 32 30 30 30 31 31 31 31 30

- Unpredictable Number: 00000899
- ATC: 005E
- The dynamic CVC3 to be verified: 587

The command looks like this:

```
<359#1mENE000,3CF871A47137424D36A2F93C3EDBB42286077443D44717B9,5A23
1A765BCFBB2E#5413123456784808#00#4235343133313233343536373834383038
5E535550504C4945442F4E4F545E3039303631303133333030303333333030303323
2323232303030313131313308000#00000899#005E#587#>
```

The Network Security Processor returns the following response:

```
<459#Y#AF59#>
```

Track 2 example

- Primary account number: 5413123456784808
- PAN sequence number: 00
- Track 2 data in hexadecimal: 54 13 12 34 56 78 48 08 D0 90 61 01 90 00 99 00 00 00 0F
- Unpredictable Number: 00000899
- ATC: 005E
- The dynamic CVC3 to be verified: 572

The command looks like this:

```
<359#1mENE000,3CF871A47137424D36A2F93C3EDBB42286077443D44717B9,5A23
1A765BCFBB2E#5413123456784808#00#5413123456784808D0906101900099000
000F8000000000#00000899#005E#572#>
```

The Network Security Processor returns the following response:

```
<459#Y#AF59#>
```

Track 2 example - PIN-CVC3

- Primary account number: 5413123456784808

- PAN sequence number: 00
- Track 2 data in hexadecimal: 54 13 12 34 56 78 48 08 D0 90 61 01 90 00 99 00 00 00 0F
- Unpredictable Number: 00000899
- ATC: 005E
- The dynamic PIN-CVC3 to be verified: 973

The command looks like this:

```
<359#1mENE000,3CF871A47137424D36A2F93C3EDBB42286077443D44717B9,5A23
1A765BCFBB2E#5413123456784808#00#5413123456784808D0906101900099000
000F8000000000#00000899#005E#973#P#>
```

The Network Security Processor returns the following response:

```
<459#Y#AF59#>
```

Verify AMEX CSC (Command 35A)

Command 35A verifies an American Express Card Security Code (CSC). This command supports any combination of 3 digit, 4 digit and 5 digit CSC values.

In version 1.30 and above, this command can be used to verify either a version 1.0 or version 2.0 CSC.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<35A#Data#[Expiration Date]#Header,EMFK.E(KCSC),MAC#[CSC-5]#[CSC-4]#
[CSC-3]#>
```

Response

```
<45A#[VF5]#[VF4]#VF3]#KCSC Check Digits#>[CRLF]
```

Calling Parameters

`35A`

Field 0, the command identifier.

`Data`

Field 1, the contents of this field depends on the version of CSC to be verified.

To verify a CSC version 1.0 value, the 15 digit Primary Account Number (PAN) is entered in this field. The leftmost two digits must be either 34 or 37.

To verify a CSC version 2.0 value, this field must contain two 16 digit account blocks (a total of 32 digits). The account number consists of the rightmost 12 digits excluding the check digit. For account block 1, the 37 Card prepends the card expiration date to the 12 account number digits, the 34 Card appends the card expiration date to the 12 account number digits. Account block 2 is the 3 digit service coded right-padded with zeros.

`[Expiration Date]`

Field 2, the expiration date is entered in the YYMM format. This field contains a 4 byte decimal value. This field should be empty when field 1 contains a 32 byte decimal value.

Header,$E_{MFK.E}$(KCSC),MAC

Field 3, the KCSC encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1mXNE000, 1mXNN000, 1mXVE000, and 1mXVN000.

```
[CSC-5]
```

Field 4, contains the 5 digit CSC, or this field may be empty.

```
[CSC-4]
```

Field 5, contains the 4 digit CSC, or this field may be empty.

```
[CSC-3]
```

Field 6, contains the 3 digit CSC, or this field may be empty.

Table 7-10 Command 35A: Verify AMEX CSC

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier.	3	35A
1	Data	15, 32	0 - 9
2	[Expiration Date]	0,4	0 - 9
3	Header,$E_{MFK.E}$(KCSC),MAC	74*	printable ASCII
4	[CSC-5]	0, 5	0 - 9
5	[CSC-4]	0, 4	0 - 9
6	[CSC-3]	0. 3	0 - 9

* Can be a volatile table location.

Responding Parameters

```
45A
```

Field 0, the response identifier.

```
[VF5]
```

Field 1, the Verify Flag result for the CSC-5 value. This field contains a Y if the CSC-5 is verified, or a N if the CSC-5 is not verified. The field is empty if field 4 of the command was empty.

```
[VF4]
```

Field 2, the Verify Flag result for the CSC-4 value. This field contains a Y if the CSC-4 is verified or a N if the CSC-4 is not verified. The field is empty if field 5 of the command was empty.

```
[VF3]
```

Field 3, the Verify Flag result for the CSC-3 value. This field contains a Y if the CSC-3 is verified or a N if the CSC-5 is not verified. The field is empty if field 6 of the command was empty.

`KCSC Check Digits`

Field 4, check digits; the first four digits that result from encrypting zeros using the KCSC. If option 88 is enabled, this field will contain six check digits.

Table 7-11 Response 45A: Verify AMEX CSC

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	3	45A
1	[VF5]	0, 1	Y, N
2	[VF4]	0, 1	Y, N
3	[VF3]	0, 1	Y, N
4	KCSC Check Digits	4 or 6	0 - 9, A - F

Usage Notes

Generate the Card Security Code Key.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text Card Security Code Key (KCSC) is:
  0123 4567 89AB CDEF FEDC BA98 7654 3210.
  1mXNE000,823244EC4A4DFA0198097F067749BE0C9448676EB51ABAE2,459E9F9E
  E15CE583

Verify version 1.0 CSCs

- Account number: 371234567890123
- Card expiration date: 9912
- CSC-5 = 61247, CSC-4 = 8720, CSC-3 = 552

The command looks like this:

```
<35A#371234567890123#9912#1mXNE000,823244EC4A4DFA0198097F067749BE0C
9448676EB51ABAE2,459E9F9EE15CE583#61247#8720#552#>
```

The Network Security Processor returns the following response:

```
<45A#Y#Y#Y#08D7#>
```

Verify version 2.0 CSCs

- Account number: 375987654321001
- Card expiration date: 9912
- Service Code: 992
- 5-digit CSC = 72417, 4-digit CSC = 7998, 3-digit CSC = 746

The command looks like this:

```
<35A#99125987654321009920000000000000##1mXNE000,823244EC4A4DFA01980
97F067749BE0C9448676EB51ABAE2,459E9F9EE15CE583#72417#7998#746#>
```

The Network Security Processor returns the following response:

```
<45A#Y#Y#Y#08D7#>
```

Generate AMEX CSC (Command 35B)

Command 35B generates the American Express Card Security Codes (CSC). The CSC algorithm produces three codes; a 5-digit CSC, a 4-digit CSC, and 3-digit CSC.

In version 1.30 and above, this command can be used to generate either version 1.0 or version 2.0 CSCs.

This command has a high security exposure and is not enabled in the Network Security Processor’s default security policy. To use this command, you must enable it in the Network Security Processor’s security policy.

Command

```
<35B#Data#[Expiration Date]#Header,EMFK.E(KCSC),MAC#>
```

Response

```
<45B#CSC-5#CSC-4#CSC-3#KCSC Check Digits#>[CRLF]
```

Calling Parameters

`35B`

Field 0, the command identifier.

`Data`

Field 1, the contents of this field depends on the version of CSC to be generated.

To generate a CSC version 1.0 value, the 15 digit Primary Account Number (PAN) is entered in this field. The leftmost two digits must be either 34 or 37.

To generate a CSC version 2.0 value, this field must contain two 16 digit account blocks (a total of 32 digits). The account number consists of the rightmost 12 digits excluding the check digit. For account block 1, the 37 Card prepends the card expiration date to the 12 account number digits, the 34 Card appends the card expiration date to the 12 account number digits. Account block 2 is the 3 digit service coded right-padded with zeros.

`[Expiration Date]`

Field 2, the expiration date is entered in the YYMM format. This field contains a 4 byte decimal value. This field should be empty when field 1 contains a 32 byte decimal value.

`Header,`$E_{MFK.E}$`(KCSC),MAC`

Field 3, the KCSC encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. The following headers are supported: 1mXNE000, 1mXNN000, 1mXGE000, and 1mXGN000.

Table 7-12 Command 35B: Generate AMEX CSC

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier.	3	35B
1	Data	15, 32	0 - 9
2	[Expiration Date]	0, 4	0 - 9
3	Header,$E_{MFK.E}$(KCSC),MAC*	74	printable ASCII

* Can be a volatile table location.

Responding Parameters

`45B`

Field 0, the response identifier.

`CSC-5`

Field 1, the generated 5 digit CSC. This field contains a 5 byte decimal value.

`CSC-4`

Field 2, the generated 4 digit CSC. This field contains a 4 byte decimal value.

`CSC-3`

Field 3, the generated 3 digit CSC. This field contains a 3 byte decimal value.

`KCSC Check Digits`

Field 4, check digits; the first four digits that result from encrypting zeros using the KCSC. If option 88 is enabled, this field will contain six check digits.

.

Table 7-13 Response 45B: Generate AMEX CSC

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	3	45B
1	5 Digit CSC	5	0-9
2	4 Digit CSC	4	0-9
3	3 Digit CSC	3	0-9
4	KCSC Check Digits	4 or 6	0 - 9, A - F

Usage Notes

Generate the Card Security Code Key.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text Card Security Code Key (KCSC) is:
  0123 4567 89AB CDEF FEDC BA98 7654 3210.
  1mXNE000,823244EC4A4DFA0198097F067749BE0C9448676EB51ABAE2,459E9F9E
  E15CE583

Generate version 1.0 CSCs

- Account number: 371234567890123
- Card expiration date: 9912

The command looks like this:

```
<35B#371234567890123#9912#1mXNE000,823244EC4A4DFA0198097F067749BE0C
9448676EB51ABAE2,459E9F9EE15CE583#>
```

The Network Security Processor returns the following response:

```
<45B#61247#8720#552#08D7#>
```

Generate version 2.0 CSCs

- Account number: 375987654321001
- Card expiration date: 9912
- Service Code: 992

The command looks like this:

```
<35B#99125987654321009920000000000000##1mXNE000,823244EC4A4DFA01980
97F067749BE0C9448676EB51ABAE2,459E9F9EE15CE583#>
```

The Network Security Processor returns the following response:

```
<45B#72417#7998#746#08D7#>
```

Verify Discover DCVV (Command 35F)

This command verifies a Discover Dynamic Card Verification Value (DCVV) generated by a contactless smart card.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<35F#Header,EMFK.E(IMK-DCVV),MAC#PAN#Expiration Date#UN#ATC#DCVV#>
```

Response

```
<45F#Verification Flag#IMK-DCVV Check Digits#AUK-DCVV Check Digits#>
[CRLF]
```

Calling Parameters

`35F`

Field 0, the command identifier.

`Header,EMFK.E(IMK-DCVV),MAC`

Field 1, the Issuer Master Key for DCVV encrypted under the MFK. This key can be a 2key-3DES key, or a 3key-3DES key if the MFK is also a 3key-3DES key. The length of the derived key (AUK-DCVV) will be the same length as the incoming IMK-DCVV. This field contains a 74 byte value. The following headers are supported: 1mONE000 and 1mONN000.

`PAN`

Field 2, the Primary Account Number. This field contains a 14, 16, or 18 byte decimal value.

`Expiration Date`

Field 3, the expiration date is entered in the YYMM format. This field contains a 4 byte decimal value.

`UN`

Field 4, the unpredictable number. This field contains a 2 byte decimal value.

`ATC`

Field 5, the application transaction counter. This field contains a 4 byte decimal value.

`DCVV`

Field 6, the dynamic card verification value. This field contains a 3 byte decimal value.

Table 7-14 Command 35F: Verify Discover DCVV

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier.	3	35F
1	Header,$E_{MFK.E}$(IMK-DCVV),MAC*	74	printable ASCII
2	PAN	14, 16 or 18	0 - 9
3	Expiration Date	4	0 - 9
4	UN	2	0 - 9
5	ATC	4	0 - 9
6	DCVV	3	0 - 9

Responding Parameters

`45F`

Field 0, the response identifier.

`Verification Flag`

Field 1, the verification flag. This field returns Y if the DCVV is verified, Otherwise, it returns N.

`IMK-DCVV Check Digits`

Field 2, the first four digits that result from encrypting zeros using the Issuer Master Key-DCVV. If option 88 is enabled, this field will contain six check digits.

`AUK-DCVV Check Digits`

Field 3, the first four digits that result from encrypting zeros using the derived Account Unique Key-DCVV (AUK-DCVV). If option 88 is enabled, this field will contain six check digits.

.

Table 7-15 Response 45F: Verify Discover DCVV

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	3	45F
1	Verification Flag	1	Y or N
2	IMK-DCVV Check Digits	4, 6	0 - 9, A - F
3	AUK-DCVV Check Digits	4, 6	0 - 9, A - F

Usage Notes

- Before using this command, generate the IMK-DCVV key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify a Discover DCVV using a 2key-3DES IMK-DCVV

- Clear-text Issuer Master Key-DCVV (IMK-DCVV) is:
  0123456789ABCDEF FEDCBA9876543210, check digits = 08D7
  1mONE000,7F9641EBD66DBE3D9D9944F6E287E6399BC7C1B19B0453D3,1103E101E
  A61B930
- PAN: 6011111111111117
- Expiration date: 0801
- Unpredictable number: 56
- Application transaction counter: 1234
- DCVV: 204

The command looks like this:

```
<35F#1mONE000,7F9641EBD66DBE3D9D9944F6E287E6399BC7C1B19B0453D3,1103
E101EA61B930#6011111111111117#0801#56#1234#204#>
```

The Network Security Processor returns the following response:

```
<45F#Y#08D7#A522#>
```

verify a Discover DCVV using a 3key-3DES IMK-DCVV

- Clear-text Issuer Master Key-DCVV (IMK-DCVV) is:
  0123456789ABCDEF FEDCBA9876543210 1234123456785678,
  check digits = A535
  1mONE000,7F9641EBD66DBE3D9D9944F6E287E639292F28A81F700A3D,470A131F
  4BCDF89D
- PAN: 6011111111111117
- Expiration date: 0801

- Unpredictable number: 56
- Application transaction counter: 1234
- DCVV: 446

The command looks like this:

```
<35F#1mONE000,7F9641EBD66DBE3D9D9944F6E287E639292F28A81F700A3D,470A
131F4BCDF89D#6011111111111117#0801#56#1234#446#>
```

The Network Security Processor returns the following response:

```
<45F#Y#A535#F37E#>
```

Verify AMEX Expresspay value - Magstripe Mode (Command 36A)

This command verifies an American Express Expresspay value using the Magstripe mode.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<36A#Derivation Type#Header,EMFK.E(IMK-AMEX)#PAN#
PAN Sequence Number#Reserved#Partial AC#AC Padded Data Block#>
```

Response

```
<46A#Verification Flag#Session Key Check Digits#
IMK-AMEX Check Digits#>[CRLF]
```

Calling Parameters

`36A`

Field 0, the command identifier.

`Derivation Type`

Field 1, the AMEX derivation algorithm. This field must contain the number 3.

`Header,`$E_{MFK.E}$`(IMK-AMEX),MAC`

Field 2, the Issuer Master Key for AMEX encrypted under the MFK. This key can be a 2key-3DES key, or a 3key-3DES key if the MFK is also a 3key-3DES key. This field contains a 74 byte value. The following headers are supported: 1mENE000 and 1mENN000.

`PAN`

Field 3, the Primary Account Number. This field contains a 1-20 byte value. This field is also used to indicate the Master Key derivation method. If this field contains the letter "B" followed by 17 to 19 digits, method B will be used, otherwise method A will be used.

`PAN Sequence Number`

Field 4, the application PAN sequence number. This field contains a 2 byte decimal value. Applications that do not have a valid PAN sequence number should set this field to 00.

`Reserved`

Field 5, this field must be empty.

`Partial AC`

Field 6, the partial Application Cryptogram to be verified. It is formed by decimalizing the rightmost 3 bytes of the standard Application Cryptogram. This field contains a 5 byte decimal value.

`AC Padded Data Block`

Field 7, the data used to generate the Application Cryptogram version 02. The data block elements are the Unpredictable Number and the Application Transaction Counter (ATC). The data block is right padded with zeros such that the length of this field is a multiple of 16 characters. It is the host application's responsibility to collect all necessary data and format it for processing. The Network Security Processor does not pad the data.

Table 7-16 Command 36A: Verify AMEX Express pay value - Magstripe Mode

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier.	3	36A
1	Derivation Type	1	3
2	Header,$E_{MFK.E}$(IMK-AMEX),MAC	74	printable ASCII
3	PAN	1 - 20	0 - 9, B
4	PAN Sequence Number	2	0 - 9
5	Reserved	0	none
6	Partial AC	5	0 - 9
7	AC Padded Data Block	16 - 1024	0 - 9, A - F

Responding Parameters

`46A`

Field 0, the response identifier.

`Verification Flag`

Field 1, the verification flag. This field returns Y if the partial AC is verified, otherwise, it returns N.

`Session Key Check Digits`

Field 2, the first four digits that result from encrypting zeros using the session key. If option 88 is enabled, this field will contain six check digits.

`IMK-AMEX Check Digits`

Field 3, the first four digits that result from encrypting zeros using the Issuer Master Key-AMEX (IMK-AMEX). If option 88 is enabled, this field will contain six check digits.

Table 7-17 Response 46A: Verify AMEX Express pay value - Magstripe Mode

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	3	46A
1	Verification Flag	1	Y or N
2	Session Key Check Digits	4, 6	0 - 9, A - F
3	IMK-AMEX Check Digits	4, 6	0 - 9, A - F

Usage Notes

- Before using this command, generate the IMK-AMEX key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text Issuer Master Key-AMEX (IMK-AMEX) is:
  BA20C2FB2A57EF9D F8D65B7623DA73C4, check digits = 925F
  1mENE000,C43BBBCAAA7601FD3F2905E3B5D684EC03E825FF801CD27C,9B80EF3
  DBBBBFA2D
- PAN: 374245455400001
- PAN Sequence Number: 01
- Partial AC: 52195
- Unpredictable Number: 00004912
- Application Transaction Counter: 001803A00000
- AC Padded Data Block: 00004912001803A0000000000000000

The command looks like this:

```
<36A#3#1mENE000,C43BBBCAAA7601FD3F2905E3B5D684EC03E825FF801CD27C,9B
80EF3DBBBBFA2D#374245455400001#01##52195#00004912001803A00000000000
000000#>
```

The Network Security Processor returns the following response:

```
<46A#Y#8C6B#925F#>
```

8

Processing EMV and VISA Stored Value Cards

Europay, MasterCard, and VISA (EMV) have established a series of specifications for integrated circuit cards used in payment systems. These specifications are available at the following website: www.emvco.com. The Network Security Processor provides the ability to verify an Application Request Cryptogram (ARQC), and if successful return an Application Response Cryptogram (ARPC). It can also be used to generate a Message Authentication code, and generate the integrated circuit card master key.

The VISA Stored Value Card (VSVC) is VISA International's implementation of an electronic cash card application. This implementation uses a chip card to store cash value that can be spent with merchants who have the hardware to read and receive money from the chip card. The holder of a chip card can use an Automated Teller Machine (ATM) to reload cash into the card. The Network Security Processor is used in conjunction with an ATM to reload a VISA Stored Value card. The Network Security Processor does not support the initial personalization of the VSVC.

To skip this introduction, go to Table 8-2 on page 8-3 for a list of commands.

EMV Master Key Derivation

Annex A of the *EMV Integrated Circuit Card Specifications for Payment Systems, Book 2, Security and Key Management, Version 4.1, May 2004* documents an additional method "Option B" for generating the 16-byte ICC Master Key used for Application Cryptogram generation, issuer authentication, and secure messaging. Smart cards that are Common Core Definitions (CCD) compliant may require that option B be used to generate the ICC Master key.

VSVC Signatures

Three signatures, S1, S2, and S3, are used in the process of reloading a VSVC chip card. When a card reload transaction is requested at an ATM, an S1 Signature is generated by the chip card and sent through the ATM and host application to the Network Security Processor for verification. Upon verification of the S1 Signature, the Network Security Processor generates an S2 Signature which becomes part of the host authorization response to the transaction.

The S2 is sent through the ATM to the VSVC for verification. If the S2 is verified, the card adjusts its fund balance and calculates an S3 Signature to indicate completion of the transaction. The S3 Signature and related data are archived by the host and may be used in the future for non-repudiation. If a customer dispute occurs, the S3 Signature is used as proof of the transaction and may be sent to the Network Security Processor for verification.

VSVC signatures are generated using Data Encryption Standard (DES) encryption. For detailed descriptions of the VSVC signature generation algorithm, see DES Key Management for VSVC andVSVC Data Elements.

DES Key Management for VSVC

The DES key used to generate signatures is either a 1key-3DES (single-length) or 2key-3DES (double-length) VSVC Session Key. This key is calculated by encrypting the card-specific data, such as expiration-date and card transaction number, with a 1key-3DES (single-length) VSVC Diversified Key which is loaded in the card and is unique to each card.

The Diversified Key can be generated in the Network Security Processor by encrypting the bank-and-card-specific data, such as bank identification number and card serial number, using a VSVC Master Key. The VSVC Master Key is a double-length DES key that is encrypted under the Network Security Processor Master File Key (MFK). The cryptogram of the VSVC Master Key is stored on the host. When a VSVC transaction is requested, the VSVC Master Key is sent to the Network Security Processor, along with other data that are required to generate the Diversified Key, Session Key, and the signatures. See VSVC Data Elements for information about generating the Diversified Key and Session Key.

VSVC Data Elements

Table 8-1 lists the data elements and their token names used in Commands BE and BF.

Table 8-1 VSVC Data Elements

Data Element	**Token**	**Type**	**Length (bytes)**
VSVC Issuer BIN (Purse Provider ID)	PPiep	binary	3
Card Serial Number	IEPid	binary	5
Card Expiration Date	DEXPiep	binary	3
Transaction Number of IEP	NTiep	binary	2
Load Request Dollar Amount	Mlda	binary	4
Currency Code	CURRlda	binary	2
Currency Exponent	CEXPlda	binary	1
Balance of the IEP (chip card)	BALiep	binary	4

Table 8-1 VSVC Data Elements (continued)

Data Element	Token	Type	Length (bytes)
Acquirer BIN	PPSAMID	binary	4
ATM Date and Time	R	binary	4
Transaction Completion Code	CCiep	binary	2
Data used to generate S1 Signature	S1 Signature Data / S1 Data	binary	19
Data used to generate S2 Signature	S2 Signature Data / S2 Data	binary	7
Data used to generate S3 Signature	S3 Signature Data / S3 Data	binary	10
S1 Signature	S1 or S1 Signature	binary	8
S2 Signature	S2 or S2 Signature	binary	8
S3 Signature	S3 or S3 Signature	binary	8
Key Version	VKLiep	binary	1

Quick Reference

Table 8-2 identifies each command by number, name, and purpose.

Table 8-2 VSVC Signature and EMV Commands

Command	Name	Purpose
BE	Verify S1 and Generate S2 Signatures	Verifies a card reload request.
BF	Verify S3 Signature	Verifies a S3 Signature.
350	Verify ARQC and ARPC	Verifies an Application Request Cryptogram (ARQC), and if successful, returns an Application Response Cryptogram (ARPC), in accordance with Europay, MasterCard, and VISA standards.
351	EMV PIN Change	Facilitates the functions required when performing an EMV PIN Change with or without the current PIN.
352	Generate EMV MAC	Generates a Message Authentication code in accordance with Europay, MasterCard, and VISA standards.
354	Generate ICC MK	Returns the ICC Master Key encrypted under the Key Exchange Key.
356	Validate CAP Token	Verifies an application cryptogram (AC) or signs transaction data.
365	Verify Visa Cloud-Based Payments	Verifies either an MSD or qVSDC cryptogram.
3FC	EMV PIN change from IBM 3624 offset	Generates the PIN from the IBM3624 offset and incorporates it into an EMV PIN change message.
3FD	Authenticate multiple MasterCard EMV scripts	Generates EMV Message Authentication Codes using the derived session key. Up to five separate data values can be provided in the command.

Verify VSVC S1 Signature and Generate VSVC S2 Signature (Command BE)

Command BE is used to verify the S1 Signature and generate the S2 Signature. The S1 Signature was generated by the VSVC and sent to the ATM as a result of requesting a card reload transaction at an ATM.

This command is a premium value command, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<BE#Header,EMFK.E(VSVCMK),MAC#Ppiep#IEPid#DEXPiep#Ntiep#S1 Data#
S2 Data#>
```

Response

```
<CE#Verification Indicator#S2#Diversified Key Check Digits#
Session Key Check Digits#>[CRLF]
```

Calling Parameters

`BE`

Field 0, the command identifier.

Header,$E_{MFK.E}$(VSVCMK),MAC

Field 1, the VSVC Master Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. This key must be either a 2key- or 3key-3DES key. The following headers are supported: 1mVNE000 and 1mVNN000.

`PPiep`

Field 2, the Purse Provider Identifier. This field is used in the generation of the VSVC Diversified Key. This field contains a 3 byte binary value that has been converted to 6 ASCII hexadecimal characters.

`IEPid`

Field 3, the IEP (Interceptor Electronic Purse) Identifier. This field is used in the generation of the VSVC Diversified Key. This field contains a 5 byte binary value that has been converted to 10 ASCII hexadecimal characters.

`DEXPiep`

Field 4, the expiration date of the IEP. This field is used in the generation of the VSVC Session Key. This field contains a 3 byte binary value that has been converted to 6 ASCII hexadecimal characters.

`NTiep`

Field 5, the transaction number of the IEP. This field is used in the generation of the VSVC Session Key. This field contains a 2 byte binary value that has been converted to 4 ASCII hexadecimal characters.

`S1 Data`

Field 6, the S1 Signature data. This field represents the six concatenated data elements, used to generate the S1 Signature. It contains 38 ASCII hexadecimal characters. The data elements are:

- Mlda, Load Request Dollar amount, a 4 byte binary value that has been converted to 8 ASCII hexadecimal characters.
- CURRlda, Currency Code, a 2 byte binary value that has been converted to 4 ASCII hexadecimal characters.
- CEXPlda, Currency Exponent, a 1 byte binary value that has been converted to 2 ASCII hexadecimal characters.
- BALiep, Balance of the IEP, a 4 byte binary value that has been converted to 8 ASCII hexadecimal characters.
- PPSAMID, Acquirer BIN, a 4 byte binary value that has been converted to 8 ASCII hexadecimal characters.
- R, ATM Date and Time, a 4 byte binary value that has been converted to 8 ASCII hexadecimal characters.

`S1`

Field 7, the S1 Signature. This value is compared with the S1 Signature that is generated by the Network Security Processor. This field contains a 16 byte hexadecimal value.

`S2 Data`

Field 8, the S2 Signature data. This field represents the concatenated data elements, used to generate the S2 Signature. It contains 14 ASCII hexadecimal characters. The data elements are:

- Mlda, Load Request Dollar amount, a 4 byte binary value that has been converted to 8 ASCII hexadecimal characters.
- CURRlda, Currency Code, a 2 byte binary value that has been converted to 4 ASCII hexadecimal characters.
- CEXPlda, Currency Exponent, a 1 byte binary value that has been converted to 2 ASCII hexadecimal characters.

Table 8-3 Command BE: Verify VSVC S1 Signature

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	2	BE
1	Header,$E_{MFK.E}$(VSVCMK),MAC*	74	printable ASCII
2	PPiep	6	0 - 9, A - F
3	IEPid	10	0 - 9, A - F
4	DEXPiep	6	0 - 9, A - F
5	NTiep	4	0 - 9, A - F
6	S1 Signature Data	38	0 - 9, A - F
7	S1 Signature	16	0 - 9, A - F
8	S2 Signature Data	14	0 - 9, A - F

* Can be a volatile table location.

Responding Parameters

`CE`

Field 0, the response identifier.

`Verification Indicator`

Field 1, the S1 Signature verification indicator. This field contains ‘Y’ if the S1 Signature is verified, otherwise ‘N’ is returned.

`S2 Signature`

Field 2, the S2 Signature. This field contains a 16 byte hexadecimal value. This field is empty if the S1 Signature is not verified.

`Diversified Key Check Digits`

Field 3, the Diversified Key check digits; the first four digits that result from encrypting zeros using the Diversified Key. If option 88 is enabled, this field will contain six check digits.

`Session Key Check Digits`

Field 4, the Session Key check digits; the first four digits that result from encrypting zeros using the Session Key. If option 88 is enabled, this field will contain six check digits.

Table 8-4 Response CE: Verify VSVC S1 Signature

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	2	CE
1	Verification Indicator	1	Y or N
2	S2 Signature	0 or 16	0 - 9, A - F
3	Diversified Key Check Digits	4 or 6	0 - 9, A - F
4	Session Key Check Digits	4 or 6	0 - 9, A - F

Usage Notes

Before using Command BE generate the VSVC Master Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify an S1 signature and generate an S2 signature

- Clear-text VSVC Master Key: 7007 C1D5 EA19 0B98 BA75 E50B 89D0 2601
  The VSVC Master Key in AKB format:
  1mVNE000,238046ABFEB1D0FEBDBF663EFA990245087B3A4D7EBA33C5,E63E703 5EEAEB721
- PPiep: 451861
- IEPid: 0000000011
- DEXPied: 970731
- NTiep: 000E
- MIda: 00000001
- CURRIda: 0840
- CEXPIda: 02
- BALiep: 00002C0C
- PPSAMID: 0000002E
- R: 0000015A

- S1 Signature: BD50 9E29 0EDC BCDA

The command looks like this:

```
<BE#1mVNE000,238046ABFEB1D0FEBDBF663EFA990245087B3A4D7EBA33C5,E63E7
035EEAEB721#451861#0000000011#970731#000E#00000000108400200002C0C000
0002E0000015A#BD509E290EDCBCDA#00000001084002#>
```

The Network Security Processor returns the following response:

```
<CE#Y#0A4CA804206DD91C#2CF5#C78C#>
```

Verify VSVC S3 Signature (Command BF)

Command BF is used to verify the S3 Signature that is calculated by the VSVC after the S2 Signature is verified.

This command is a premium value command, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<BF#Header,EMFK.E(VSVCMK),MAC#PPiep#IEPid#DEXPiep#NTiep#S3 Data#S3#>
```

Response

```
<CF#Verification Indicator#Diversified Key Check Digits#
Session Key Check Digits#>[CRLF]
```

Calling Parameters

`BF`

Field 0, the command identifier.

`Header,E`$_{MFK.E}$`(VSVCMK),MAC`

Field 1, the VSVC Master Key encrypted under the MFK. This field contains a 74 byte value, or a volatile table location. This key must be either a 2key- or 3key-3DES key. The following headers are supported: 1mVNE000 and 1mVNN000.

`PPiep`

Field 2, the Purse Provider Identifier. This field is used in the generation of the VSVC Diversified Key. This field contains a 3 byte binary value that has been converted to 6 ASCII hexadecimal characters.

`IEPid`

Field 3, the IEP (Intersector Electronic Purse) Identifier. This field is used in the generation of the VSVC Diversified Key. This field contains a 5 byte binary value that has been converted to 10 ASCII hexadecimal characters.

`DEXPiep`

Field 4, the expiration date of the IEP. This field is used in the generation of the VSVC Session Key. This field contains a 3 byte binary value that has been converted to 6 ASCII hexadecimal characters.

`NTiep`

Field 5, the transaction number of the IEP. This field is used in the generation of the VSVC Session Key. This field contains a 2 byte binary value that has been converted to 4 ASCII hexadecimal characters.

`S3 Signature Data`

Field 6, the S3 Signature data. This field represents the concatenated data elements, used to generate the S3 Signature. It contains 20 ASCII hexadecimal characters. The data elements are:

- PPSAMID, Acquirer BIN, a 4 byte binary value that has been converted to 8 ASCII hexadecimal characters.
- R, ATM Date and Time, a 4 byte binary value that has been converted to 8 ASCII hexadecimal characters.
- CCiep, Transaction Completion Code, a 2 byte binary value that has been converted to 4 ASCII hexadecimal characters.

`S3 Signature`

Field 7, the S3 Signature. This field contains a 16 byte hexadecimal value.

Table 8-5 Command BF: Verify VSVC S3 Signature

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	2	BF
1	Header,$E_{MFK.E}$(VSVCMK),MAC*	74	printable ASCII
2	PPiep	6	0 - 9, A - F
3	IEPid	10	0 - 9, A - F
4	DEXPiep	6	0 - 9, A - F
5	NTiep	4	0 - 9, A - F
6	S3 Signature Data	20	0 - 9, A - F
7	S3 Signature	16	0 - 9, A - F

* Can be a volatile table location.

Responding Parameters

`CF`

Field 0, the response identifier.

`Verification Indicator`

Field 1, the S3 Signature verification indicator. This field contains 'Y' if S3 Signature is verified, otherwise 'N' is returned.

`Diversified Key Check Digits`

Field 2, the Diversified Key check digits; the first four digits that result from encrypting zeros using the Diversified Key. If option 88 is enabled, this field will contain six check digits.

`Session Key Check Digits`

Field 3, the Session Key check digits; the first four digits that result from encrypting zeros using the Session Key. If option 88 is enabled, this field will contain six check digits.

Table 8-6 Response CF: Verify VSVC S3 Signature

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	2	CF
1	Verification Indicator	1	Y or N
2	Diversified Key Check Digits	4 or 6	0 - 9, A - F
3	Session Key Check Digits	4 or 6	0 - 9, A - F

Usage Notes

Before using Command BF generate the VSVC Master Key.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify an S3 signature

- Clear-text VSVC Master Key: 7007 C1D5 EA19 0B98 BA75 E50B 89D0 2601
  The VSVC Master Key in AKB format:
  1mVNE000,238046ABFEB1D0FEBDBF663EFA990245087B3A4D7EBA33C5,E63E703 5EEAEB721
- PPiep: 4518 61
- IEPid: 0000 0000 11
- DEXPied: 9707 31

- NTiep: 000E
- PPSAMID: 0000 002E
- R: 0000 015A
- CCiep: 9000
- S3 Signature: 1B48 ED0A F1BA 1A98

The command looks like this:

```
<BF#1mVNE000,238046ABFEB1D0FEBDBF663EFA990245087B3A4D7EBA33C5,E63E7
035EEAEB721#451861#0000000011#970731#000E#0000002E0000015A9000#1B48
ED0AF1BA1A98#>
```

The Network Security Processor returns the following response:

```
<CF#Y#2CF5#C78C#>
```

Verify EMV ARQC (Command 350)

Command 350 will generate an EMV Authorization Request Cryptogram (ARQC) and compare it with an ARQC that is supplied in the command. If they match, an Authorization Response Cryptogram (ARPC) will be returned.

In version 1.42 and above, the UnionPay International derivation type is supported.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<350#EMV Derivation Type#Header,EMFK.E(IMK),MAC#Application PAN#
[Application PAN Sequence Number]#[Diversification Data]#
Authorization Request Cryptogram#Padded Data Block#
Authorization Response Code#[H#IV#Index#][Failure Response Code#]>
```

Response

```
<450#[Authorization Response Cryptogram]#Session Key Check Digits#
Issuer Master Key Check Digits#[Verification Indicator#]>[CRLF]
```

Calling Parameters

`350`

Field 0, the command identifier.

`EMV Derivation Type`

Field 1, the derivation technique used by the Network Security Processor to generate the ARQC and ARPC. This field contains a 1 byte decimal value defined as follows:

Value	Derivation Type
0	Europay/MasterCard MChip4 version 1.1
1, 3	VISA - CVN 10 and CVN 17
2	Common Session (EMV 4.1 and Specification Update Bulletin 46) - CVN 18 and CVN 22. EMV cryptogram type 5.
4	Union Pay International
8	EMV2000-Tree - CVN 14
9	EMV-Tree

Derivation types 0 and 9 use the derived session key to verify the ARQC and the ICC Master Key to generate the ARPC. Derivation types 1 and 3 use the ICC Master Key to generate both the ARQC and ARPC. Derivation types 2 and 8 use the session key for both ARQC and ARPC.

`Header,E`$_{MFK.E}$`(IMK),MAC`

Field 2, the Issuer Master Key encrypted under the MFK. This field contains a 74 byte value. This key can be a 2key- or 3key-3DES key. If option 6C is enabled, it can be a 1key-3DES key. The following headers are supported: 1mENE000 and 1mENN000.

`Application PAN`

Field 3, the Application Primary Account Number (PAN). The PAN can contain a minimum of 13 decimal digits and a maximum of 19 decimal digits. This field is also used to indicate the Master Key derivation method. When this field contains only decimal digits, Master Key derivation option A will be used. When this field starts with the letter “B” and is followed by 17 to 19 decimal digits, Master Key derivation option B will be used.

`[Application PAN Sequence Number]`

Field 4, the application PAN sequence number. This field contains a 2 hexadecimal character value. If empty, the default value of a 00 will be used.

`[Diversification Data]`

Field 5, the value of this field depends on the derivation type specified in field 1.

For the common session derivation algorithm (if the derivation type, Field 1, is 2), this field contains a 16 byte hexadecimal value consisting of the following two items:

- 2 byte Application Transaction Counter (ATC). This binary value is expressed as 4 hexadecimal characters.
- 6 byte fixed value. This binary value ‘000000000000’ is expressed as 12 hexadecimal characters.

For the UnionPay International derivation algorithm (if the derivation type, Field 1, is 4), this field contains a 2 byte Application Transaction Counter (ATC). This binary value is expressed as 4 hexadecimal characters.

For the EMV-Tree derivation algorithm (if the derivation type, Field 1, is 8 or 9), this field contains the four hexadecimal characters (2 bytes) of the Application Transaction Counter (ATC).

For the Visa derivation algorithm (if the derivation type, Field 1, is 1 or 3), this field must be empty.

For the Europay/MasterCard derivation algorithm (if the derivation type, Field 1, is 0), this field will contain either the same fields as the common session algorithm, or the four character ATC concatenated with 4 zero characters ‘0000’, followed by 4 bytes of hexadecimal characters (the unpredictable number).

`Authorization Request Cryptogram`

Field 6, the incoming Authorization Request Cryptogram (ARQC) to be validated. This field contains a 16 hexadecimal character value.

`Padded Data Block`

Field 7, the padded data block. The length of this field is 16 to 1024 bytes. The data block consists of transaction specific data elements generated by the terminal and smart card. To determine the contents of this field, refer to the Application Cryptogram Generation section of the *EMV Book 2 Security and Key Management* and also consult with your smart card personalization and smart card application vendors.

For derivation types 0, 2, 4, 8, and 9, the data should be right-padded with one 0x80 byte (expressed as two hexadecimal characters ‘80’), followed by a variable number of binary zeros bytes (expressed as two hexadecimal characters “00”) to make the total data length a multiple of 8 bytes (16 hexadecimal characters). If the data length is a multiple of 8, the data is padded with a single byte 80 (expressed as two hexadecimal characters ‘80’) followed by 7 bytes of binary zeros (expressed as 00000000000000).

For example, assume 37 bytes of data (expressed as 74 hexadecimal characters).

```
000000010000000000000000826000000800000
56000912002975E7015C00001600AB0975
```

The padding would contain 1 byte of hex 80 followed by 2 bytes of binary zero. `800000`

The padded data block would be 40 bytes (expressed as 80 hexadecimal characters):

```
000000010000000000000000826000000800000
56000912002975E7015C00001600AB0975800000
```

For derivation types 1 and 3, the data should be padded with a variable number of binary zeros bytes (expressed as 00 hex) to make the total data length a multiple of 8. If the data length is a multiple of 8, the data is padded with a 8 bytes of binary zeros (expressed as 0000000000000000).

For example, assume 30 bytes of data (expressed as 60 hexadecimal characters).

```
000000010000000000000000826000000800000
56000912002975E7015C
```

The padded data block would be 32 bytes (expressed as 64 hexadecimal characters):

```
000000010000000000000000826000000800000
56000912002975E7015C0000
```

The Network Security Processor does not enforce these data formats, it only requires that the length of data is a multiple of 16 hexadecimal characters.

`Authorization Response Code`

Field 8, the Authorization Response Code (ARC) used to calculate the ARPC if the ARQC verified. See [Failure Response Code#] if the ARQC does not verify.

If this field contains a 2 byte (4 hexadecimal characters) value, method 1 will be used to calculate the ARPC.

If method 2 should be used to calculate the ARPC, this field must contain the 4 byte (8 hexadecimal characters) Card Status Update value. Proprietary Authentication Data is optional, if present, it must be concatenated to the right of the Card Status Update value. The maximum size of the Proprietary Authentication Data is 8 bytes (16 hexadecimal characters). The ARPC is the leftmost 4 bytes (8 hexadecimal characters) of the MAC (ISO/IEC 9797-1 Algorithm 3). The data used in the MAC calculation is as follows:

```
ARQC||Card Status Update||Proprietary Authentication Data
```

`[H#IV#Index#]`

These next three fields are present only if the derivation type is 8 or 9, EMV-Tree derivation.

`[H#`

Field 9, the height value used for EMV-Tree derivation. This field contains the value 8 or 16, or it can be empty. If this field is empty, the height value of 8 will be used.

`IV#`

Field 10, the clear Initialization Vector used for EMV-Tree derivation. This field contains a 16 byte (32 hexadecimal characters) value, or it can be empty. If this field is empty, 16 bytes of 0 will be used.

`Index#]`

Field 11, the index value used for EMV-Tree derivation. The index specifies the byte location of the key that will be exclusive or'd with the ATC coefficient. An index value of zero indicates the leftmost byte of the key will be exclusive or'd with the ATC coefficient. An index value of 7 indicates the rightmost byte of the key will be exclusive or'd with the ATC coefficient. If the key is double-length, the index value is applied to both halves of the double length key.

This field contains a 1 digit decimal value between 0-7, or it can be empty. If this field is empty, the index value of 7 will be used.

`[Failure Response Code#]`

Field 12, the Failure Response Code (FRC) used to calculate the ARPC if the ARQC verification fails. This field contains a 2 byte (4 hexadecimal characters) value. This field can be present only when the derivation type (field 1) contains the value 2, 3, or 8. In version 1.42 and above, these additional derivation types 0 and 4 support the FRC.

Table 8-7 Command 350: Verify EMV ARQC

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	350
1	Derivation Type	1	0 - 4, 8, 9
2	Header,$E_{MFK.E}$(IMK),MAC	74	printable ASCII
3	Application PAN	13 - 20	0 - 9, B
4	[Application PAN Sequence Number]	empty, 2	0 - 9, A - F
5	[Diversification Data]*	0, 4, 16	0 - 9, A - F
6	Authorization Request Cryptogram	16	0 - 9, A - F
7	Padded Data Block**	16-1024	0 - 9, A - F
8	Authorization Response Code ARPC Method 1 ARPC Method 2	 4 8 - 24	 0 - 9, A - F 0 - 9, A - F
9	[H#	0-2	8, 16
10	IV#	0, 32	0 - 9, A - F
11	Index#]	0, 1	0 - 7
12	[Failure Response Code#]	4	0 - 9, A - F

* Empty if Field 1 contains a 1.
** Length must be a multiple of 16.

Responding Parameters

`450`

Field 0, the response identifier.

`[Authorization Response Cryptogram]`

Field 1, the Authorization Response Cryptogram. The length of this field depends upon the ARPC method; for method 1 this field contains a 8 byte (16 hexadecimal characters) value, for method 2 this field contains a 4 byte (8 hexadecimal character) value. This field is empty if ARQC did not verify and [Failure Response Code#] was not included in the command. This field will not be empty when the Verification Indicator is present.

`Session Key Check Digits`

Field 2, the first four digits of the result from encrypting zeros using the generated Session Key. If option 88 is enabled, this field will contain six check digits.

`Issuer Master Key Check Digits`

Field 3, the first four hexadecimal characters of the result from encrypting zeros using the Issuer Master Key. If option 88 is enabled, this field will contain six check digits.

`[Verification Indicator]`

Field 4, only signifies success or failure of the ARQC verification. This field will be present when the Failure Response Code (field 12 of the command) is present. This field will be omitted when the [Failure Response Code#] is not present. When the Verification Indicator is present, the ARPC (Authorization Response Cryptogram) field will not be empty. This field contains one character defined as follows:

Value	Description
N	ARQC verification failed and ARPC is calculated using the [Failure Response Code#]
Y	ARQC verification passed and ARPC is calculated using the ARC (Field 8)

Table 8-8 Response 450: Verify EMV ARQC

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	3	450
1	[Authorization Response Cryptogram]	0, 8, 16	0 - 9, A - F
2	Session Key Check Digits	4 or 6	0 - 9, A - F
3	Issuer Master Key Check Digits	4 or 6	0 - 9, A - F
4	[Verification Indicator]	1	Y, N

Usage Notes

- Before using this command, obtain the AKB for the Issuer Master Key.
- Visa CVNs are usable for both contact and contactless (Paywave) implementations. However, the contents of the Padded data block (Field 7) may be slightly different for these different applications. Refer to the Visa Integrated Circuit Card Specification (VIS) and Visa Contactless Payment Specification (VCPS) for more information.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Europay/MasterCard ARQC Verification using MK, Option A Master Key Derivation, ARPC Method 1

- Clear-text Issuer Master Key: 0123456789ABCDEF FEDCBA9876543210, check digits 08D7
  The Issuer Master Key in AKB format:
  1mENE000,8AB6708833051A7258342A74919328D5BCAADF60322ABF10,A05A674C696D25AA

- PAN: 9901234567890123

- Sequence Number: 45

- ATC + UN: 1234000012345678

- ARQC: 1F6BA35EE3DDD871

- DATA: 0123456789ABCDEF0123456789ABCDEF

- ARC: 0000

The command looks like this:

```
<350#0#1mENE000,8AB6708833051A7258342A74919328D5BCAADF60322ABF10,A0
5A674C696D25AA#9901234567890123#45#1234000012345678#1F6BA35EE3DDD87
1#0123456789ABCDEF0123456789ABCDEF#0000#>
```

The Network Security Processor returns the following response:

```
<450#73B46C01BEC191F7#4A5F#08D7#>
```

Europay/MasterCard ARQC Verification using derived Session Key, Option A Master Key Derivation, ARPC Method 1

- Clear-text Issuer Master Key: 0123456789ABCDEF FEDCBA9876543210, check digits 08D7
  The Issuer Master Key in AKB format:
  1mENE000,8AB6708833051A7258342A74919328D5BCAADF60322ABF10,A05A674C696D25AA

- PAN: 9901234567890123

- Sequence Number: 45

- ATC + UN: 1234000012345678

- ARQC: 1F6BA35EE3DDD871

- DATA: 0123456789ABCDEF0123456789ABCDEF

- ARC: 0001

- Failure Response Code: 0123

The command looks like this:

```
<350#2#1mENE000,8AB6708833051A7258342A74919328D5BCAADF60322ABF10,A0
5A674C696D25AA#9901234567890123#45#1234000012345678#1F6BA35EE3DDD87
1#0123456789ABCDEF0123456789ABCDEF#0001#0123#>
```

The Network Security Processor returns the following response:

```
<450#FF5A6133364B9505#4A5F#08D7#Y#>
```

Europay/MasterCard ARQC Verification using derived Session Key, Option A Master Key Derivation, ARPC Method 1

- Clear-text Issuer Master Key: 0123456789ABCDEF FEDCBA9876543210, check digits 08D7
  The Issuer Master Key in AKB format:
  1mENE000,8AB6708833051A7258342A74919328D5BCAADF60322ABF10,A05A674C696D25AA
- PAN: 9901234567890123
- Sequence Number: 45
- ATC+UN: 1234000012345678
- ARQC: 1F6BA35EE3DDD871
- DATA: 0123456789ABCDEF0123456789ABCDEF
- ARC: 0000
- Failure Response Code: 0123

The command looks like this:

```
<350#2#1mENE000,8AB6708833051A7258342A74919328D5BCAADF60322ABF10,A0
5A674C696D25AA#9901234567890123#45#1234000012345678#1F6BA35EE3DDD87
1#0123456789ABCDEF0123456789ABCDEF#0000#0123#>
```

The Network Security Processor returns the following response:

```
<450#25FD7457EA05B0CF#4A5F#08D7#Y#>
```

VISA ARQC Verification, Option A Master Key Derivation, ARPC Method 1

- Clear-text Issuer Master Key: 0123456789ABCDEF FEDCBA9876543210, check digits 08D7
  The Issuer Master Key in AKB format:
  1mENE000,8AB6708833051A7258342A74919328D5BCAADF60322ABF10,A05A674C696D25AA
- PAN: 9990123456789012
- Sequence Number: 45

- ARQC: EA53002B4A6AF97A
- DATA: 0123456789ABCDEF0123456789ABCDEF
- ARC: 0000

The command looks like this:

```
<350#1#1mENE000,8AB6708833051A7258342A74919328D5BCAADF60322ABF10,A0
5A674C696D25AA#9901234567890123#45##EA53002B4A6AF97A#0123456789ABC
DEF0123456789ABCDEF#0000#>
```

The Network Security Processor returns the following response:

```
<450#6962061DBF48E2A0#1DA5#08D7#>
```

EMV Tree Derivation ARQC Verification using derived Session Key, Option A Master Key Derivation, ARPC Method 1

- Clear-text Issuer Master Key: 589CA02B6BAC5BDD97238A7EDAF71298, check digits FDD1
  The Issuer Master Key in AKB format:
  1mENE000,A1BF8D48F9A4A61627219B2211AADDD3C4B6697EEE0110F5,C90434C6D398B7A9
- PAN = 9901234567890123
- Sequence Number = 45
- ATC = 293A
- ARQC = 4F5413D5EAB69B18 (match)
- Data = 0123456789ABCDEF0123456789ABCDEF
- ARC = EF12
- Height = 8
- IV = Null
- Index = 7

The command looks like this:

```
<350#8#1mENE000,A1BF8D48F9A4A61627219B2211AADDD3C4B6697EEE0110F5,C9
0434C6D398B7A9#9901234567890123#45#293A#4F5413D5EAB69B18#0123456789
ABCDEF0123456789ABCDEF#EF12#8###7#>
```

The Network Security Processor returns the following response:

```
<450#BA6EC017FBE0AF8D#7FA0#FDD1#>
```

UnionPay International ARQC Verification, Option A Master Key Derivation, ARPC Method 1

- Clear-text Issuer Master Key: 0123456789ABCDEF FEDCBA9876543210, check digits 08D7
  The Issuer Master Key in AKB format:
  1mENE000,8AB6708833051A7258342A74919328D5BCAADF60322ABF10,A05A674C696D25AA
- PAN: 9901234567890123
- Sequence Number: 45
- ATC: 0005
- ARQC: 5AD169A6A69F4C80
- DATA: 0123456789ABCDEF0123456789ABCDEF
- ARC: 0101
- FRC: 0000

The command looks like this:

```
<350#4#1mENE000,8AB6708833051A7258342A74919328D5BCAADF60322ABF10,A0
5A674C696D25AA#9901234567890123#45#0005#5AD169A6A69F4C80#0123456789
ABCDEF0123456789ABCDEF#0101#0000#>
```

The Network Security Processor returns the following response:

```
<450#BFB8E05555F6008C#62B7#08D7#Y#>
```

EMV PIN Change (Command 351)

Command 351 – Facilitates the functions required when performing an EMV PIN change with or without the current (old) PIN.

This command supports multiple output PIN blocks types based on the derivation type.

In version 1.40 and above, the Common Core Definition PIN block is supported.

In version 1.42 and above, the UnionPay International derivation types are supported.

To enable this command you must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

<351#Derivation Type#Incoming PIN Block type#[Reserved]#
Header, $E_{MFK.E}$(KPE),MAC#Header,$E_{MFK.E}$(IMK$_{ENC}$),MAC#
Header,$E_{MFK.E}$(IMK$_{MAC}$),MAC#[Header,$E_{MFK.E}$(IMK$_{AC}$),MAC]#
E_{KPE}(new PIN Block)#[PIN Issue Number]#Application PAN#
PAN Sequence Number#Diversification Data#[Application data]#
[PIN Block Data]#[E_{KPE}(old PIN Block)#][H#IV#Index#]>

Response

<451#Sanity Check#[Encrypted PIN block]#[MAC]#KPE Check Digits#
IMK$_{ENC}$ Check Digits#IMK$_{MAC}$ Check Digits#[IMK$_{AC}$ Check Digits]#
[SK$_{ENC}$ Check Digits]#[SK$_{MAC}$ Check Digits]#>[CRLF]

Calling Parameters

`351`

Field 0, the command identifier.

`Derivation Type`

Field 1, the derivation type. This field contains a 1 byte decimal value that describes both the session key derivation method and the type of outgoing PIN block to generate. Valid values are defined as follows:

Value	Derivation Type
0	Common Session (per EMV Version 4.1 and Specification Update Bulletin 46) derivation with ISO format 2 PIN block.
1	VISA derivation technique with VISA PIN block
2	VISA legacy derivation technique with VISA8 PIN block

Value	Derivation Type
3	EMV2000 (Tree-based technique) with ISO format 2 PIN block
4	EMV2000 (Tree-based technique) with VISA PIN block
5	EMV2000 (Tree-based technique) with legacy VISA8 PIN block
6	UnionPay International - XOR Session Key Left with PIN block
7	UnionPay International - XOR Master Key Left with PIN block
8	Common Core Definition PIN Block - Specification Update Bulletin 46 derivation with ISO format 2 PIN block with mandatory padding.

`Incoming PIN Block type`

Field 2, specifies the incoming PIN block type provided in fields 8 and 15. This field is 1 byte, it can contain the values 0, 1, or L. The following table identifies the value for each PIN block type.

Value	PIN Block Type
0	ISO format 1
1	ANSI
L	Lloyds

`[Reserved]`

Field 3, a reserved field. It should be empty.

Header,$E_{MFK.E}$(KPE),MAC

Field 4, the PIN Encryption Key (KPE) encrypted under the MFK. The KPE can be a 2key-3DES (double-length). If option 6C is enabled, it can be a 1key-3DES key. This field contains a 74 byte value. The following headers are supported: 1PUNE000, 1PUNN000, 1PUDE000, 1PUDN000, 1PDNE000, 1PDNN000, or 1PDDN000.

Header,$E_{MFK.E}$(IMK_{ENC}),MAC

Field 5, the Issuer Master Key for Message Confidentiality (IMK_{ENC}) encrypted under the MFK. The IMK_{ENC} can be 2key-3DES (double-length). If option 6C is enabled, it can be a 1key-3DES key. This field contains a 74 byte value. The following headers are supported: 1mENE00E or 1mENN00E.

Header,$E_{MFK.E}$(IMK_{MAC}),MAC

Field 6, the Issuer Master Key for Message Integrity (IMK_{MAC}) encrypted under the MFK. The IMK_{MAC} must be 2key-3DES (double-length). If option 6C is enabled, it can be a 1key-3DES key. This field contains a 74 byte value. The following headers are supported: 1mENE00M or 1mENN00M.

`[Header,`$E_{MFK.E}$`(`IMK_{AC}`),MAC]`

Field 7, the Issuer Master Key for Application Cryptogram generation (IMK_{AC}) encrypted under the MFK. When field 1 contains the number 0,3, or 8, this field must be empty. When field 1 contains a value of either 1 or 2, this field must contain a 74 byte value. The IMK_{AC} can be a 2key-3DES (double-length), or if option 6C is enabled, it can be a 1key-3DES key. The following headers are supported: 1mENE000 or 1mENN000.

E_{KPE}`(new PIN Block)`

Field 8, the encrypted PIN block. An error is returned if this PIN block fails the sanity check. This field contains a 16 byte hexadecimal value.

`[PIN Issue Number]`

Field 9, If field 2 contains the value ‘`L`’, this field will contain a decimal value between 000-255. Otherwise this field should be empty.

`Application PAN`

Field 10, the Application Primary Account Number (PAN). The PAN can contain a minimum of 13 decimal digits and a maximum of 19 decimal digits. This field is also used to indicate the Master Key derivation method. When this field contains only decimal digits, Master Key derivation option A will be used. When this field starts with the letter “B” and is followed by 17 to 19 decimal digits, Master Key derivation option B will be used.

`PAN Sequence Number`

Field 11, the Primary Account Number sequence number. This field contains a 2 digit decimal value.

`Diversification Data`

Field 12, the value of this field depends on the derivation type specified in field 1.

For the common session derivation algorithm (if the derivation type, Field 1, is 0 or 8), this field contains a 16 byte hexadecimal value as defined in EMV SU-46.

For all other derivation types, this field contains the four hexadecimal characters (2 bytes) of the Application Transaction Counter.

`[Application Data]`

Field 13, the APP Data field may contain the 5-byte EMV command message header (CLA, INS, P1, P2, and Lc) followed by other optional items such as the Application Transaction Counter (ATC), or the Application Cryptogram (ARQC). If the optional ATC and ARQC are included in the calculation of MAC, it is the application’s responsibility to prepend them in the Application data that is provided in the command. The Application data will be concatenated with the encrypted PIN block (i.e., Application data || Encrypted PIN block) to form a script message to calculate the MAC. The content of Lc byte is not validated or manipulated by the Network Security Processor, it must contain the appropriate value per EMV specification. It must be an even number of ASCII-hexadecimal characters. The maximum amount of data in this field is 3600 bytes.

`[PIN Block Data]`

Field 14, PIN block data. Its contents depend on the PIN block type. If field 2 contains ‘0’ or ‘L’, this field must be empty, otherwise it must contain the 12-digit PAN used to create the ANSI PIN block.

[E_{KPE}(old PIN Block)]

Field 15, the encrypted PIN block. This field is provided only if the old PIN will be exclusive or’d with the new PIN. If present, it should contain a 16 byte hexadecimal value, otherwise it should be empty.

`[H#IV#Index#]`

These next three fields are present only if the derivation type is 3, 4 or 5, EMV-Tree derivation.

`[H#`

Field 16, the height value used for EMV-Tree derivation. This field contains the value 8 or 16, or it can be empty. If this field is empty, the height value of 8 will be used.

`IV#`

Field 17, the clear Initialization Vector used for EMV-Tree derivation. This field contains a 32-byte hexadecimal value, or it can be empty. If this field is empty, an IV of 32-bytes of binary zeros (nulls) will be used.

`Index#]`

Field 18, the index value used for EMV-Tree derivation. The index specifies the byte location of the key that will be exclusive or’d with the ATC coefficient. An index value of zero indicates the leftmost byte of the key will be exclusive or’d with the ATC coefficient. An index value of 7 indicates the rightmost byte of the key will be exclusive or’d with the ATC coefficient. If the key is double-length, the index value is applied to both halves of the double length key.

This field contains a 1 digit decimal value between 0-7, or it can be empty. If this field is empty, the index value of 7 will be used.

Table 8-9 Command 351: PIN Change – EMV

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	3	351
1	Derivation type	1	0 - 8
2	Incoming PIN block type	1	0, 1, or L
3	[Reserved]	0	Empty

Table 8-9 Command 351: PIN Change – EMV (continued)

Field	Contents	Length (bytes)	Legal Characters
4	Header,$E_{MFK.E}$(KPE),MAC*	74	Correct Header, 0 - 9, A - F, and ‘,’
5	Header,$E_{MFK.E}$(IMK_{ENC}),MAC*	74	Correct Header, 0 - 9, A - F, and ‘,’
6	Header,$E_{MFK.E}$(IMK_{MAC}),MAC*	74	Correct Header, 0 - 9, A - F, and ‘,’
7	[Header,$E_{MFK.E}$(IMK_{AC}),MAC*]	0 or 74	Empty; or Correct Header, 0 - 9, A - F, and ‘,’
8	E_{KPE}(new PIN Block)	16	0 - 9, A - F
9	[PIN issue number]	0 or 3	Empty, or 000-255
10	Application PAN	13 - 20	0 - 9, B
11	PAN sequence number	2	0 - 9
12	Diversification Data	4 or 16	0 - 9, A - F
13	[Application data]	0 - 3600	Empty; or 0 - 9, A - F
14	[PIN block data]	0 or 12	Empty; or 0 - 9
15	[E_{KPE}(old PIN Block)]	0 or 16	Empty; or 0 - 9, A - F
16	[H#	0-2	8, 16
17	IV#	0, 32	0 - 9, A - F
18	Index#]	0, 1	0 - 7

* Can be a volatile table location.

Responding Parameters

```
451
```

Field 0, the response identifier.

```
Sanity Check
```

Field 1, the sanity check status. This field can contain one of the following values:

Value	Description
Y	Old PIN verified successfully.
LR	Indicated PIN length is less than the minimum PIN length

Value	Description
I	Incorrect rightmost padding characters
SR	Incorrect control field, indicated PIN length is greater than 12, or non-numeric PIN digits

In version 1.40 and above, this command supports option 4B. When option 4B is enabled, the only these PIN block digits are validated: control digit, PIN length digit, the first two PIN digits, and the last two padding digits.

`Encrypted PIN block`

Field 2, the PIN block encrypted by $\mathtt{SK}_{\mathtt{ENC}}$. This field is 0, 16, or 32 bytes. This field will contain an encrypted PIN block when field 1 of the response contains the letter Y, otherwise it will be empty.

When the Derivation Type is 0 or 3, the PIN block will be in the ISO Format-2 format.

When the Derivation Type is 1 or 4, the PIN block will be in the Visa format.

When the Derivation Type is 2 or 5, the PIN block will be in Visa8 format.

When the Derivation Type is 6 or 7, the PIN block will be in UnionPay International format.

When the Derivation Type is 8, the PIN block will be in ISO-2 format followed by a block of padding which consists of a byte of 0x80 followed by seven zero bytes. This two block structure will be 3DES-CBC encrypted using an all zero initialization vector.

`MAC`

Field 3, the MAC of the issuer script message. This field is 0 or 16 bytes.

`KPE Check Digits`

Field 4, the first six hexadecimal characters of the result from encrypting zeros using the key for PIN encryption.

$\mathtt{IMK}_{\mathtt{ENC}}$ `Check Digits`

Field 5, the first six hexadecimal characters of the result from encrypting zeros using the Issuer Master Key for message confidentiality ($\mathrm{IMK}_{\mathrm{ENC}}$).

$\mathtt{IMK}_{\mathtt{MAC}}$ `Check Digits`

Field 6, the first six hexadecimal characters of the result from encrypting zeros using the Issuer Master Key for message integrity ($\mathrm{IMK}_{\mathrm{MAC}}$).

`[IMK`$_{\mathtt{AC}}$ `Check Digits]`

Field 7, this field is empty if field 7 in the command is empty, otherwise this field contains the first six hexadecimal characters of the result from encrypting zeros using the Issuer Master Key for Application Cryptogram ($\mathrm{IMK}_{\mathrm{AC}}$).

`[SK`$_{ENC}$` Check Digits]`

Field 8, this field is empty if there is a sanity error, otherwise this field contains the first six hexadecimal characters of the result from encrypting zeros using the session key for message confidentiality (SK$_{ENC}$).

`[SK`$_{MAC}$` Check Digits]`

Field 9, this field is empty if there is a sanity error, otherwise this field contains the first six hexadecimal characters of the result from encrypting zeros using the message integrity (SK$_{MAC}$).

Table 8-10 Response 451: PIN Change – EMV

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	3	451
1	Sanity check status	1 or 2	Y, SR, LR, I
2	[Encrypted PIN block]	0, 16, or 32	0 - 9, A - F
3	[MAC]	0 or 16	0 - 9, A - F
4	KPE check digits	6	0 - 9, A - F
5	IMK$_{ENC}$ Check Digits	6	0 - 9, A - F
6	IMK$_{MAC}$ Check Digits	6	0 - 9, A - F
7	[IMK$_{AC}$ Check Digits]	0 or 6	0 - 9, A - F
8	[SK$_{ENC}$ Check Digits]	0 or 6	0 - 9, A - F
9	[SK$_{MAC}$ Check Digits]	0 or 6	0 - 9, A - F

Usage Notes

- The allowed minimum PIN length is 4. This command does not support option A0.
- The EMV VISA8 PIN block is constructed by exclusive oring the ISO format 0 PIN block with the derived AC ICC MK. This result is then encrypted using the session key. The EMV VISA8 PIN block is 16 characters in length.
- The EMV VISA PIN block is constructed by exclusive oring the ISO format 0 PIN block with the derived AC ICC MK. This result is left padded with application data and right padded to produce a 32 character value which is then encrypted using the session key. The EMV VISA PIN block is 32 characters in length.
- The Common Core Definition PIN block is an ISO format 2 PIN block that is right-padded with a fixed block of 8000000000000000. This value is encrypted using the session key. The Common Core Definition PIN block is 32 characters in length.

- For UnionPay International derivation type 6, the PIN block is constructed by exclusive oring the ISO format 0 PIN block with the derived Encryption (SMC) ICC MK. The result is then prepended with the length (0x08) and padded. The resulting block is encrypted with the session key.
- For UnionPay International derivation type 7, the PIN block is exactly like type 6, except the key used for the exclusive or is the session key rather than the ICC MK.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Europay/MasterCard, Option A Master Key Derivation

- ISO format 1 PIN block
- cleartext KPE: 0123456789ABCDEF FEDCBA9876543210
  The KPE in AKB format:
  1PUNE000,27F0026930C71646A3ADED3356C36BB0CCC2D6874DE6AB43,83237B4A
  EA731054
- cleartext IMK for confidentiality: 1234123456785678 8765876543214321
  The IMK for confidentiality in AKB format:
  1mENE00E,464C5BF3DAEA54ED57CDB348C5E948732AC2CD3B698EF626,90A92EB
  A3C936168
- cleartext IMK for integrity: ABCDABCDEF01EF01 10FE10FEDCBADCBA
  The IMK for integrity in AKB format:
  1mENE00M,65C24774BD1159AF77D0A4FFA9ABE64E9D5553ECB16F2EF3,5A4BAF84
  52E1E4EB
- PIN = 654321
- ISO Format 1 PIN Block = 16654321FFFFFFFF
- Incoming encrypted PIN block E_{KPEI}(PIN Block) = 30E96734FD6501AB
- Application PAN = 5555557890123456
- PAN Sequence Number = 73
- Unpredictable Number = 5093000087654321
- APP Data = 8424000210

The command looks like this:

```
<351#0#0##1PUNE000,27F0026930C71646A3ADED3356C36BB0CCC2D6874DE6AB43,
83237B4AEA731054#1mENE00E,464C5BF3DAEA54ED57CDB348C5E948732AC2CD3B69
8EF626,90A92EBA3C936168#1mENE00M,65C24774BD1159AF77D0A4FFA9ABE64E9D5
553ECB16F2EF3,5A4BAF8452E1E4EB##30E96734FD6501AB##5555557890123456#7
3#5093000087654321#8424000210##>
```

The Network Security Processor response is:

```
<451#Y#B5660CC137F464AF#ED8A944FA0DC75F0#08D7B4#61DEBE#718B4C##7356
D5#34EA9F#>
```

Europay/MasterCard Tree Derivation, Option B Master Key Derivation

- ISO format 1 PIN block
- cleartext KPE: 0123456789ABCDEF FEDCBA9876543210
  The KPE in AKB format:
  1PUNE000,27F0026930C71646A3ADED3356C36BB0CCC2D6874DE6AB43,83237B4A
  EA731054
- cleartext IMK for confidentiality: 1234123456785678 8765876543214321
  The IMK for confidentiality in AKB format:
  1mENE00E,464C5BF3DAEA54ED57CDB348C5E948732AC2CD3B698EF626,90A92EB
  A3C936168
- cleartext IMK for integrity: ABCDABCDEF01EF01 10FE10FEDCBADCBA
  The IMK for integrity in AKB format:
  1mENE00M,65C24774BD1159AF77D0A4FFA9ABE64E9D5553ECB16F2EF3,5A4BAF84
  52E1E4EB
- The encrypted new PIN block: 30E96734FD6501AB
- Application PAN: B5555557890123456 78
- Sequence Number: 73
- ATC: FFFF
- Application Data: 8424000210
- Height: 8
- IV: all zeros
- Index: 7

The command looks like this:

```
<351#3#0##1PUNE000,27F0026930C71646A3ADED3356C36BB0CCC2D6874DE6AB43,
83237B4AEA731054#1mENE00E,464C5BF3DAEA54ED57CDB348C5E948732AC2CD3B69
8EF626,90A92EBA3C936168#1mENE00M,65C24774BD1159AF77D0A4FFA9ABE64E9D5
553ECB16F2EF3,5A4BAF8452E1E4EB##30E96734FD6501AB##B55555578901234567
8#73#FFFF#8424000210###8###7#>
```

The Network Security Processor returns the following response:

```
<451#Y#8A7241B51BDDD782#9356AA4AAEB6004E#08D7B4#61DEBE#718B4C##0A70
0B#D300BD#>
```

Generate EMV MAC (Command 352)

Command 352 generates an EMV MAC.

In version 1.42 and above, the UnionPay International derivation type is supported.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<352#Derivation Type#Header,EMFK.E(IMK),MAC#[Application PAN]#
Application PAN Sequence Number#Diversification Data#MAC Length#
[Header,EMFK.E(Continuation-IV),MAC]#Padded Data#[H#IV#Index#]>
```

Response

```
<452#MAC Length#MAC or Header,EMFK.E(Continuation-IV),MAC#
KMAC Check Digits#Issuer Master Key Check Digits#>[CRLF]
```

Calling Parameters

`352`

Field 0, the command identifier.

`Derivation Type`

Field 1, the derivation type. This field contains a 1 byte decimal value defined as follows:

Value	Derivation Type
0	Common Session (per EMV 4.1 and Specification Update Bulletin 46)
1	VISA technique
4	Union Pay International
9	EMV2000 (Tree-based technique)

Header,$E_{MFK.E}$(IMK),MAC

Field 2, the Issuer Master Key encrypted under the MFK. This field contains a 74 byte value. This key can be either a 2key- or 3key-3DES key. If option 6C is enabled, it can be a 1key-3DES key. The following headers are supported: 1mENE00M and 1mENN00M.

`Application PAN`

Field 3, the Application Primary Account Number (PAN). The PAN can contain a minimum of 13 decimal digits and a maximum of 19 decimal digits. This field is also used to indicate the Master Key derivation method. When this field contains only decimal digits, Master Key derivation option A will be used. When this field starts with the letter "B" and is followed by 17 to 19 decimal digits, Master Key derivation option B will be used.

`Application PAN Sequence Number`

Field 4, the sequence number. This field contains a 2 digit decimal value.

`Diversification Data`

Field 5, the value of this field depends on the derivation type specified in field 1.

For the common session derivation algorithm (if the derivation type, Field 1, is 0) this field contains a 16 byte hexadecimal value as defined in EMV SU-46.

For all other derivation types, this field contains the four hexadecimal characters (2 bytes) of the Application Transaction Counter.

`MAC Length`

Field 6, the length of the MAC. This field contains one decimal digit in the range of 0 - 3. The following table indicates the possible MAC sizes.

Value	MAC Size
0	More data expected; no MAC verified
1	32 bits
2	48 bits
3	64 bits

A 32 bit MAC is expressed as eight hexadecimal characters (0 - 9, A - F) and written as two groups of four digits, separated by a space. A 48 bit or 64 bit MAC is expressed as three or four groups of four hexadecimal characters, separated by a space.

[Header,$E_{MFK.E}$(Continuation-IV),MAC]

Field 7, contains the continuation-IV, only if the MAC calculation is continued from a previous command. It must be empty for the first command of a multiple command sequence. This field contains either a 74 byte value, or is empty. The following header is supported: 1IDNE000.

`Padded Data`

Field 8, is the data used to calculate the MAC. The minimum length is 16 hexadecimal characters, the maximum length is 3,600 hexadecimal characters.

Per the EMV specification, the data should be right-padded with a single byte (expressed as two hexadecimal characters "80"), followed by a variable number of binary zeros bytes (expressed as two hexadecimal characters "00") to make the total data length a multiple of 8 bytes (16 hexadecimal characters). If the data length is a multiple of 8 bytes (expressed as 16 hexadecimal characters), the data is padded with a single byte (expressed as two hexadecimal characters "80") followed by 7 bytes of binary zeros (expressed as 00000000000000).

For example, assume 37 bytes of data (expressed as 74 hexadecimal characters).

```
12345678901234567890123456789012345678900000000010000000000000000082
600000080000056000912002975E7015C00001600AB0975
```

The padding would contain 1 byte of hex 80 followed by 2 bytes of binary zero. `800000`

The padded data block would be 40 bytes (expressed as 80 hexadecimal characters):

```
000000001000000000000000082600000080000056000912002975E7015C0000160
0AB0975800000
```

The Network Security Processor does not enforce this data format, it only requires that the length of data is a multiple of 16 hexadecimal bytes.

`[H#IV#Index#]`

These next three fields are present only if the derivation type is 9, EMV-Tree derivation.

`[H#`

Field 9, the height value used for EMV-Tree derivation. This field contains the value 8 or 16, or it can be empty. If this field is empty, the height value of 8 will be used.

`IV#`

Field 10, the clear Initialization Vector used for EMV-Tree derivation. This field contains a 32-byte hexadecimal value, or it can be empty. If this field is empty, an IV of 32-bytes of binary zeros (nulls) will be used.

`Index#]`

Field 11, the index value used for EMV-Tree derivation. The index specifies the byte location of the key that will be exclusive or'd with the ATC coefficient. An index value of zero indicates the leftmost byte of the key will be exclusive or'd with the ATC coefficient. An index value of 7 indicates the rightmost byte of the key will be exclusive or'd with the ATC coefficient. If the key is double-length, the index value is applied to both halves of the double length key.

This field contains a 1 digit decimal value in the range of 0-7, or it can be empty. If this field is empty, the index value of 7 will be used.

Table 8-11 Command 352: Generate EMV MAC

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command Identifier	3	352
1	EMV Derivation Type	1	0, 1, 4, o r 9
2	Header,$E_{MFK.E}$(IMK),MAC	74	printable ASCII

Table 8-11 Command 352: Generate EMV MAC (continued)

Field	Contents	Length (bytes)	Legal Characters
3	Application PAN	13 - 20	0 - 9, B
4	Application PAN Sequence Number	2	0 - 9
5	DIversification Data	4, 16	0 - 9, A - F
6	MAC Length	1	0 - 3
7	[Header,$E_{MFK.E}$(Continuation-IV),MAC]*	0, 74	printable ASCII
8	Padded Data**	16-4096	0 - 9, A - F
9	[H#	0-2	8, 16
10	IV#	0, 32	0 - 9, A - F
11	Index#]	0, 1	0 - 7

* Contains data only if the MAC calculation is continued from a previous command. It is empty in the first command of a multiple command sequence.
** Length must be a multiple of 16.

Responding Parameters

`452`

Field 0, the response identifier.

`MAC Length`

Field 1, the length of the MAC. This field contains the value of field 6 in the command.

`MAC` **or** `Header,`$E_{MFK.E}$`(Continuation-IV),MAC`

Field 2, if field 1 of the response is set to 0, this field will contain the Continuation-Initialization Vector encrypted under the MFK. If field 1 of the response is not 0, this field will contain the MAC.

If your use of this command results in the generation of an Continuation-IV in this field, input this value in subsequent MAC commands used to continue generating the MAC.

`KMAC Check Digits`

Field 3, the first four digits of the result from encrypting zeros using the derived Message Authentication Key (KMAC). If option 88 is enabled, this field will contain six check digits.

`Issuer Master Key Check Digits`

Field 4, the first four digits of the result from encrypting zeros using the Issuer Master Key. If option 88 is enabled, this field will contain six check digits.

Table 8-12 Response 452: Generate EMV MAC

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	3	452
1	MAC Length	1	0-3
2	MAC or Header,$E_{MFK.E}$(Continuation-IV),MAC	9, 14, 19 74	0 - 9, A - F; Space printable ASCII
3	KMAC Check Digits	4 or 6	0 - 9, A - F
4	Issuer Master Key Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- Generate the Issuer Master Key in AKB format.

Examples

VISA MAC Generation, Option A Master Key Derivation

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Clear-text Issuer Master Key: 160E 5EA2 D670 8083 DA13 1332 7051 62DF
  The Issuer Master Key in AKB format:
  1mENE00M,A2BEF4CF10E142EA38F8DB0CDB94D6DA297E1A403B232FD5,
  6001C35F600D2E06
- Application PAN: 9901234567890123
- Sequence Number: 45
- ATC: 0000
- MAC Type: 3
- DATA: 12345678901234560123456789ABCDEF

The command looks like this:

```
<352#1#1mENE00M,A2BEF4CF10E142EA38F8DB0CDB94D6DA297E1A403B232FD5,60
01C35F600D2E06#9901234567890123#45#0000#3##1234567890123456012345 67
89ABCDEF#>
```

The Network Security Processor returns the following response:

```
<452#3#E719 FD96 FE66 BEB5#CF72#5128#>
```

EMV-Tree MAC Generation, Option A Master Key Derivation

- Clear-text Issuer Master Key: 589CA02B6BAC5BDD 97238A7EDAF71298
  The Issuer Master Key in AKB format:
  1mENE00M,C40CF8DE7760901DA28CC0300D4C8E7E5FDDF69704636C7D,7C66412
  3FFA54B64
- Application PAN: 9901234567890123
- Sequence Number: 45
- ATC: 293A
- MAC Length: 3
- Data: 0123456789ABCDEF0123456789ABCDEF
- Height: 8
- IV: all zeros
- Index: 7

The command looks like this:

```
<352#9#1mENE00M,C40CF8DE7760901DA28CC0300D4C8E7E5FDDF69704636C7D,7C
664123FFA54B64#9901234567890123#45#293A#3##0123456789ABCDEF01234567
89ABCDEF#8##7#>
```

The Network Security Processor returns the following response:

```
<452#3#4F54 13D5 EAB6 9B18#7FA0#FDD1#>
```

EMV-Tree MAC Generation, Option B Master Key Derivation

- Clear-text Issuer Master Key: 589CA02B6BAC5BDD 97238A7EDAF71298
  The Issuer Master Key in AKB format:
  1mENE00M,C40CF8DE7760901DA28CC0300D4C8E7E5FDDF69704636C7D,7C66412
  3FFA54B64
- Application PAN: B9901234567890123456
- Sequence Number: 45
- ATC: 293A
- MAC Length: 3

- Data: 0123456789ABCDEF0123456789ABCDEF
- Height: 8
- IV: all zeros
- Index: 7

The command looks like this:

```
<352#9#1mENE00M,C40CF8DE7760901DA28CC0300D4C8E7E5FDDF69704636C7D,7C
664123FFA54B64#B99012345678901234456#45#293A#3##0123456789ABCDEF0123
456789ABCDEF#8##7#>
```

The Network Security Processor returns the following response:

```
<452#3#4699 6EDD D16F 0BD9#5492#FDD1#>
```

Generate EMV ICC Master Key (Command 354)

This command generates the Integrated Circuit Card Master Key and returns it encrypted under a Key Exchange Key.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<354#Header,EMFK.E(IMK),MAC#Application PAN#
Application PAN Sequence Number#Header,EMFK.E(KEK),MAC#>
```

Response

```
<454#EKEK(ICC Master Key)#ICC Master Key Check Digits#
Issuer Master Key Check Digits#Key Exchange Key Check Digits#>[CRLF]
```

Calling Parameters

`354`

Field 0, the command identifier.

`Header,EMFK.E(IMK),MAC`

Field 1, the Issuer Master Key encrypted under the MFK. This field contains a 74 byte value. This key can be either a 2key- or 3key-3DES key. If option 6C is enabled, it can be a 1key-3DES key. The following headers are supported: 1miNE000, 1miNN000, 1mENE000 and 1mENN000. In version 1.40 and above, these additional headers are supported: 1mENE00M, 1mENN00M, 1mENE00E and 1mENN00E.

`Application PAN`

Field 2, the Application Primary Account Number (PAN). The PAN can contain a minimum of 13 decimal digits and a maximum of 19 decimal digits. This field is also used to indicate the Master Key derivation method. When this field contains only decimal digits, Master Key derivation option A will be used. When this field starts with the letter “B” and is followed by 17 to 19 decimal digits, Master Key derivation option B will be used.

`Application PAN Sequence Number`

Field 3, the sequence number. This field contains a 2 digit decimal value.

`Header,EMFK.E(KEK),MAC`

Field 4, is the Key Exchange Key encrypted under the MFK. This field contains a 74 byte value. This key must be either a 2key- or 3key-3DES key. The following headers are supported: 1KDEE000 and 1KDEN000.

Table 8-13 Command 354: Generate ICC Master Key

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	354
1	Header,$E_{MFK.E}$(IMK),MAC	74	printable ASCII
2	Application PAN	13 - 20	0 - 9, B
3	Application PAN Sequence Number	2	0 - 9
4	Header,$E_{MFK.E}$(KEK),MAC	74	printable ASCII

Responding Parameters

`454`

Field 0, the response identifier.

`E`$_{KEK}$`(ICC Master Key)`

Field 1, the ICC Master key encrypted under the Key Exchange Key. This field will contain 32 hexadecimal characters.

`ICC Master Key Check Digits`

Field 2, the first four digits of the result from encrypting zeros using the derived ICC Master Key. If option 88 is enabled, this field will contain six check digits.

`Issuer Master Key Check Digits`

Field 3, the first four digits of the result from encrypting zeros using the Issuer Master Key. If option 88 is enabled, this field will contain six check digits.

`Key Exchange Key Check Digits`

Field 4, the first four digits of the result from encrypting zeros using the Key Exchange Key. If option 88 is enabled, this field will contain six check digits.

Table 8-14 Response 454: Generate ICC Master Key

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	3	454
1	E_{KEK}(ICC Master Key)	32	0 - 9, A - F
2	ICC Master Key Check Digits	4 or 6	0 - 9, A - F
3	Issuer Master Key Check Digits	4 or 6	0 - 9, A - F
4	Key Exchange Key Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- The Issuer Master Key must be encrypted under the MFK.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Master Key generated using Option A

- Clear-text Issuer Master Key: 0123 4567 89AB CDEF FEDC BA98 7654 3210.
  The Issuer Master Key in AKB format:
  1miNE000,B6E08D984234651F996C0EE50A6FC3A81883BF8F59043894,0DBCA34E
  8B39530E

- PAN: 3110 4999 9100 34

- Sequence Number: 01

- Clear-text Key Exchange Key: 0123 4567 89AB CDEF FEDC BA98 7654 3210
  The Key Exchange Key in AKB format:
  1KDEE000,00B12295FA353767B3E2F92C51F8C4799EE072B479C8C889,815074BF95
  E27120

The command looks like this:

```
<354#1miNE000,B6E08D984234651F996C0EE50A6FC3A81883BF8F59043894,0DBC
A34E8B39530E#31104999910034#01#1KDEE000,00B12295FA353767B3E2F92C51F
8C4799EE072B479C8C889,815074BF95E27120#>
```

The Network Security Processor returns the following response:

```
<454#18C70B43939B5C0C1EEFEF782AB4397B#4C63#08D7#08D7#>
```

Master Key generated using Option B

- Clear-text Issuer Master Key: 0123 4567 89AB CDEF FEDC BA98 7654 3210
  The Issuer Master Key in AKB format:
  1mENN000,ADFDEC0910108CE69B33772921599B2E95117733DA9FBD01,1939FAC7D
  EBB9B38

- PAN: B990123456789012345

- Sequence Number: 45

- Clear-text Key Exchange Key: FEDC BA98 7654 3210 0123456789012345.
  The Key Exchange Key in AKB format:
  1KDEE000,C5106B2421E3A50CCB4F4076DBE6857CD5786FA1D6083F83,C46B75779
  BD27DC6

The command looks like this:

```
<354#1mENN000,ADFDEC0910108CE69B33772921599B2E95117733DA9FBD01,1939
FAC7DEBB9B38#B990123456789012345#45#1KDEE000,C5106B2421E3A50CCB4F40
76DBE6857CD5786FA1D6083F83,C46B75779BD27DC6#>
```

The Network Security Processor returns the following response:

```
<454#7E741319CC3C7A2EB2C1A00E773F67D4#E985#08D7#7B83#>
```

Validate CAP Token (Command 356)

Command 356 supports both partial application cryptogram (AC) validation and transaction data signing (TDS). For partial AC validation, the Network Security Processor generates an EMV application cryptogram, selects a subset of the bits according to a supplied Issuer Proprietary Bitmap (IPB), and compares the selected bits to the partial AC. If transaction data signing is selected instead, the Network Security Processor generates the EMV AC, and then uses the AC as a key to single-DES CBC MAC the transaction data. The Network Security Processor then selects a subset of the bits from the MAC result according to the IPB and compares the result to the input partial MAC.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<356#EMV Derivation Type#Header,EMFK.E(IMK),MAC#Application PAN#
[PAN Sequence Number]#[Diversification Data]#Partial AC or MAC#
AC Padded Data Block#[Partial IPB]#[H#IV#Index#][TDS Data Block#]>
```

Response

```
<456#Verification Indicator#Session Key Check Digits#
Issuer Master Key Check Digits#>[CRLF]
```

Calling Parameters

`356`

Field 0, the command identifier.

`Derivation Type`

Field 1, the derivation type. This field contains a 1 byte decimal value defined as follows:

Value	Derivation Type
2	Common Session (per EMV 4.1 and Specification Update Bulletin 46)
3	VISA technique
8	EMV2000 (Tree-based technique)

Header, $E_{MFK.E}$(IMK),MAC

Field 2, the Issuer Master Key encrypted under the MFK. This field contains a 74 byte value. This key cant be either a 2key- or 3key-3DES key. If option 6C is enabled, it can be a 1key-3DES key. The following headers are supported: 1mENE000 and 1mENN000.

`Application PAN`

Field 3, the Application Primary Account Number (PAN). The PAN can contain a minimum of 13 decimal digits and a maximum of 19 decimal digits. This field is also used to indicate the Master Key derivation method. When this field contains only decimal digits, Master Key derivation option A will be used. When this field starts with the letter “B” and is followed by 17 to 19 decimal digits, Master Key derivation option B will be used.

`[PAN Sequence Number]`

Field 4, the optional application PAN sequence number. This field contains a 2 digit decimal value. If empty, a PAN Sequence Number of 00 will be used.

`[Diversification Data]`

Field 5, the value of this field depends on the derivation type specified in field 1.

For the common session derivation algorithm (if the derivation type, Field 1, is 2), this field contains a 16 byte hexadecimal value consisting of the following two items:

- 2 byte Application Transaction Counter (ATC). This binary value is expressed as 4 hexadecimal characters.
- 6 byte fixed value. This binary value “000000000000” is expressed as 12 hexadecimal characters.

For the EMV-Tree derivation algorithm (if the derivation type, Field 1, is 8), this field contains the four hexadecimal characters (2 bytes) of the Application Transaction Counter (ATC).

For the Visa derivation algorithm (if the derivation type, Field 1, is 3), this field must be empty.

For the Europay/MasterCard derivation algorithm (if the derivation type, Field 1, is 2), this field will contain either the same fields as the common session algorithm, or the four character ATC concatenated with 4 zero characters ‘0000’, followed by 4 bytes of hexadecimal characters (the unpredictable number).

`Partial AC or MAC`

Field 6, the value to be verified from the CAP token. This field must contain 4 to 16 hexadecimal characters; its length must be a multiple of 2. The value of this field depends on the content of field 12.

The effective IPB length (the number of 1-bits in the IPB) determines the maximum number of partial AC or MAC hexadecimal characters to supply in the command. The supplied partial AC or MAC must be right padded with zeros when total number of 1 bits of the IPB, divided by 4, is not an even number. For example, an IPB of FFFFF00000000000 has twenty bits that have a value of 1. Twenty divided by 4 is 5, which is not an even number, therefore the partial AC or MAC must be right padded with a zero so its length will be 6. If more than the maximum number of partial AC or MAC characters are provided in the command, field 1 of the response will be “N”, indicating that the token did not verify.

When the application is using CAP MODE 1 or MODE 2 without TDS, field 12 must be empty, and this field must contain the partial AC. When the default Partial IPB is used (field 8 is empty), the partial AC must be 4 hexadecimal characters.

When the application is using CAP MODE 2 with TDS, field 12 is not empty, and this field must contain a MAC value.

`AC Padded Data Block`

Field 7, the data to be MACed to generate the Application Cryptogram. The length of this field is 16 to 1024 hexadecimal characters. The length of this field must be a multiple of 16 characters.

It is the host application’s responsibility to collect all necessary data and format it for processing. The Network Security Processor does not uncompress the CAP token to recover any portion of this data.

`[Partial IPB]`

Field 8, the 8 bytes from the Issuer Proprietary Bitmap (IPB) that indicates which bits of the calculated AC should be compared to the value in field 6. The shaded area in the following table highlights the location of the appropriate bytes.

PSN	**CID**	**ATC**	**AC**	**IAD**
1 byte	1 byte	2 bytes	8 bytes	0 - 32 bytes

If this field is empty, a default value of FFFF0000 00000000 is used. The length of this field must be zero or 16 hexadecimal characters.

`[H#IV#Index#]`

These next three fields are present only if the derivation type is 8, EMV Tree Derivation.

`[H#`

Field 9, the height value used for EMV-Tree derivation. This field contains the value 8 or 16, or it can be empty. If this field is empty, the height value of 8 will be used.

`IV#`

Field 10, the clear Initialization Vector used for EMV-Tree derivation. This field contains a 32-byte hexadecimal value, or it can be empty. If this field is empty, an IV of 32-bytes of binary zeros (nulls) will be used.

`Index#]`

Field 11, the index value used for EMV-Tree derivation. This field contains a 1 digit decimal value between 0-7, or it can be empty. If this field is empty, the index value of 7 will be used.

The index specifies the byte location of the key that will be exclusive or'd with the ATC coefficient. An index value of zero indicates the leftmost byte of the key will be exclusive or'd with the ATC coefficient. An index value of 7 indicates the rightmost byte of the key will be exclusive or'd with the ATC coefficient. If the key is double-length, the index value is applied to both halves of the double length key.

`[TDS data block#]`

Field 12, if this field is present, the command will perform validation for CAP MODE 2 with TDS. The TDS data block consists of the transaction data that is to be MACed in this mode. The length of this field must be a multiple of 16 characters.

Table 8-15 Command 356: Validate CAP Token

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	356
1	Derivation Type	1	2, 3, 8
2	Header,$E_{MFK.E}$(IMK),MAC	74	printable ASCII
3	Application PAN	13 - 20	0 - 9, B
4	[PAN Sequence Number]	0, 2	0 - 9
5	[Diversification Data]	0, 4, 16	0 - 9, A - F
6	Partial AC or MAC	4 - 16	0 - 9, A - F
7	AC Padded Data Block	16 - 1024	0 - 9, A - F
8	[Partial IPB]	0, 16	0 - 9, A - F
9	[H#	0 - 2	8, 16
10	IV#	0, 32	0 - 9, A - F
11	Index#]	0, 1	0 - 7
12	[TDS Data Block#]	0 - 1024	0 - 9, A - F

Responding Parameters

`456`

Field 0, the response identifier.

`Verification Indicator`

Field 1, signifies success or failure of the AC or MAC verification. This field contains 1 byte character either 'Y' (verification pass) or 'N' (verification fail).

`Session Key Check Digits`

Field 2, the first four digits of the result from encrypting zeros using the generated session key. If option 88 is enabled, this field will contain six check digits.

`Issuer Master Key Check Digits`

Field 3, the first four digits of the result from encrypting zeros using the Issuer Master Key. If option 88 is enabled, this field will contain six check digits.

Table 8-16 Response 456: Validate CAP Token

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	3	456
1	Verification Indicator	1	Y or N
2	Session Key Check Digits	4 or 6	0 - 9, A - F
3	Issuer Master Key Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- The Issuer Master Key must be in AKB format.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Option A, CAP MODE 1 (no TDS Data Block field)

- EMV Derivation Type = 2 (EPI/MCI)
- Clear-text Issuer Master Key: 0123 4567 89AB CDEF FEDC BA98 7654 3210
  The Issuer Master Key in AKB format:
  1mENE000,8AB6708833051A7258342A74919328D5BCAADF60322ABF10,A05A674C696D25AA
- Application PAN = 9901234567890123
- Application PAN Sequence Number = 45
- [ATC or Random Number] = 1234567890123456
- partial AC or MAC = 9309
- Padded Data Block = 0123456789ABCDEF0123456789ABCDEF
- partial IPB = 8181818181818181

The command looks like this:

```
<356#2#1mENE000,8AB6708833051A7258342A74919328D5BCAADF60322ABF10,A0
5A674C696D25AA#9901234567890123#45#1234567890123456#9309#0123456789
ABCDEF0123456789ABCDEF#8181818181818181#>
```

The Network Security Processor returns the following response:

```
<456#Y#0995#08D7#>
```

Example 2: Option A, CAP MODE 2 (with TDS Data Block)

- EMV Derivation Type = 2 (EPI/MCI)
- Clear-text Issuer Master Key: 165441472D13CED3 CFC7CB6ADF63C31A
  The Issuer Master Key in AKB format:
  1mENE000,3C15DDE16CFADBC3DF8553B227E7CF91FC26485ECA973C27,D93DA94F
  E373ADDF
- Application PAN = 71372600550304
- Application PAN Sequence Number = 67
- [ATC or Random Number] = 6F1197963F72BBAD
- partial AC or MAC = D091
- Padded Data Block = DD3144D8C92138C5
- partial IPB = FFFF123400FFABCD
- TDS Data Block = DD3144D8C92138C5

The command looks like this:

```
<356#2#1mENE000,3C15DDE16CFADBC3DF8553B227E7CF91FC26485ECA973C27,D9
3DA94FE373ADDF#71372600550304#67#6F1197963F72BBAD#D091#DD3144D8C921
38C5#FFFF123400FFABCD#DD3144D8C92138C5#>
```

The Network Security Processor returns the following response:

```
<456#Y#7F90#2BD1#>
```

Verify Visa Cloud-Based Payments (Command 365)

Command 365 supports the Visa Cloud-Based Payments Contactless Specification, versions 1.3 and 1.5. It verifies contactless cryptograms calculated using the Visa defined Limited Use Keys. This command will generally be used for processing payments from devices without a secure element, such as Android™ devices using Host Card Emulation (HCE). It supports both Magnetic Stripe Data (MSD) transactions and quick Visa Smart Debit/Credit (qVSDC) chip transactions.

Transactions from contactless cards and mobile devices using a secure element should generally be processed using commands 350 or 357.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<365#Cryptogram Type#Header,EMFK.E(MDK),MAC#Application PAN#
[PAN Sequence Number]#Derivation Data#Cryptogram#Cryptogram Data#>
```

Response

```
<465#Verification Indicator#Session Key Check Digits#
Master Derivation Key Check Digits#>[CRLF]
```

Calling Parameters

`365`

Field 0, the command identifier.

`Cryptogram Type`

Field 1, the cryptogram type. This field can contain one of these values:

Value	Cryptogram Type
D	Decimal (MSD)
H	Hexadecimal (qVSDC - CVN 43)

Header,$E_{MFK.E}$(MDK),MAC

Field 2, the Master Derivation Key encrypted under the MFK. This field contains a 74 byte value. This key can be either a 2key- or 3key-3DES key. If option 6C is enabled, it can be a 1key-3DES key. The following headers are supported: 1mENE000 and 1mENN000.

`Application PAN`

Field 3, the Application Primary Account Number. This field contains a 1 - 19 digit decimal value.

`[PAN Sequence Number]`

Field 4, the optional application PAN sequence number. This field contains a 2 digit decimal value. If empty, a PAN Sequence Number of 00 will be used.

`Derivation Data`

Field 5, the data used to generate the Limited Use Key. This field contains a 16 digit decimal value. The derivation data is based on a data string of 1YHHHHCC, where:

- 1 is the digit 1.
- Y (0–9) is the least significant digit of the current year.
- HHHH (0001–8784) is the number of hours since start of January 1 of the current year (first hour of January 1 = 0001) expressed as digits.
- CC (00–99) is a counter that starts at 00 at the beginning of each hour and is incremented by 1 each time a Limited Use Key is generated.

The data string is right-padded with this value: 80000000.

The Network Security Processor will not add padding or check the validity of this field.

`Cryptogram`

Field 6, the cryptogram to be verified. When field 1 contains the letter “D”, this field must contain either 3 or 6 decimal digits. When field 1 contains the letter “H”, this field must contain 16 hexadecimal characters.

`Cryptogram Data`

Field 7, the data required to generate the cryptogram. The length of this field is 16 to 256 hexadecimal characters.

When field 1 contains the letter “D”, this field must contain 16 hexadecimal characters. For Visa Cloud-Based Payments version 1.3, this field should contain “0000000000000001”. For Visa Cloud-Based Payments version 1.5, this field should contain the rightmost (least significant) four digits of the current value of Application Transaction Counter (tag ‘9F36’) padded with either “000000000001” or “0000000000F1”. Refer to the appropriate Visa Cloud-Based Payments Contactless Specification for details.

When field 1 contains the letter “H”, the data should be formatted as defined in Visa Cloud-Based Payments Contactless Specification. It is the host application’s responsibility to pad the data. The Network Security Processor will not add padding or check the format.

Table 8-17 Command 365: Verify VISA Cloud-Based Payments

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	365
1	Cryptogram Type	1	D or H
2	Header,$E_{MFK.E}$(MDK),MAC	74	printable ASCII
3	Application PAN	1-19	0 - 9
4	[PAN Sequence Number]	0, 2	0 - 9
5	Derivation Data	16	0 - 9
6	Cryptogram	3 or 6, 16	0 - 9 or 0 - 9, A - F
7	Cryptogram Data	16 - 256	0 - 9, A - F

Responding Parameters

`465`

Field 0, the response identifier.

`Verification Indicator`

Field 1, signifies success or failure of the cryptogram verification. This field contains 1 character either ‘Y’ (successful verification) or ‘N’ (failed verification).

`Limited Use Key Check Digits`

Field 2, the first four digits of the result from encrypting zeros using the generated Limited Use Key. If option 88 is enabled, this field will contain six check digits.

`Master Derivation Key Check Digits`

Field 3, the first four digits of the result from encrypting zeros using the Master Derivation Key. If option 88 is enabled, this field will contain six check digits.

Table 8-18 Response 465: Verify Cloud-Based Payments

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	3	465
1	Verification Indicator	1	Y or N
2	Limited Use Key Check Digits	4 or 6	0 - 9, A - F
3	Master Derivation Key Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- The Master Derivation Key must be in AKB format.
- Refer to the Visa document, Visa Cloud-Based Payments, Contactless Specification, Version 1.3, 8 July 2014
- Refer to Visa Cloud-Based Payments Contactless Specification Visa Supplemental Requirements, Version 1.5, 20 March 2015. In version 1.50, support for version 1.5 of the Visa Cloud-Based Payments Contactless Specification has been added.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

MSD transaction

- Cryptogram Type = D
- Clear-text Master Derivation Key: 6161 6161 6161 6161 7070 7070 7070 7070
  The Master Derivation Key in AKB format:
  1mENE000,623CBC252210FA0D7D79A9136FAE58C1C45D748ED4C31830,CF33FC763DCB8026
- Application PAN = 4761739001010010
- Application PAN Sequence Number = 00
- Derivation Data = 1403450180000000
- Cryptogram = 987
- ATC = 4321
- Cryptogram Data = 4321000000000001

The command looks like this:

```
<365#D#1mENE000,623CBC252210FA0D7D79A9136FAE58C1C45D748ED4C31830,CF
33FC763DCB8026#4761739001010010#00#1403450180000000#987#43210000000
00001#>
```

The Network Security Processor returns the following response:

```
<465#Y#7375#6AE9#>
```

qVSDC transaction

- Cryptogram Type = H

- Clear-text Master Derivation Key: 6161 6161 6161 6161 7070 7070 7070 7070
  The Master Derivation Key in AKB format:
  1mENE000,623CBC252210FA0D7D79A9136FAE58C1C45D748ED4C31830,CF33FC763DCB8026
- Application PAN = 4761739001010010
- Application PAN Sequence Number = 00
- Derivation Data = 1403450180000000
- Cryptogram = 553437298C9AC22B
- Cryptogram Data =
  0000102030400000000000005553AABBCCDDEE5553091514AAF1E2D3C400401A2B0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF80000000000000

The command looks like this:

```
<365#H#1mENE000,623CBC252210FA0D7D79A9136FAE58C1C45D748ED4C31830,CF
33FC763DCB8026#4761739001010010#00#1403450180000000#553437298C9AC22
B#0000102030400000000000005553AABBCCDDEE5553091514AAF1E2D3C400401A2
B0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF80
000000000000#>
```

The Network Security Processor returns the following response:

```
<465#Y#7375#6AE9#>
```

EMV PIN change from IBM3624 offset (Command 3FC)

Command 3FC obtains the PIN from the offset, and then uses it to generate an EMV PIN change script message. Multiple key derivation types are supported.

To enable this command you must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<3FC#Derivation Type#PIN Verification Algorithm#
Header,E_MFK.E(IMK_ENC),MAC#Header,E_MFK.E(IMK_MAC),MAC#
[Header,E_MFK.E(IMK_AC),MAC]#Application PAN#
PAN Sequence Number#Diversification Data#[Application Data]#
Header,E_MFK.E(KPV),MAC#Conversion Table#Offset#
Validation Data#Pad Character#>
```

Response

```
<4FC#Encrypted PIN Block#MAC#KPV Check Digits#
IMK_ENC Check Digits#IMK_MAC Check Digits#[IMK_AC Check Digits]#
Session Encryption Key Check Digits#
Session MAC Key Check Digits>[CRLF]
```

Calling Parameters

`3FC`

Field 0, the command identifier.

`Derivation Type`

Field 1, the derivation type. This field contains a single digit decimal value that describes both the session key derivation method and the type of outgoing PIN block to generate. Valid values are defined as follows:

Value	Derivation Type
0	Common Session (per EMV Version 4.1 and Specification Update Bulletin 46) derivation with ISO format 2 PIN block.
1	VISA derivation technique with VISA PIN block.
6	Union Pay format using the session key for Message Confidentiality to encrypt the PIN block.
7	Union Pay format using the Issuer Master Key for Message Confidentiality (IMK_{ENC}) to encrypt the PIN block.
8	Common Core Definition PIN Block - Specification Update Bulletin 46 derivation with ISO format 2 PIN block with mandatory padding.

`PIN Verification Algorithm`

Field 2, the PIN Verification Algorithm. This field must contain the number 2, which specifies IBM 3624.

Header, $E_{MFK.E}(IMK_{ENC})$, MAC

Field 3, the Issuer Master Key for Message Confidentiality (IMK_{ENC}) encrypted under the MFK. The IMK_{ENC} can be either a 2key- or 3key-3DES key. If option 6C is enabled, it can be a 1key-3DES key. This field contains a 74 byte value. The following headers are supported: 1mENE00E or 1mENN00E.

Header, $E_{MFK.E}(IMK_{MAC})$, MAC

Field 4, the Issuer Master Key for Message Integrity (IMK_{MAC}) encrypted under the MFK. The IMK_{MAC} This key can be either a 2key- or 3key-3DES key. If option 6C is enabled, it can be a 1key-3DES key. This field contains a 74 byte value. The following headers are supported: 1mENE00M or 1mENN00M.

[Header, $E_{MFK.E}(IMK_{AC})$, MAC]

Field 5, the Issuer Master Key for Application Cryptogram generation (IMK_{AC}) encrypted under the MFK. When field 1 contains the number 0, 3, or 8, this field must be empty. When field 1 contains the number 1, this field must contain a 74 byte value. If present, the IMK_{AC} can be either a 2key- or 3key-3DES key, or if option 6C is enabled, it can be a 1key-3DES key. The following headers are supported: 1mENE000 or 1mENN000.

`Application PAN`

Field 6, the Application Primary Account Number (PAN). The PAN can contain a minimum of 13 decimal digits and a maximum of 19 decimal digits. This field is also used to indicate the Master Key derivation method. When this field contains only decimal digits, Master Key derivation option A will be used. When this field starts with the letter “B” and is followed by 17 to 19 decimal digits, Master Key derivation option B will be used.

`PAN Sequence Number`

Field 7, the application PAN sequence number. This field contains a 2 digit decimal value.

`Diversification Data`

Field 8, the value of this field depends on the derivation type specified in field 1.

For the common session derivation algorithm (if the derivation type, field 1, is 0 or 8) this field contains 16 hexadecimal characters which are the Application Transaction Counter (four hexadecimal characters) followed by 12 zeros.

For all other derivation types, this field contains the four hexadecimal characters (2 bytes) of the Application Transaction Counter.

`[Application Data]`

Field 9, the APP Data field may contain the 5-byte EMV command message header (CLA, INS, P1, P2, and Lc) followed by other optional items such as the Application Transaction Counter (ATC), or the Application Cryptogram (ARQC). If the optional ATC and ARQC are included in the calculation of the MAC, it is the application’s responsibility to prepend them in the Application data. The Application data will be concatenated with the encrypted PIN block (i.e., Application data || Encrypted PIN block) to form a script message to calculate the MAC. The content of Lc byte is not validated or manipulated by the Network Security Processor, it contains the appropriate value per EMV specification. It must be an even number of ASCII-hexadecimal characters. The maximum amount of data in this field is 3600 bytes.

`Header,`$\mathtt{E_{MFK.E}}$`(KPV),MAC`

Field 10, the PIN Verification Key (KPV) encrypted under the MFK. This field contains a 74 byte value. When option 4A is enabled, this key can be a 1key-3DES (single-length) key. The following headers are supported: 1V3NE000, 1V3NN000, 1V3VE000, and 1V3VN000.

`Conversion Table`

Field 11, a table that maps hexadecimal characters (0 through 9, A through F) to decimal digits (0 through 9). This field contains a 16 byte decimal value containing the clear-text Conversion Table or a volatile table location. When option 48 is enabled, this field contains the conversion table in AKB format with a header of either 1nCNE000 or 1nCNN000. When option 4E is enabled, all three forms of the conversion table (clear-text, decrypted, or value stored in volatile table location) must adhere to these rules:

- The conversion table must have at least eight unique digits.
- No single digit can occur more than four times.

`Offset`

Field 12, an offset value applied to the IBM 3624 algorithm-generated natural PIN to produce the PIN. This field contains a 4 to 12 byte decimal value. The PIN will be the same length as the offset. If the natural PIN will be used for the customer PIN, this field should contain a string of zeros equal to the PIN length. For example, for a 4-digit natural PIN this field would contain “0000”.

`Validation Data`

Field 13, validation data. This value is unique for each card holder and is typically the account number. This field contains a 4 to 16 byte hexadecimal value.

`Pad`

Field 14, the pad character which right-pads the validation data. This field contains a one byte hexadecimal value. The pad character is only applied when the validation data is less than 16 digits.

Table 8-19 Command 3FC: EMV PIN change from IBM3624 offset

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	3FC
1	Derivation Type	1	0, 1, 6, 7, 8
2	PIN Verification Algorithm	1	2
3	Header,$E_{MFK.E}(IMK_{ENC})$,MAC	74	printable ASCII
4	Header,$E_{MFK.E}(IMK_{MAC})$,MAC	74	printable ASCII
5	[Header,$E_{MFK.E}(IMK_{AC})$,MAC]	0, 74	printable ASCII
6	Application PAN	13-20	0 - 9, B
7	PAN Sequence Number	2	0 - 9
8	Diversification Data	4 or 16	0 - 9, A - F
9	[Application Data]	0, or 2 - 3600	0 - 9, A - F
10	Header,$E_{MFK.E}(KPV)$,MAC	74	printable ASCII
11	Conversion Table	16, 74	printable ASCII
12	Offset	4 - 12	0 - 9
13	Validation Data	4 - 16	0 - 9
14	Pad	1	0 - 9, A - F

Responding Parameters

`4FC`

Field 0, the response identifier.

`Encrypted PIN Block`

Field 1, the PIN block encrypted under the session key. When the derivation type is 0, this field contains 16 hexadecimal characters, for all other derivation types this field contains 32 hexadecimal characters.

When the derivation type is 0, the PIN block will be in the ISO Format-2 format.

When the derivation type is 1, the PIN block will be in the Visa format.

When the derivation type is either 6 or 7, the PIN block will be in UnionPay format.

When the derivation type is 8, the PIN block will be in ISO-2 format followed by a block of padding which consists of a byte of 0x80 followed by seven zero bytes. This two block structure will be 3DES-CBC encrypted using an all zero initialization vector.

`MAC`

Field 2, the Message Authentication Code of the issuer script message. This field will contain 16 hexadecimal characters.

`KPV Check Digits`

Field 3, the first six digits of the result from encrypting zeros using the PIN Verification Key.

IMK$_{ENC}$ Key Check Digits

Field 4, the first six digits of the result from encrypting zeros using the Issuer Master Key for Confidentiality.

IMK$_{MAC}$ Key Check Digits

Field 5, first six digits of the result from encrypting zeros using the Issuer Master Key for Integrity.

[IMK$_{AC}$ Key Check Digits]

Field 6, the first six digits of the result from encrypting zeros using the Issuer Master Key for Application Cryptogram. This field will be empty if field 5 of the command is empty.

`Session Encryption Key Check Digits`

Field 7, the first six digits of the result from encrypting zeros using the session encryption key.

`Session MAC Key Check Digits`

Field 8, the first six digits of the result from encrypting zeros using the session MAC key.

Table 8-20 Response 4FC: EMV PIN change from IBM3624 offset

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	3	4FC
1	Encrypted PIN Block	16 or 32	0 - 9, A - F
2	MAC	16	0 - 9, A - F
3	PIN Verification Key Check Digits	6	0 - 9, A - F
4	IMK$_{ENC}$ Check Digits	6	0 - 9, A - F
5	IMK$_{MAC}$ Check Digits	6	0 - 9, A - F

Table 8-20 Response 4FC: EMV PIN change from IBM3624 offset (continued)

Field	Contents	Length (bytes)	Legal Characters
6	[IMK$_{AC}$ Check Digits]	0, 6	0 - 9, A - F
7	SK$_{ENC}$ Check Digits	6	0 - 9, A - F
8	SK$_{MAC}$ Check Digits	6	0 - 9, A - F

Usage Notes

- When option 48 is enabled, field 11 may contain a volatile key table location.
- When the Network Security Processor's Master File Key is a 3key-3DES key (triple-length), the keys supplied in fields 3, 4, 5, and 10 may be 3key-3DES keys.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Derivation Type = 0
- Clear-text Issuer Master Key for Message Confidentiality: 0123456789ABCDEF FEDCBA9876543210, check digits = 08D7B4
  The Issuer Master Key for Message Confidentiality in AKB format:
  1mENE00E,99690A595620452A02411F2760ED8091F3A343CEDF6CBC2D,BF09FD42 93656DEE
- Clear-text Issuer Master Key for Message Integrity: 0123456789ABCDEF FEDCBA9876543210, check digits = 08D7B4
  The Issuer Master Key for Message Integrity in AKB format:
  1mENE00M,2FA8B705EFC04CA11F960B57669420B54CB2C892E258A650,C4AADD1 88A232674
- Application PAN = 1234567890123456
- Application PAN Sequence Number = 00
- Diversification Data = 0123456789ABCDEF
- Application Data = 0123456789ABCDEF
- Clear-text PIN Verification Key: FEDCBA9876543210 0123456789ABCDEF, check digits = 7B8358. The PIN Verification Key in AKB format:
  1V3NE000,6C47797C7E8F14AC52C11029B506B5A3B81BDD73217DF74A,AC8AFB30AF 0A69B9

- Conversion Table = 0123456789012345
- Offset = 7781
- Validation Data = 0123456789
- Pad = F

The command looks like this:

```
<3FC#0#2#1mENE00E,99690A595620452A02411F2760ED8091F3A343CEDF6CBC2D,
BF09FD4293656DEE#1mENE00M,2FA8B705EFC04CA11F960B57669420B54CB2C892E
258A650,C4AADD188A232674##1234567890123456#00#0123456789ABCDEF#0123
456789ABCDEF#1V3NE000,6C47797C7E8F14AC52C11029B506B5A3B81BDD73217DF
74A,AC8AFB30AF0A69B9#0123456789012345#7781#0123456789#F#>
```

The Network Security Processor returns the following response:

```
<4FC#E8CDAB8F8C411638#B6DDD8DED50D2DED#7B8358#08D7B4#08D7B4##383677
#383677#>
```

Authenticate multiple EMV scripts (Command 3FD)

Command 3FD generates a Message Authentication Code (MAC) using a derived EMV session key. Up to 15 script messages can be supplied in the command. Multiple key derivation types are supported.

To enable this command you must purchase this command in the form of a command 105, and then enable it in the Network Security Processor’s security policy.

Command

```
<3FD#Derivation Type#Header,$E_{MFK.E}$($IMK_{MAC}$),MAC#Application PAN#
PAN Sequence Number#Diversification Data#MAC Length#Script Message1#
[[[[[[[[[[[[[[Script Message2#]Script Message3#]Script Message4#]
Script Message5#]Script Message6#]Script Message7#]Script Message8#]
Script Message9#]Script Message10#]Script Message11#]
Script Message12#]Script Message13#]Script Message14#]
Script Message15#]>
```

Response

```
<4FD#MAC Length#Outgoing Counter#SK Check Digits#
$IMK_{MAC}$ Check Digits#MAC1#[[[[[[[[[[[[[[MAC2#]MAC3#]MAC4#]MAC5#]
MAC6#]MAC7#]MAC8#]MAC9#]MAC10#]MAC11#]MAC12#]MAC13#]MAC14#]MAC15#]>
[CRLF]
```

Calling Parameters

`3FD`

Field 0, the command identifier.

`Derivation Type`

Field 1, the session key derivation type. This field can contain one of these values:

Value	Description
0	Common session key derivation. The same MAC session key is used to generate a MAC for all script messages present in the command.
1	VISA derivation technique. The same MAC session key is used to generate a MAC all script messages present in the command.
2	Common session key derivation. The NSP will increment the diversification data for each script message supplied in the command. Therefore the MAC session key will be different for each script message.

`Header,`$E_{MFK.E}$`(`IMK_{MAC}`),MAC`

Field 2, the Issuer Master Key for Authentication (IMK_{MAC}) encrypted under the MFK.

This field contains a 74 byte value. This key can be either a 2key- or 3key-3DES key. If option 6C is enabled, it can be a 1key-3DES key. The following headers are supported: 1mENE00M and 1mENN00M.

`Application PAN`

Field 3, the Application Primary Account Number (PAN). The PAN can contain a minimum of 13 decimal digits and a maximum of 19 decimal digits. This field is also used to indicate the Master Key derivation method. When this field contains only decimal digits, Master Key derivation option A will be used. When this field starts with the letter “B” and is followed by 17 to 19 decimal digits, Master Key derivation option B will be used.

`PAN Sequence Number`

Field 4, the application PAN sequence number. This field contains a 2 digit decimal value.

`Diversification Data`

Field 5, the data which is encrypted with the IMK_{MAC} to generate the MAC session key.

When field 1 contains the value 0 or 2, this field contains 16 hexadecimal characters.

When field 1 contains the value 1, this field contains the four hexadecimal characters (2 bytes) of the Application Transaction Counter.

`MAC Length`

Field 6, a value that represents the length of the MAC returned in the response. This field contains a 1 digit decimal value.

Value	**MAC Length**
1	32 bits
2	48 bits
3	64 bits

All MAC lengths are expressed as groups of 4 hexadecimal characters separated by a space. For example, a 32-bit MAC consists of a group of 4 hexadecimal characters followed by a space which is followed by a group of 4 hexadecimal characters.

`Script1 Message`

Field 7, the script1 message to be MACed. This field must contain between 16 and 1024 hexadecimal characters. The length of the script message must be a multiple of 16.

`[Script2 Message#]`

Field 8, the script2 message to be MACed. This field is only present if there are 2 or more script messages to be MACed. If present, this field must contain between 16 and 1024 hexadecimal characters, and its length must be a multiple of 16.

`[Script3 Message#]`

Field 9, the script3 message to be MACed. This field is only present if there are 3 or more script messages to be MACed. If present, this field must contain between 16 and 1024 hexadecimal characters, and its length must be a multiple of 16.

`[Script4 Message#]`

Field 10, the script4 message to be MACed. This field is only present if there are 4 or more script messages to be MACed. If present, this field must contain between 16 and 1024 hexadecimal characters, and its length must be a multiple of 16.

`[Script5 Message#]`

Field 11, the script5 message to be MACed. This field is only present if there are 5 or more script messages to be MACed. If present, this field must contain between 16 and 1024 hexadecimal characters, and its length must be a multiple of 16.

`[Script6 Message#]`

Field 12, the script6 message to be MACed. This field is only present if there are 6 or more script messages to be MACed. If present, this field must contain between 16 and 1024 hexadecimal characters, and its length must be a multiple of 16.

`[Script7 Message#]`

Field 13, the script7 message to be MACed. This field is only present if there are 7 or more script messages to be MACed. If present, this field must contain between 16 and 1024 hexadecimal characters, and its length must be a multiple of 16.

`[Script8 Message#]`

Field 14, the script8 message to be MACed. This field is only present if there are 8 or more script messages to be MACed. If present, this field must contain between 16 and 1024 hexadecimal characters, and its length must be a multiple of 16.

`[Script9 Message#]`

Field 15, the script9 message to be MACed. This field is only present if there are 9 or more script messages to be MACed. If present, this field must contain between 16 and 1024 hexadecimal characters, and its length must be a multiple of 16.

`[Script10 Message#]`

Field 16, the script10 message to be MACed. This field is only present if there are 10 or more script messages to be MACed. If present, this field must contain between 16 and 1024 hexadecimal characters, and its length must be a multiple of 16.

`[Script11 Message#]`

Field 17, the script11 message to be MACed. This field is only present if there are 11 or more script messages to be MACed. If present, this field must contain between 16 and 1024 hexadecimal characters, and its length must be a multiple of 16.

`[Script12 Message#]`

Field 18, the script12 message to be MACed. This field is only present if there are 12 or more script messages to be MACed. If present, this field must contain between 16 and 1024 hexadecimal characters, and its length must be a multiple of 16.

`[Script13 Message#]`

Field 19, the script13 message to be MACed. This field is only present if there are 13 or more script messages to be MACed. If present, this field must contain between 16 and 1024 hexadecimal characters, and its length must be a multiple of 16.

`[Script14 Message#]`

Field 20, the script14 message to be MACed. This field is only present if there are 14 or more script messages to be MACed. If present, this field must contain between 16 and 1024 hexadecimal characters, and its length must be a multiple of 16.

`[Script15 Message#]`

Field 21, the script15 message to be MACed. This field is only present if there are 15 script messages to be MACed. If present, this field must contain between 16 and 1024 hexadecimal characters, and its length must be a multiple of 16.

Table 8-21 Command 3FD: Authenticate Multiple MasterCard EMV scripts

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	3FC
1	Derivation Type	1	0, 1, 2
2	Header,$E_{MFK.E}$(IMK$_{MAC}$),MAC	74	printable ASCII
3	Application PAN	13 - 20	0 - 9, B
4	PAN Sequence Number	2	0 - 9
5	Diversification Data	4, 16	0 - 9, A - F
6	MAC Length	1	1, 2, or 3
7	Script1 Message	16 - 1024	0 - 9, A - F
8	[Script2 Message#]	0, 16 - 1024	0 - 9, A - F
9	[Script3 Message#]	0, 16 - 1024	0 - 9, A - F
10	[Script4 Message#]	0, 16 - 1024	0 - 9, A - F

Table 8-21 Command 3FD: Authenticate Multiple MasterCard EMV scripts (continued)

Field	Contents	Length (bytes)	Legal Characters
11	[Script5 Message#]	0, 16 - 1024	0 - 9, A - F
12	[Script6 Message#]	0, 16 - 1024	0 - 9, A - F
13	[Script7 Message#]	0, 16 - 1024	0 - 9, A - F
14	[Script8 Message#]	0, 16 - 1024	0 - 9, A - F
15	[Script9 Message#]	0, 16 - 1024	0 - 9, A - F
16	[Script10 Message#]	0, 16 - 1024	0 - 9, A - F
17	[Script11 Message#]	0, 16 - 1024	0 - 9, A - F
18	[Script12 Message#]	0, 16 - 1024	0 - 9, A - F
19	[Script13 Message#]	0, 16 - 1024	0 - 9, A - F
20	[Script14 Message#]	0, 16 - 1024	0 - 9, A - F
21	[Script15 Message#]	0, 16 - 1024	0 - 9, A - F

Responding Parameters

`4FD`

Field 0, the response identifier.

`MAC Length`

Field 1, the length of the MAC returned in the response. This field contains the value supplied in field 6 of the command.

`Outgoing Counter`

Field 2, the diversification data for the next MAC calculation. When field 1 of the command (Derivation Type) contains the value 0 or 1, this field will contain the same value as supplied in field 5 (Diversification Data) of the command.

When field 1 of the command (Derivation Type) contains the value 2, this field will contain the diversification data for the next MAC to be generated. For example, if 3 script messages are supplied in the command, and the diversification data supplied in field 5 is 0000000000000001, this field will contain the outgoing counter value of 0000000000000004 which should be used to calculate the MAC for script message 4.

`SK Check Digits`

Field 3, the first four hexadecimal characters of the result from encrypting zeros using the generated session key. If option 88 is enabled, this field will contain six check digits.

$\texttt{IMK}_{\texttt{MAC}}$ `Check Digits`

Field 4, the first four hexadecimal characters of the result from encrypting zeros using the Issuer Master Key for Authentication (IMK_{MAC}). If option 88 is enabled, this field will contain six check digits.

`MAC1`

Field 5, the MAC of the script1 message. The length of this field is based on the value supplied in field 6 of the command. This field will contain 9, 14, or 19 characters.

`[MAC2#]`

Field 6, the MAC of the script2 message. This field is present only if field 8 of the command is present. The length of this field is based on the value supplied in field 6 of the command. If present, this field will contain 9, 14, or 19 characters.

`[MAC3#]`

Field 7, the MAC of the script3 message. This field is present only if field 9 of the command is present. The length of this field is based on the value supplied in field 6 of the command. If present, this field will contain 9, 14, or 19 characters.

`[MAC4#]`

Field 8, the MAC of the script4 message. This field is present only if field 10 of the command is present. The length of this field is based on the value supplied in field 6 of the command. If present, this field will contain 9, 14, or 19 characters.

`[MAC5#]`

Field 9, the MAC of the script5 message. This field is present only if field 11 of the command is present. The length of this field is based on the value supplied in field 6 of the command. If present, this field will contain 9, 14, or 19 characters.

`[MAC6#]`

Field 10, the MAC of the script6 message. This field is present only if field 12 of the command is present. The length of this field is based on the value supplied in field 6 of the command. If present, this field will contain 9, 14, or 19 characters.

`[MAC7#]`

Field 11, the MAC of the script7 message. This field is present only if field 13 of the command is present. The length of this field is based on the value supplied in field 6 of the command. If present, this field will contain 9, 14, or 19 characters.

`[MAC8#]`

Field 12, the MAC of the script8 message. This field is present only if field 14 of the command is present. The length of this field is based on the value supplied in field 6 of the command. If present, this field will contain 9, 14, or 19 characters.

`[MAC9#]`

Field 13, the MAC of the script9 message. This field is present only if field 15 of the command is present. The length of this field is based on the value supplied in field 6 of the command. If present, this field will contain 9, 14, or 19 characters.

`[MAC10#]`

Field 14, the MAC of the script10 message. This field is present only if field 16 of the command is present. The length of this field is based on the value supplied in field 6 of the command. If present, this field will contain 9, 14, or 19 characters.

`[MAC11#]`

Field 15, the MAC of the script11 message. This field is present only if field 17 of the command is present. The length of this field is based on the value supplied in field 6 of the command. If present, this field will contain 9, 14, or 19 characters.

`[MAC12#]`

Field 16, the MAC of the script12 message. This field is present only if field 18 of the command is present. The length of this field is based on the value supplied in field 6 of the command. If present, this field will contain 9, 14, or 19 characters.

`[MAC13#]`

Field 16, the MAC of the script13 message. This field is present only if field 19 of the command is present. The length of this field is based on the value supplied in field 6 of the command. If present, this field will contain 9, 14, or 19 characters.

`[MAC14#]`

Field 18, the MAC of the script14 message. This field is present only if field 20 of the command is present. The length of this field is based on the value supplied in field 6 of the command. If present, this field will contain 9, 14, or 19 characters.

`[MAC15#]`

Field 19, the MAC of the script15 message. This field is present only if field 21 of the command is present. The length of this field is based on the value supplied in field 6 of the command. If present, this field will contain 9, 14, or 19 characters.

Table 8-22 Response 4FD: Authenticate Multiple MasterCard EMV scripts

Field	Contents	Length (bytes)	Legal Characters
0	Response indicator	3	4FD
1	MAC Length	1	1, 2, 3
2	Outgoing Counter	4, 16	0 - 9, A - F
3	SK1 Check Digits	4 or 6	0 - 9, A - F

Table 8-22 Response 4FD: Authenticate Multiple MasterCard EMV scripts (continued)

Field	Contents	Length (bytes)	Legal Characters
4	IMK$_{MAC}$ Check Digits	4 or 6	0 - 9, A - F
5	MAC1	9, 14, or 19	0 - 9, A - F
6	[MAC2#]	0, 9, 14, or 19	0 - 9, A - F
7	[MAC3#]	0, 9, 14, or 19	0 - 9, A - F
8	[MAC4#]	0, 9, 14, or 19	0 - 9, A - F
9	[MAC5#]	0, 9, 14, or 19	0 - 9, A - F
10	[MAC6#]	0, 9, 14, or 19	0 - 9, A - F
11	[MAC7#]	0, 9, 14, or 19	0 - 9, A - F
12	[MAC8#]	0, 9, 14, or 19	0 - 9, A - F
13	[MAC9#]	0, 9, 14, or 19	0 - 9, A - F
14	[MAC10#]	0, 9, 14, or 19	0 - 9, A - F
15	[MAC11#]	0, 9, 14, or 19	0 - 9, A - F
16	[MAC12#]	0, 9, 14, or 19	0 - 9, A - F
17	[MAC13#]	0, 9, 14, or 19	0 - 9, A - F
18	[MAC14#]	0, 9, 14, or 19	0 - 9, A - F
19	[MAC15#]	0, 9, 14, or 19	0 - 9, A - F

Usage Notes

- When the Network Security Processor’s Master File Key is a 3key-3DES key (triple-length), the key supplied in field 2 may be a 3key-3DES key.
- For derivation types 0 and 2 the session key derivation is in accordance with SU-46, Replacement of EMV Session Key Derivation Method, October 2005.
- It is the host application’s responsibility to pad the script messages in fields 7 through 21 according to the EMV specification. The Network Security Processor will not enforce any data format; it only requires that the length of the data is a multiple of 16.
- Empty or blank script message fields are not allowed, only fields that contain script messages are allowed. For example, if 3 MACs are to generated fields 7, 8 and 9 must contain the script messages, and field 9 must be the last field of the command.
- The number of MACs returned in the response is equal to the number of script message fields supplied in the command. For example, if 3 script message fields are supplied in the command (fields 7 through 9), the response will contain 3 MACs, in fields 5 through 7, and field 7 will be the last field of the response.

- When the padded script message contains more than 1024 hexadecimal characters, use command 352 instead of this command to generate the MAC.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Derivation Type 0

- Clear-text Issuer Master Key for Message Integrity: 0123456789ABCDEF FEDCBA9876543210, check digits = 08D7B4
  The Issuer Master Key for Message Integrity in AKB format:
  1mENE00M,2FA8B705EFC04CA11F960B57669420B54CB2C892E258A650,C4AADD188A232674
- Application PAN = 1234567890123456
- Application PAN Sequence Number = 00
- Diversification Data = 0123456789ABCDEF
- MAC Length = 3 (64 bits)
- Script 1 Message = 0123456789ABCDEF
- Script 2 Message = FEDCBA9876543210
- Script 3 Message = 0000000000000000FEDCBA9876543210

The command looks like this:

```
<3FD#0#1mENE00M,2FA8B705EFC04CA11F960B57669420B54CB2C892E258A650,C4
AADD188A232674#1234567890123456#00#0123456789ABCDEF#3#0123456789ABC
DEF#FEDCBA9876543210#0000000000000000FEDCBA9876543210#>
```

The Network Security Processor returns the following response:

```
<4FD#3#0123456789ABCDEF#3836#08D7#DF0D 4B9D F532 F079#CF7E 0FD3 2237
C1A4#55FC 4727 D51D 1C5D#>
```

Derivation Type 2

- Clear-text Issuer Master Key for Message Integrity: 0123456789ABCDEF FEDCBA9876543210, check digits = 08D7B4
  The Issuer Master Key for Message Integrity in AKB format:
  1mENE00M,2FA8B705EFC04CA11F960B57669420B54CB2C892E258A650,C4AADD188A232674
- Application PAN = 1234567890123456

- Application PAN Sequence Number = 00
- Diversification Data = 0123456789ABCDEF
- MAC Length = 3 (64 bits)
- Script 1 Message = 0123456789ABCDEF
- Script 2 Message = FEDCBA9876543210
- Script 3 Message = 0000000000000000FEDCBA9876543210

The command looks like this:

```
<3FD#2#1mENE00M,2FA8B705EFC04CA11F960B57669420B54CB2C892E258A650,C4
AADD188A232674#1234567890123456#00#0123456789ABCDEF#3#0123456789ABC
DEF#FEDCBA9876543210#0000000000000000FEDCBA9876543210#>
```

The Network Security Processor returns the following response:

```
<4FD#3#0123456789ABCDF2#3836#08D7#DF0D 4B9D F532 F079#275B EAA8 A200
83C1#58E1 3E45 BCC8 E53B#>
```

9

Storing Values in the Volatile Table

The volatile table is an area of Network Security Processor memory where you can temporarily store 3DES working keys, conversion tables, and Diebold Number Tables. This section describes the volatile table commands. A separate RSA key table exists for storage of RSA keys, see commands 120, 121 and 12C for more information on how to store RSA keys.

To skip this introduction, go to Table 9-1 on page 9-3 for a list of commands.

About the Volatile Table

The volatile table memory is erased when the Network Security Processor experiences a power outage, is reset to factory state, or when the Master File Key is promoted via command 9F.

The volatile table consists of two sections. Section 0, the default section, which is used to store DES keys and conversion tables; this section can hold 1,000 values. Commands 70 and 7F are used to load DES keys and conversion tables in section 0. Section 1 is used to store rows of Diebold Number tables; this section can hold 10,000 rows. The Network Security Processor uses values in section 1 only when processing PIN verification and PIN change commands that support the Diebold Number Table. Command 74 is used to load rows of Diebold Number Tables in section1. Commands 71, 72, and 73 have an optional parameter [Table Section#] which is used to specify which section of the table the command affects.

Referencing a location

Instead of providing the 74-byte Atalla Key block in a command, the host application specifies the location in the volatile table were the value is stored. The location is specified in the following manner:

```
Tn
```

where *n* is the location where the value has been stored. The table section number is not provided in the command.

For example, assume that volatile table location 75 contains a Key Exchange Key. The syntax for command 10 to generate a 2key-3DES PIN Encryption Key would be as follows:

```
<10#1PUNE000#T75#>
```

Volatile Table Tasks

Using the volatile table typically involves the following tasks:

- Loading values into the table.
- Verifying the existence of values within the table.
- Deleting values when they are no longer needed.

Loading the Volatile Table

The volatile table requires all values to be in AKB format. To load a DES key, conversion table, or a row of a Diebold Number Table into the volatile table, you must first generate the appropriate Atalla Key Block. You then include the Atalla Key Block in one of the commands used to load the volatile table. See commands 70, 74, or 7F.

Keep a record of the Atalla Key Blocks you store in the volatile table. Reconstructing the table can be difficult unless you have a record of its contents.

Verifying Values in the Volatile Table

When the Network Security Processor loses power, is reset to factory state, or the Master File Key is promoted via command 9F, the volatile table is erased. With this in mind, you should periodically verify the table, using command 72, to be sure that it contains the correct values.

Deleting Values from the Volatile Table

On occasion you may need to erase values that are no longer being used. Use command 73 to delete all values in a section of the volatile table or use command 71 to delete a value from a single location. There is no command to delete Diebold Number Tables. Command 74, which is used to load a row of the Diebold Number Table, will overwrite any value in the specified location.

Volatile Table Commands

Quick Reference

Table 9-1 on page 9-3 identifies each command by number, name, and purpose. Commands are listed in numerical order. All commands are enabled in the Network Security Processor's default security policy.

Table 9-1 Volatile Table Commands

Command	Name	Purpose
70	Load Volatile Table Value	Loads a DES key or conversion table into the next available location in the table.
71	Delete Volatile Table Value	Deletes a value stored in a specific location.
72	Verify Volatile Table Value	Retrieves the check digits of a value stored in a specific location.
73	Clear Volatile Table	Clears a section of the volatile table.
74	Load Diebold Number Table Row	Loads a row of the Diebold number table.
7F	Load Value to a specific location	Loads a DES key or conversion table into a specified location.

Load Volatile Table Value (Command 70)

Command 70 is used to load either a DES key or conversion table into the first available location of the volatile table. Alternately you can use command 7F to load a DES key or conversion table into a specific location.

This command can accept a 1key-3DES (single-length) key only when option 6C is enabled. When loading a conversion table, the Network Security Processor does not require option 6C to be enabled. The Network Security Processor does not check that the clear-text value of a conversion table contains all numeric digits.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<70#Reserved#Header,EMFK.E(WK),MAC#>
```

Response

```
<80#Location#Remaining Locations#Check Digits#>[CRLF]
```

Calling Parameters

`70`

Field 0, the command identifier.

`Reserved`

Field 1, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

`Header,`$E_{MFK.E}$`(WK),MAC`

Field 2, the AKB of the DES working key or conversion table being loaded. This field contains a 74 byte value. The DES working key can have any valid header. See Working Key Headers for a partial list of supported headers. A conversion table must have a header of 1nCNE000.

Table 9-2 Command 70: Load Volatile Table Value

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	70
1	Reserved	0, 1, 2	0 - 9
2	Header,$E_{MFK.E}$(WK),MAC	74	printable ASCII

Responding Parameters

`80`

Field 0, the response identifier.

`Location`

Field 1, the location where the DES working key or conversion table is stored. This field contains a number in the range of 0000 through 0999.

`Remaining Locations`

Field 2, the number of locations available after this command is executed. This field contains a number in the range of 0000 through 0999.

`Check Digits`

Field 3, check digits; the first four digits that result from encrypting zeros using the DES working key or conversion table. When option 88 is enabled, this field will contain six check digits.

Table 9-3 Response 80: Load Volatile Table Value

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	80
1	Location	4	0 - 9
2	Remaining locations	4	0 - 9
3	Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- Before using Command 70, generate the AKB of the DES working key or conversion table to be loaded into the volatile table.
- Do not use this command to store HMAC-SHA1 or HMAC-SHA256 keys.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Load a DES working key into the volatile table

Clear-text PIN Encryption Key: 3333 3333 3333 3333, check digits = ADC6
The PIN Encrytion Key in AKB format:
1PUNE000,648CC44390726074BA90F7D2EB760D1D3BCDE456C1FA7915,4B8EF70DF76
A5C9D

The command looks like this:

```
<70##1PUNE000,648CC44390726074BA90F7D2EB760D1D3BCDE456C1FA7915,4B8E
F70DF76A5C9D#>
```

The Network Security Processor returns a response similar to this:

```
<80#0000#0999#ADC6#>
```

Delete Volatile Table Value (Command 71)

Command 71 deletes a value stored in a specific location within a section of the volatile table. This command is enabled in the Network Security Processor's default security policy.

Command

```
<71#Location#[Table Section#]>
```

Response

```
<81#Available Locations#>[CRLF]
```

Calling Parameters

`71`

Field 0, the command identifier.

`Location`

Field 1, the location to be deleted. When deleting a value from section 0, this field contains a number in the range of 0000 through 0999. When deleting a row of a Diebold Number Table from section 1, this field contains a number in the range of 0000 through 9999.

`[Table Section#]`

Field 2, optional, the section of the table. When this field is not present, the table section will be section 0.

Table Section Number	Content
0 (default)	DES Keys and Conversion Tables
1	Rows of Diebold Number Tables

Table 9-4 Command 71: Delete Value from Volatile Table

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	71
1	Location	1 - 4	0 - 9
2	[Table Section#]	0 or 1	0, 1, or empty

Responding Parameters

`81`

Field 0, the response identifier.

`Remaining Available Locations`

Field 1, the number of locations available after the command has been executed. When field 2 of the command contains the value 0, this field will contain a number in the range of 0000 through 0999. When field 2 of the command contains the value 1, this field will contain a number in the range of 0000 through 9999.

Table 9-5 Response 81: Delete Volatile Table Value

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	81
1	Remaining Available Locations	4	0 - 9

Examples

The following example illustrates deleting a value stored in location 35 of section 0.

The command looks like this:

```
<71#35#>
```

The Network Security Processor returns a response similar to this:

```
<81#0999#>
```

The following example illustrates deleting a value stored in location 437 of section 1.

The command looks like this:

```
<71#437#1#>
```

The Network Security Processor returns a response similar to this:

```
<81#2743#>
```

Verify Volatile Table Value (Command 72)

Command 72 retrieves the check digits for the value stored in a specific location within a section of the volatile table. This command is enabled in the Network Security Processor's default security policy.

Command

```
<72#Location#[Table Section#]>
```

Response

```
<82#Available Locations#Check Digits#>[CRLF]
```

Calling Parameters

`72`

Field 0, the command identifier.

`Location`

Field 1, the location that contains the value you want to verify. To verify a value from section 0, this field can contain a number in the range of 0000 through 0999. To verify a row of a Diebold Number Table from section 1, this field can contain a number in the range of 0000 through 9999.

`[Table Section#]`

Field 2, optional, the section of the table. When this field is not present, the table section will be section 0.

Table Section Number	Content
0 (default)	DES Keys and Conversion Tables
1	Rows of Diebold Number Tables

Table 9-6 Command 72: Verify Volatile Table Value

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	72
1	Location	1 - 4	0 - 9
2	[Table Section#]	0 or 1	0, 1, or empty

Responding Parameters

```
82
```

Field 0, the response identifier.

```
Available Locations
```

Field 1, the number of locations available after the command has been executed. When field 2 of the command contains the value 0, this field will contain a number in the range of 0000 through 0999. When field 2 of the command contains the value 1, this field will contain a number in the range of 0000 through 9999.

```
Check Digits
```

Field 2, the check digits for the value stored in the specified location. Check digits are the first 4 digits that result from encrypting zeros using the value. When option 88 is enabled, this field will contain the first six digits of the result from encrypting zeros using the value.

Table 9-7 Response 82: Verify Volatile Table Value

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	82
1	Available Locations	4	0 - 9
2	Check Digits	4 or 6	0 - 9, A - F

Usage Notes

The Network Security Processor returns an error message <00#0701...> when the specified location is empty.

Examples

Verify a DES key in location 17

The command looks like this:

```
<72#17#>
```

The Network Security Processor returns a response similar to this:

```
<82#1950#D5D4#>
```

Verify a row of a Diebold Number Table in location 438

The command looks like this:

```
<72#438#1#>
```

The Network Security Processor returns a response similar to this:

```
<82#9950#D5D4#>
```

Clear Section of Volatile Table (Command 73)

Command 73 clears a section of the volatile table. This command is enabled in the Network Security Processor's default security policy.

warning Before you send this command, make sure no other host application is using the volatile table.

Command

```
<73#[Table Section#]>
```

Response

```
<83#OK#>[CRLF]
```

Calling Parameters

`73`

Field 0, the command identifier.

`[Table Section#]`

Field 1, optional, the section of the table. When this field is not present, the table section will be section 0.

Table Section Number	Content
0 (default)	DES Keys and Conversion Tables
1	Rows of Diebold Number Tables

Table 9-8 Command 73: Clear Section of Volatile Table

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	73
1	[Table Section#]	0, 1	0, 1 or empty

Responding Parameters

`83`

Field 0, the response identifier.

```
OK
```

Field 1, the indicator that the command has been executed.

Table 9-9 Response 83: Clear Section of Volatile Table

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	83
1	OK	2	OK

Usage Notes

Use Command 73 prior to loading the volatile table to ensure that values are loaded into the correct locations.

Examples

The following example illustrates clearing section 0 of the volatile table.

The command looks like this:

```
<73#>
```

The Network Security Processor returns the following response:

```
<83#OK#>
```

The following example illustrates clearing section 1 of the volatile table.

The command looks like this:

```
<73#1#>
```

The Network Security Processor returns the following response:

```
<83#OK#>
```

Load Diebold Number Table (Command 74)

Command 74 loads one row of a Diebold Number Table into the volatile table. A Diebold Number Table contains 512 characters. It is organized as 32 rows, each of which is 16 hexadecimal characters. To load a Diebold Number Table, you must load each row using a separate command 74, and specify consecutive volatile table locations. This command is enabled in the Network Security Processor's default security policy.

Command

```
<74#Location#Reserved#Header,E_MFK.E(DNT Row),MAC#>
```

Response

```
<84#Location#>[CRLF]
```

Calling Parameters

`74`

Field 0, the command identifier.

`Location`

Field 1, the location where the row will be stored. The specified location will be loaded regardless of its current contents. This field contains a number between 0000 and 9999.

`Reserved`

Field 2, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

Header,$E_{MFK.E}$(DNT Row),MAC

Field 3, one row of the Diebold Number Table encrypted under the MFK. This field contains a 74 byte value. The Atalla Key Block that contains a row of the Diebold Table must have one of the following headers: 1ndNE000 and 1ndNN000.

Table 9-10 Command 74: Load Diebold Number Table

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	74
1	Location	1 - 4	0 - 9
2	Reserved	0, 1, 2	0 - 9
3	Header,$E_{MFK.E}$(DNT Row),MAC	74	printable ASCI

Responding Parameters

`84`

Field 0, the response indicator.

`Location`

Field 1, the location where the row of the Diebold Number Table has been stored. This field contains a number between 0000 and 9999.

Table 9-11 Response 84: Load Diebold Number Table

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	84
1	Location	4	0 - 9

Usage Notes

The rows of the Diebold Number Table are required to be in AKB format. The rows must be loaded into consecutive locations.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- Volatile Table Location number: 350
- Clear-text row of the Diebold Number Table: C860 2A41 4D38 2C5B
  The row of the Diebold Number Table in AKB format:
  1ndNE000,87A94104D5C5452C50838D4244ADE06B22864B12E6BCD638,3D6565A1AD6B7279

The command looks like this:

```
<74#350##1ndNE000,87A94104D5C5452C50838D4244ADE06B22864B12E6BCD638,
3D6565A1AD6B7279#>
```

The Network Security Processor returns the following response:

```
<84#0350#>
```

Load Value to a Specific Volatile Table Location (Command 7F)

Command 7F is used to load either a DES key or conversion table into an empty specified location in the volatile table. When the location referenced in the command already contains a DES key or conversion table, an error message <00#0603xx#> is returned.

This command can accept a 1key-3DES (single-length) key only when option 6C is enabled. When loading a conversion table, the Network Security Processor does not require option 6C to be enabled. The Network Security Processor does not check that the clear-text value of a conversion table contains all numeric digits.

This command is enabled in the Network Security Processor's default security policy.

Command

```
<7F#Reserved#Header,EMFK.E(WK),MAC#Location#>
```

Response

```
<8F#Location#Available Locations#Check Digits#>[CRLF]
```

Calling Parameters

`7F`

Field 0, the command identifier.

`Reserved`

Field 1, this field is reserved for future use, any value in this field will be ignored by the Network Security Processor. This field can be 0, 1, or 2 bytes long.

`Header,E`$_{MFK.E}$`(WK),MAC`

Field 2, the DES working key or conversion table in AKB format. This field contains a 74 byte value. The DES working key can have any valid header. See Working Key Headers for a partial list of supported headers. A conversion table must have a header of 1nCNE000.

`Location`

Field 3, the location where the value will be stored, this key location must be empty. This field contains a number between 0000 and 0999.

Table 9-12 Command 7F: Load Value to a Specific Volatile Table Location

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	7F
1	Reserved	0, 1, 2	printable ASCII
2	Header,$E_{MFK.E}$(WK),MAC	74	printable ASCII
3	Location	1 - 4	0 - 9

Responding Parameters

`8F`

Field 0, the response identifier.

`Location`

Field 1, the location where the value has been stored. This field contains a number in the range of 0000 through 0999.

`Available Locations`

Field 2, the number of locations available to store values. This field contains a number in the range of 0000 through 0999.

`Check Digits`

Field 3, the check digits for the value stored in the specified location. Check digits are the first 4 digits that result from encrypting zeros using the value. When option 88 is enabled, this field will contain six check digits.

Table 9-13 Response 8F: Load Value to a Specific Volatile Table Location

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	8F
1	Location	4	0 - 9
2	Available Locations	4	0 - 9
3	Check Digits	4 or 6	0 - 9, A - F

Usage Notes

- Generate the AKB for the value to be loaded into the volatile table.
- Use command 72 to make sure the location does not already contain a key or conversion table.

- Do not use this command to store HMAC-SHA1 or HMAC-SHA256 keys.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Load a DES key into a specific location of the volatile table

- Clear-text DES working key: 3333 3333 3333 3333, check digits = ADC6
  The working key in AKB format:
  1PUNE000,648CC44390726074BA90F7D2EB760D1D3BCDE456C1FA7915,4B8EF70D
  F76A5C9D

- Location: 0008

The command looks like this:

```
<7F##1PUNE000,648CC44390726074BA90F7D2EB760D1D3BCDE456C1FA7915,4B8E
F70DF76A5C9D#0008#>
```

The Network Security Processor returns a response similar to this:

```
<8F#0008#0999#ADC6#>
```

10

Using Public Key Technology to Initialize Remote Devices

This section contains information on the commands that utilize RSA keys for remote key management. These commands generate RSA key pairs, import public keys, generate message digests, encrypt an ATM Master key using the public key stored in the EPP, and generate and verify digital signatures.

All data must be sent as ASCII-hexadecimal characters. Any vendor specific data encoding, such as the NDC base-94 coding, must be converted to ASCII-hexadecimal characters by the host application.

To skip this introduction, go to Table 10-2 on page 10-7 for a list of commands.

Atalla Key Block (AKB) formats for RSA Keys

DES and RSA keys must be in AKB format. Using DES Keys provides details about the AKB format for DES keys. The AKB formats for the RSA keys are described in this section.

The RSA public key AKB (AKB$_{PUB}$) consists of three components which are comma delimited.

- Header (8 ASCII characters)
- Public key data (clear-text ASCII-hexadecimal characters, variable length).
  The format is:

```
public exponent length|public exponent|modulus length|modulus
```

note length is the number of bytes expressed as a hexadecimal value. For example, the length of a 2048-bit modulus is 0100 (256 bytes = 2048 bits).

- Message Authentication Code (16 ASCII-hexadecimal characters)

The RSA private key AKB (AKB$_{PRV}$) consists of three components which are comma delimited.

- Header (8 ASCII characters)

- Encrypted private key data. (ASCII-hexadecimal characters, variable length)
  The format of the cleartext key data is:

  ```
  number of primes|length of modulus|modulus|
  length of public exponent|public exponent|
  length of private exponent|private exponent|
  length of prime1|prime1|length of prime2|prime2|
  length of prime1 exponent|prime1 exponent|
  length of prime2 exponent|prime2 exponent|
  length of coefficient|coefficient|zero padding if necessary
  ```

  note length is the number of bytes expressed as a hexadecimal value. For example, the length of a 2048-bit modulus is 0100 (256 bytes = 2048 bits).

- Message Authentication Code (16 ASCII-hexadecimal characters)

AKB Headers For RSA Keys

Table 10-1 AKB Headers for RSA Keys

Header	Description
1KREE000	The RSA public key used to encrypt key information.
1KRDE000	The RSA private key used to decrypt key information.
1SRGE000	The RSA private key used to generate a digital signature.
1SRVE000	The RSA public key used to verify a digital signature.
1kRDE000	The RSA private key used to decrypt a TR-34 Key Block.
1kREE000	The RSA public key used to encrypt a TR-34 Key Block.
1pRGE000	The RSA private key.
1wRGE000	The RSA private key used to generate a TR-34 digital signature.
1wRVE000	The RSA public key used to verify a TR-34 digital signature.

MAC of an RSA AKB

The Message Authentication Code, calculated over the clear header and the clear public key or the encrypted private key, ensures that these values have not been altered. When generating the MAC, if the data length is not a multiple of 8 it is right-padded with binary zeros.

The MAC algorithm is 3DES-CBC, the initialization vector is binary zeros. The MAC key is a variant of the Master File Key. The size of MAC is 8 bytes.

Check digits

RSA key block check digits are the leftmost 8 hexadecimal characters of the SHA-1 message digest of the public or private key data. Check digits can be used to confirm that the correct value of the public or private key is stored in the RSA key table.

Importing RSA Keys

The Network Security Processor will only process RSA keys which are in AKB format. The purpose of the AKB is to validate trust in the key, and also to define how it can be used. For RSA public keys generated externally from the Network Security Processor, trust is established via a digital certificate. The digital certificate contains a signature created by a trusted Certificate Authority (CA), that validates the value and usage of the public key. Not all certificates are in X.509 format, the Network Security Processor supports X.509 certificates as well as proprietary format certificates which are used by some ATM vendors.

The first step in the process is to establish trust in the CA's public key. This step requires dual control authorization to ensure that only the correct and truly trusted CA public key, appropriate for the application, will be used to validate a device's digital certificate.

note CA's used for other applications, such as creating SSL certificates, should never be used to verify a TR-34 keying protocol message. Using the incorrect key as a CA public key allows an attacker to compromise all exportable keys.

Use command 12A to create the AKB for a CA's public key. (Command 122 can be used for legacy purposes but is deprecated for new applications.) Command 12A does not require a self-signed certificate for the CA because not all older financial applications used a self-signed root CA.

The final step is to use Command 123 to create an AKB for device's public key. Inputs to command 123 include the device's public key, the certificate data used to create the signature, the signature, and the CA's public key in AKB format. Command 123 verifies that the device's digital certificate was signed by the CA's public key, and then creates an AKB for the device's public key. The device's public key, in AKB format, can now be used for encryption or signature operations.

RSA private keys must be generated by the Network Security Processor, they cannot be imported.

Command Usage Scenarios for NCR devices

The NCR trusted center serves as a Certificate Authority that signs the public keys of the Network Security Processor and the EPP. The host application will use the Network Security Processor in the following manner:

1) Generate RSA key pair for the host application

Generate RSA Key Pair (Command 120)

Input: Key Usage = S, public exponent, size of modulus
Output: AKB_{PUB}-HostApp, AKB_{PRV}-HostApp, and check digits.

2) Import NCR public key

Generate Message Digest (Command 126)

Input: public exponent | modulus
Output: SHA1 hash

Generate AKB for Root Public Key (Command 122)

Input: NCR Root public key, $E_{MFK.30}$(Hash-16)
Output: AKB_{PUB}-NCR.

3) Verify the serial number of the EPP signed by the NCR root key

Verify Digital Signature (Command 125)

Input: algo, serial number, digital signature, AKB_{PUB}-NCR
Output: Y or N

4) Import public key signed by NCR root key

Verify Public Key and Generate AKBPUB (Command 123)

Input: PK-EPP, Key Usage = K, digital signature, AKB_{PUB}-NCR
Output: AKB_{PUB}-EPP

5) Generate ATM Master Key and encrypt it under EPP public key

Generate ATM Master Key and encrypt with public key (Command 12F)

Input: Key Length, Key Block data length = 0, AKB_{PUB}-EPP
Output: AKB_{MK}, $E_{RSA\text{-}PUB\text{-}EPP}$(MK)

6) Generate digital signature using the host application's private key

Generate Digital Signature (Command 124)

Input: algo, data to be signed, AKB_{PRV}-HostApp
Output: Digital signature

Command Usage Scenarios for Diebold devices

The host application must send a key, KKTK, to the ATM. In the key transport protocol, the ATM will initiate the process by sending randomly generated data to the host. The host application will use the Network Security Processor in the following manner:

1) Generate RSA key pair for the host application

Generate RSA Key Pair (Command 120)

Input: Key Usage = S, public exponent, size of modulus
Output: AKB_{PUB}-HostApp, AKB_{PRV}-HostApp, and check digits

2) Import Diebold public key

Generate Message Digest (Command 126)

Input: public exponent | modulus
Output: SHA1(PK-DBD)

Generate AKB for Root Public Key (Command 122)

Input: Diebold public key, $E_{MFK.30}$(Hash-16)
Output: AKB_{PUB}-DBD

3) Import public key signed by Diebold root key

Verify Public Key and Generate AKBPUB (Command 123)

Input: PK-EPP, Key Usage, digital signature, AKB_{PUB}-DBD
Output: AKB_{PUB}-EPP

4) Generate Master Key and encrypt it under EPP public key

Generate ATM Master Key and encrypt with public key (Command 12F)

Input: Key Length, Key Block data = (IHOST || key identifier), AKB_{PUB}-EPP
Output: AKB_{KTK}, $E_{RSA\text{-}PUB\text{-}EPP}$(Key Block data & KTK)

5) Generate digital signature using the host application's private key

Generate Digital Signature (Command 124)

Input: algo, data to be signed, AKB_{PRV}-HostApp
Output: Digital signature

6) Verify the signature of data signed by the EPP private key

Verify Digital Signature (Command 125)

Input: algo, data or (rHOST | rATM | lHOST), digital signature,
AKB$_{PUB}$-EPP

Output: Y or N

Command Usage Scenario for sending TR-34 messages

Initial setup per Certificate Authority (CA)

1. Use command 12A to get the CA's public key into AKB format with a header of 1RRVN00k. This AKB is used as field 8 in command 123. The CA's public key is required to verify the recipient's certificate (which contains the recipient's public key).

2. Use command 120 to generate a public and private key pair. Field 1 of the command 120 will contain the letter "w". The private key will have a header of 1wRGE000, for use as field 9 in command 139. The public key will have a header of 1wRVE000 (however this AKB is not used as the NSP does not supports receiving TR-34 key blocks). The public key must be provided to the recipient so they can use it to verify the message signature created by command 139.

3. Use the public key information from the step 2 to create a KDH certificate request message. This step is performed by the host application.

4. Sign the certificate request message using command 139.

5. Receive the KDH certificate from the CA and store it on the host for use in the TR34 protocol.

note The Certificate can be verified with command 123 or by using the host application. (The NSP does not need to use this certificate since the key was generated by the NSP, but terminal will need to use this certificate.)

TR-34 Bind Phase

6. Receive CRD Credential Token (terminal certificate). The host application parses the certificate to find the modulus and public key. Use command 123 to get the recipient's public key into AKB format, the header is 1kREE000. This AKB is used as field 7 in command 136.

7. The host application sends the KDH certificate (from step 5) and current CRL to the terminal.

Symmetric key transport Phase

8. The host application optionally receives Random Number Token and saves for inclusion in step 10.

9. Use command 136 to:

   - Generate a key in AKB format (TR-34 Step B2)
   - Generate an ephemeral key (TR-34 Step B3)
   - Generate an IV
   - Use the ephemeral key and IV to 3DES-CBC encrypt the key block (version, IDKDH, Kn, KBH). In other words, encipher the BE data supplied as field 4, the key generated in the first bullet and the header supplied as field 2. (TR-34 Step B4)
   - Encrypt the ephemeral key under the recipient's public key (TR-34 Step B5)
   - Generate an AKB of the signing key token with a header of 1bRVN000, for use as field 8 in command 139.

10. The host application builds the data to sign for the key token, including optional random number from step 8, the envelop data, ephemeral key encrypted under the recipient's public key, IV, and encrypted BE value from step 9, timestamps, etc.

11. Use command 139 to hash and sign the to-be-signed data from step 10 using the private key created in step 2 above.

12. The host application constructs the key token from the formatted data from step 10, the signature created in step 11, and any appropriate CRLs. The host application then sends the key token to the terminal.

Initialize Remote Device Commands

Quick Reference

Table 10-2 identifies each command by number, name, and purpose.

Table 10-2 Initialize Remote Device Commands

Command	Name	Purpose
120	Generate RSA Key Pair	Generates an RSA public and private key pair.
121	Load RSA Key into the RSA Key Table	Stores an RSA key in the volatile RSA key table.
122	Generate AKB for Root Public Key	Generates the AKB for the root public key. This command is deprecated, use command 12A instead.
123	Verify Public Key and Generate AKB$_{PUB}$	Verifies the digital signature of a public key (or certificate), and generates the AKB for the public key.

Table 10-2 Initialize Remote Device Commands (continued)

Command	Name	Purpose
124	Generate Digital Signature	Generates a digital signature using either the SHA1, SHA-256, or MD5 algorithms.
125	Verify Digital Signature	Verifies a digital signature using either the SHA1, SHA-256, or MD5 algorithms.
126	Generate Message Digest	Generates a message digest (hash) using the SHA1, SHA-256, or MD5 algorithms.
12A	Load Root Public Key	Generates the AKB for the public key.
12B	Translate Public or Private Key AKB from MFK to PMFK	Translates an RSA Key from encryption under the Master File Key to encryption under the Pending Master File Key.
12C	Verify or Delete RSA Key Table Entry	Verifies or deletes an RSA key table entry.
12F	Generate ATM Master Key and encrypt with public key	Generates an ATM Master Key and encrypts it under an RSA public key.
135	Export RSA Private Key	Returns the components of the RSA private key in ECB encrypted format.
136	Generate TR-34 Key Block	Generates a random 3DES key and returns a signing token.
139	Sign TR-34 Message	Signs a TR-34 message.
358	Generate ISO/IEC 9796-2 Digital Signature	Generates an ISO/IEC 9796-2 digital signature.

Generate RSA Key Pair (Command 120)

Use this command to generate an RSA private and public key pair. The generated RSA key pair will be formatted in the Atalla Key Block and encrypted under the Master File Key. The private key can also be stored in the volatile RSA key table.

In version 1.50 and above, support for TR-34 signing and wrapping keys has been added.

This command is a premium value command, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<120#Key Usage#Public Exponent#Size of Modulus#[Key Slot#]>
```

Response

<220#AKB$_{PUB}$#AKB$_{PRV}$#Check Digits#[Key Slot#]>[CRLF]

Calling Parameters

`120`

Field 0, the command identifier.

`Key Usage`

Field 1, defines how the RSA key will be used. The content of this field determines the header of the generated RSA key. The possible values of this field are:

Value	Description	Private Key Header	Public Key Header
K	Key Encryption Key	1KRDE000	1KREE000
S	Signing Key	1SRGE000	1SRVE000
T*	Token Exchange Key	1TRDE000	1TREE000
b*	Custom	1bRDE000	1bREE000
k	TR-34 Key Block Encryption Key	1kRDE000	1kREE000
p*	Signing Key	1pRGE000	1pRVE000
s*	Signing Key	1sRGE000	1sRVE000
w	TR-34 Key Block Signing Key	1wRGE000	1wRVE000

* For use in a customer specific command.

`Public Exponent`

Field 2, the public exponent, in ASCII-hexadecimal format. The public exponent length must be less than or equal to modulus length, its value must be odd and less than the value of the modulus. For example, if a 2048 bit modulus will be generated, and the public exponent will also be 2048 bits in length, the public exponent's most significant bit must be zero. The value of this field cannot be 01. The Fermat 4 value (65537) is represented as 010001.

`Size of Modulus`

Field 3, size of modulus in bits. The value in this field must be in the range of 1024 through 4096.

`[Key Slot#]`

Field 4, the key slot in the RSA key table where the generated RSA private key will be stored. This field is optional. When the value is in the range of 0 through 199, the public and private key will be AKB formatted and returned in the response. The private key will also be stored in the specified key slot, overwriting any existing value that may be present in the slot. When the value is an asterisk character "*", the public and private key will be AKB formatted and returned in the response. The private key will also be stored in the first available empty key slot.

Command 121 can be used to store a public key in the RSA key table.

Table 10-3 Command 120: Generate RSA Key Pair

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	120
1	Key Usage	1	K, S,T, b, k, p, s, or w
2	Public Exponent	Variable	0 - 9, A - F
3	Size of Modulus	4	1024 - 4096
4	[Key Slot#]	0-3	0 - 199, *

Responding Parameters

`220`

Field 0, the response identifier.

$\mathrm{AKB}_{\mathrm{PUB}}$

Field 1, the RSA public key in AKB format.

$\mathrm{AKB}_{\mathrm{PRV}}$

Field 2, the RSA private key in AKB format.

`Check Digits`

Field 3, the RSA private key check digits. Check digits are the leftmost 8 bytes of the SHA-1 message digest of the RSA key.

`[Key Slot#]`

Field 4, the key slot in the RSA key table where the RSA private key is stored. This field is returned only when field 4 is present in the command.

Table 10-4 Response 220: Generate RSA Key Pair

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	3	220
1	AKB_{PUB}	Variable	Printable characters
2	AKB_{PRV}	Variable	Printable characters
3	Check Digits	8	0 - 9, A - F
4	[Key Slot#]	0-3	0 - 199

Usage Notes

- When the size of modulus is not a multiple of 64 or is greater than 2048, this command may take 10 or more seconds to return a response.
- This command generates a private key where p > q.

Example

Generate a 2048 bit RSA signing key pair, and store the private key in slot 6

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

The command looks like this:

```
<120#S#010001#2048#6#>
```

The Network Security Processor returns a response similar to this:

```
<220#1SRVE000,00030100010100A569A253DE19422B6389097F98336D0ECC4A2
F01C5589BF7DF3E2E20D08D82CA768AEA4773612C0D539C3674732F86922E9BD5
CF11AF566BDAEC4A3F01EF659BA60218928F730122F1E2E4D0493B4F4CF84CC4B
5D19839BAA7743232F3747717917FAFF83C3C0D75033563CFB7959EEB832551E0
E3B4BFDCBF683E94E5AA47B7A1CE5372450DDF58BAEBE5235B97999DC3833D6A3
C9DC54FD72519911BFC979977BC5246310EC1FA52A17C828FC138A7C3779D69F3
1E152317E45A3E011D2359DDF8EAE1C2DBAB7327E88F19962E2829B78A014A752
1A3A8CD16665E8CB77EF7A0FBA4FA9F56BFAFA737D266C064BBF648C755DCD077
```

```
12CB5C75A195B09D4297,68CA8A795D25414E#1SRGE000,742DF5AACBB74E4B7F
C994B2D9ED44D9257436688C92300DB8ED4946D48646776D1663164AA5B24DA5B
9E903D7C12EA0B82AFA31B821CB0B91EA3E7374D1BF7BDF540FA32F5D72C7602E
BCFC6AB493D19AF87D9B24204EFAAAAB70F034D6616B549961E624AB7C69E9EB
A00248D5FBB1976AE7E58B8F487560ADB9730C8C8256325D860FB4716D1DA10CF
6E9620E3A37D420F656BFA2D99454A88A5A82ABCFDC6EB9E8E309C3A24831846E
7F781619BC44A204A9192362F38F56093332FC920BDD6DEEB02BA3FB0B9DB0151
68B109453F7AC38A40615316F667B09C2082566EA4F403D2617179DCCB3CE8541
9B0EED2526DC54984795D7B9973B24B7E97C0EB797DD0D87BE89FFFA5DE39F61C
D81AAD2CFDFC1F34D5520CDC8513CDAFACD7F1437AB8A49E4DE0C88CCF53A9F3B
3229B863B65F6FCA8CDF26F862BD7FAB682411862E2D5B776531014C57A47381E
C7A7179BB156F79012850AE86BEAAE48AE77E40B25E3181876D6EB8308815EBD2
020D16674E36E33CF86A654B0A7654D846B66A7513770377AD5A70C67CB63093E
016C31AA244C5B046D63FC224B721121B8CA62161F9EC0914D553D5FBA2C93305
CEB0F367258F3C2E1EB60785A0E6E1742FDF9D81A4699694E758D496CB417A3B8
6488013C1ACA111D99C75453B340582ABE4BA267BB32E5A499B6667BA3F72C840
1048EA0C3845013CD3C6AD267E0B1F31511A6B61BE738DEAF15E5809D437A4646
3906159B06B8436895897BD84C940F5DDF51635D36D5581AD3F91D89E7DAD808C
61CC3D6C726620F326FDEE9AF91D48FE4963AE7C57A30E2B6D99A2DEB41B8051D
F9DDBA249AC7B8D7DE30F40F378FEAD1FBB00EC330A34341C569F653FFF42556B
2AB971D36D6C4A1FA1409D80F8EF1AA527515495B84CDBFA645E8AB27868674A8
8891E967542C84362E96583EC4FB12333E81CF738FFB19BEC33C78489314ECBBD
0254144A90FBB69E6F04BCC15C9755D3121E45DAA2B3B67153A7700951AEA35EB
02DB16083D80D4AD0AE3EC264F1790935CA8D60E5354A0B76F081B04E1AFE5C95
8FA6CE5AC61B12495955A5AD40131E2F04E59563CD431D2A56C439E217A1DE013
ECD33986B67B8B82B8DAF3E7B121134A5A81DF091A27F226DD07A42429CE4AB10
83FA69B3EEFD2E83DF7376ECE320D31D729583B4505B4B11B39E88DC115AFEB02
FD51F84B67E475E9C0F1588244B1197E811AE42FE6DCFB769C0B349F94E311C7A
F324C8EFEB555BAEE3F205AD8B97F802D4E4599EAA6E91B04F40010BFC90ECD26
B1448829FB64FC62E53DE0F08115AD291F0D72A94ECEF17EF64BD5BF832D46965
9744C61339AE904435D35E9AC697493FEC18CADC1C2AC724AF0DFE652C7693D55
3799147894DDDC61A21D7E9F9711EB844E3D92DCFF04B40A5989F1B525D8C45FC
FB35DD038048A5330BFA97AA40C2342A1F28CFB964B470811E06CA81DA90815EA
D18471B1ADE1632CAD6D60840E4E350E3F4FAB309136B86F4FA0F5CA2F66F0DF4
14624875DD8E07E3401FA0B345FF224A394ABCE25CDEFD2E5108153ECB46B7D61
53E454E83C1781BB3318666B8F3242F021EDE0C3F153BA4384A836C0A15CFDEDB
EB0B400013687F7A207A55FF23F4411A74EDD9CF37459D2E93588C35974,9A031
480C9C3EA95#08B8E534#006#>
```

Load RSA Key into RSA Key Table (Command 121)

This command stores either a public or private RSA key in the volatile RSA key table. Any existing value in the specified key slot will be overwritten when this command is executed.

The value of the public exponent must be odd, cannot be 01, and must not exceed the value of the modulus.

This command is a premium value command, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

<121#Key Slot#AKB$_{PUB}$ or AKB$_{PRV}$#>

Response

```
<221#Key Slot#Check Digits#>[CRLF]
```

Calling Parameters

`121`

Field 0, the command identifier.

`Key Slot`

Field 1, a key slot in the RSA key table where the RSA key will be stored. This field must contain either a value in the range of 0 through 199 or an asterisk character "*". When the value is an asterisk character "*", the RSA key will be stored in the first available empty key slot.

AKB$_{PUB}$ or AKB$_{PRV}$

Field 2, the AKB of the public key or private key to be stored in the RSA key table. In version 1.40 and above, this new header value 1RRVN000 is allowed. In version NSP 1.50 and above, these header values are allowed: 1KRDE000, 1KREE000, 1RRVN000, 1RRVN00k, 1SRGE000, 1SRVE000, 1TRDE000, 1TREE000, 1kRDE000, 1kRDN000, 1kREE000, 1kREN000, 1pRGN000, 1pRVN000, 1wRGE000, 1wRGN000, 1wRVN000, and 1wRVE000.

Table 10-5 Command 121: Load RSA Key into RSA Key Table

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	121

Table 10-5 Command 121: Load RSA Key into RSA Key Table (continued)

Field	Contents	Length (bytes)	Legal Characters
1	Key Slot	1-3	0 - 199 or *
2	AKB_{PUB} or AKB_{PRV}	Variable	Printable characters

Responding Parameters

`221`

Field 0, the response identifier.

`Key Slot`

Field 1, the key slot in the RSA key table where the RSA key has been stored.

`Check Digits`

Field 2, RSA key check digits. Check digits are the leftmost 8 bytes of the SHA-1 message digest of the RSA key.

Table 10-6 Response 221: Load RSA Key into RSA Key Table

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	3	221
1	Key Slot	1-3	0 - 199
2	Check Digits	8	0 - 9, A - F

Example

Use command 121 to load an AKB of a public key into key slot 1 of the RSA key table.

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

The command looks like this:

```
<121#1#1SRVE000,00030100010100AF79E1E223A5DDC8EE7599D4817FDFAD04A
AA3911D82D615C8F166E4C6391A9E5C8724628E62D9075836535642ABD14EAA68
1CD7EC8A42166870FB366E0863FC2C3A3B4969B9F46F1DA0C31D1B313B404ECC7
3AE15677D13FE086CE0758890FDF7455905948D4FD827655FD96908EA4180611A
2ED38D876CF31B23EC739F6F3F34048BFA3AEDA6090AFA205E1C19D0E3252366D
3DC2C6BECBA6ABDDA79C899150302B3192F986AEF29674BA56A648A928A9ECB84
DE2E4600D80538811AA7475CB6B559BA6D5D3C0D0F7F91A393B30266AAD1FB27D
F6BDC7A938B64D96F9132501E5320CE0C93BD244C9B6A41172ABF9B9D927AB233
13B17677293E9E15DB9ABB,F7929A1B295C6A00#>
```

The Network Security Processor returns the following response:

```
<221#001#8F6D7481#>
```

Generate AKB for Root Public Key (Command 122)

This command generates an AKB for the root public key. The AKB header value will be 1SRVN000.

This command has been deprecated, new applications should use command 12A.

This command is a premium value command, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<122#Public Exponent#Modulus#EMFK.30(Hash-32)#>
```

Response

```
<222#AKBPUB#>[CRLF]
```

Calling Parameters

`122`

Field 0, the command identifier.

`Public Exponent`

Field 1, the public exponent in ASCII-hexadecimal format. The value of the public exponent must be odd, cannot be 01, and must be less than the value of the modulus.

`Modulus`

Field 2, the modulus in ASCII-hexadecimal format.

$E_{MFK.30}$`(Hash-32)`

Field 3, the Hash-32 encrypted under variant 30 of the MFK. Hash-32 is the leftmost 32 characters of the SHA1 hash of the public key. Use Command 126 to generate the SHA-1 hash of the public key. When generating the public key hash, the public key must be provided in the following format:

```
public exponent length|public exponent|modulus length|modulus
```

The Example details the procedure to create this value.

Table 10-7 Command 122: Generate AKB for Root Public Key

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	122

Table 10-7 Command 122: Generate AKB for Root Public Key (continued)

Field	Contents	Length (bytes)	Legal Characters
1	Public Exponent	Variable	0 - 9, A - F
2	Modulus	Variable	0 - 9, A - F
3	$E_{MFK.30}$(Hash-32)	32	0 - 9, A - F

Responding Parameters

`222`

Field 0, the response identifier.

AKB$_{PUB}$

Field 1, the root public key in AKB format.

Table 10-8 Response 222: Generate AKB for Root Public Key

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	3	222
1	AKB$_{PUB}$	Variable	Printable characters

Usage Notes

In version 1.42 and above, the NSP will return an error if the value of the public exponent is even or 01.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Command 122 requires the SHA1 hash of the public key to be encrypted under variant 30 of the MFK. A two-step procedure is required to generate this value.

1) Use command 126 to generate the SHA-1 hash.

- Public exponent = 010001
- Modulus =
AF79E1E223A5DDC8EE7599D4817FDFAD04AAA3911D82D615C8F166E4C6391A9
E5C8724628E62D9075836535642ABD14EAA681CD7EC8A42166870FB366E0863F
C2C3A3B4969B9F46F1DA0C31D1B313B404ECC73AE15677D13FE086CE0758890F
DF7455905948D4FD827655FD96908EA4180611A2ED38D876CF31B23EC739F6F3

F34048BFA3AEDA6090AFA205E1C19D0E3252366D3DC2C6BECBA6ABDDA79C8
99150302B3192F986AEF29674BA56A648A928A9ECB84DE2E4600D80538811AA
7475CB6B559BA6D5D3C0D0F7F91A393B30266AAD1FB27DF6BDC7A938B64D96
F9132501E5320CE0C93BD244C9B6A41172ABF9B9D927AB23313B17677293E9E15
DB9ABB

The command 126 looks like this:

```
<126#0#3#0#263#00030100010100AF79E1E223A5DDC8EE7599D4817FDFAD04AA
A3911D82D615C8F166E4C6391A9E5C8724628E62D9075836535642ABD14EAA681
CD7EC8A42166870FB366E0863FC2C3A3B4969B9F46F1DA0C31D1B313B404ECC73
AE15677D13FE086CE0758890FDF7455905948D4FD827655FD96908EA4180611A2
ED38D876CF31B23EC739F6F3F34048BFA3AEDA6090AFA205E1C19D0E3252366D3
DC2C6BECBA6ABDDA79C899150302B3192F986AEF29674BA56A648A928A9ECB84D
E2E4600D80538811AA7475CB6B559BA6D5D3C0D0F7F91A393B30266AAD1FB27DF
6BDC7A938B64D96F9132501E5320CE0C93BD244C9B6A41172ABF9B9D927AB2331
3B17677293E9E15DB9ABB##>
```

The Network Security Processor returns the following response:

```
<226#0#8F6D7481A85C02DAEAAAA11F4CFF3F3A245F6CA9#>
```

- Hash-32 -The leftmost 32 characters of the hash returned in field 2 of the response. The Hash-32 is: `8F6D7481A85C02DAEAAAA11F4CFF3F3A`

2) Use the SCA's challenge/response feature, which is located in the Calculate AKB function, to encrypt the Hash-32 value under variant 30 of the MFK. Choose 2key-3DES (Double) for the challenge length.

- Emfk.30(Hash-32) = 5B747FD2E5F8C8FC11443A86474466AF

Verify the public key and generate the AKB

The command looks like this:

```
<122#010001#AF79E1E223A5DDC8EE7599D4817FDFAD04AAA3911D82D615C8F16
6E4C6391A9E5C8724628E62D9075836535642ABD14EAA681CD7EC8A42166870FB
366E0863FC2C3A3B4969B9F46F1DA0C31D1B313B404ECC73AE15677D13FE086CE
0758890FDF7455905948D4FD827655FD96908EA4180611A2ED38D876CF31B23EC
739F6F3F34048BFA3AEDA6090AFA205E1C19D0E3252366D3DC2C6BECBA6ABDDA7
9C899150302B3192F986AEF29674BA56A648A928A9ECB84DE2E4600D80538811A
A7475CB6B559BA6D5D3C0D0F7F91A393B30266AAD1FB27DF6BDC7A938B64D96F9
132501E5320CE0C93BD244C9B6A41172ABF9B9D927AB23313B17677293E9E15DB
9ABB#5B747FD2E5F8C8FC11443A86474466AF#>
```

The Network Security Processor returns the following response:

```
<222#1SRVN000,00030100010100AF79E1E223A5DDC8EE7599D4817FDFAD04AAA
3911D82D615C8F166E4C6391A9E5C8724628E62D9075836535642ABD14EAA681C
D7EC8A42166870FB366E0863FC2C3A3B4969B9F46F1DA0C31D1B313B404ECC73A
E15677D13FE086CE0758890FDF7455905948D4FD827655FD96908EA4180611A2E
D38D876CF31B23EC739F6F3F34048BFA3AEDA6090AFA205E1C19D0E3252366D3D
C2C6BECBA6ABDDA79C899150302B3192F986AEF29674BA56A648A928A9ECB84DE
```

```
2E4600D80538811AA7475CB6B559BA6D5D3C0D0F7F91A393B30266AAD1FB27DF6
BDC7A938B64D96F9132501E5320CE0C93BD244C9B6A41172ABF9B9D927AB23313
B17677293E9E15DB9ABB,B4B87296C77F55E9#>
```

Verify Public Key and Generate AKB_{PUB} (Command 123)

Command 123 – This command verifies the digital signature of a public key (or certificate) and then returns the public key in AKB format.

In version 1.40 and above, support for SHA-256 has been added.

In version 1.50 and above, support for TR-34 encryption and signing keys has been added.

This command is a premium value command, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
 <123#Key Usage#Algo#Hash Encoding#Public Exponent#
Modulus#Public Key Data#Digital Signature#Key Slot or AKBPUB#>
```

Response

```
<223#AKBPUB#[Check Digits#]>[CRLF]
```

Calling Parameters

`123`

Field 0, the command identifier.

`Key Usage`

Field 1, after the public key has been verified, it is returned in AKB format. This field defines byte 1 of the AKB returned in field 1 of the response.

Value	Description
K	Key Encryption Key
S	Signing Key
k	TR-34 Key Block Encryption Key
w	TR-34 Key Block Signing Key

`Algo`

Field 2, the algorithm that was used to generate message digest when the certificate was generated.

Value	Description
2	MD5
3	SHA1
4	SHA-256

`Hash Encoding`

Field 3, the encoding method that was used to format the message digest when the signature was generated.

Value	Description
0	No encoding
1	Basic Encoding Rules (BER) encoding

`Public Exponent`

Field 4, the public exponent, in ASCII-hexadecimal format, of the public key to be verified. The value of the public exponent must be odd, cannot be 01, and must be less than the value of the modulus.

`Modulus`

Field 5, the modulus, in ASCII-hexadecimal format, of the public key to be verified.

`Public Key Data`

Field 6, this field must contain the exact data that the certificate authority used to generate the digital signature present in the certificate. The public exponent and modulus, specified in fields 4 and 5 respectively, must be present in this field.

`Digital Signature`

Field 7, the digital signature of the public key to be verified.

`Key Slot`

Field 8, the key slot in the RSA key table where the Certificate Authority's public key is stored. Command 121 must be used to store the public key in the RSA key table.

Or

$\texttt{AKB}_{\texttt{PUB}}$

Field 8, the Certificate Authority's public key in AKB format. In version 1.40 and above, this header 1RRVN000 is supported. In version 1.50 and above, this additional header is supported: 1RRVN00k. When option 43 is enabled, these additional headers are allowed: 1SRVN000 and 1SRVE000.

Table 10-9 Command 123: Verify Public Key and Generate AKB

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	123
1	Key Usage	1	K, S, k, or w
2	Algo	1	2, 3 or 4
3	Hash Encoding	1	0 or 1
4	Public Exponent	Variable	0 - 9, A - F
5	Modulus	2*SOM	0 - 9, A - F
6	Public Key Data	Variable	0 - 9, A - F
7	Digital Signature	2*SOM	0 - 9, A - F
8	Key Slot	1-3	0 - 199
or			
8	AKB$_{PUB}$	Variable	Printable characters

Responding Parameters

`223`

Field 0, the response identifier.

`AKB`$_{PUB}$

Field 1, the verified public key in AKB format.

`[Check Digits#]`

Field 2, the verified public key’s check digits. Check digits are the leftmost 8 bytes of the SHA-1 message digest of the RSA key. This field is present only when field 8 of the command contains a value in the range of 0 through 199.

Table 10-10 Response 223: Verify Public Key and Generate AKB

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	3	223
1	AKB$_{PUB}$	Variable	Printable characters
2	[Check Digits#]	8	0 - 9, A - F

Usage Notes

- For an X.509 certificate, the data supplied in field 6 is the structure TBSCertificate as defined in RFC 5280.
- An error message that starts with <00#0708 indicates that the key slot specified in field 8 of the command is empty or contains an invalid AKB.
- An error response that starts with <00#0807 or <00#0107 indicates that the signature did not verify.
- The AKB header byte 3 (Algorithm) for the Certificate Authority's public key, supplied in field 8, determines the AKB header byte 3 (Algorithm) of the verified public key returned in field 1 of the response.
- The Key Usage value, specified in field 1 of the command, determines the AKB header bye 1 (Key Usage) of the verified public key returned in field 1 of the response.
- Some public key systems prepend a byte of 00 to the modulus. For example, a modulus for a 2048-bit public key should contain 256 bytes, not 257 bytes (00 + the actual 256 byte modulus). Do not include this 00 byte when supplying the modulus in command field 5.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

The command looks like this:

```
<123#S#2#0#010001#C5C2AF49A4550F48EBC98537A19C4D0C88790F0EC6FC937
2C40C8379ECDABB8EBB2B9DCA9397193AE306E58E8775A304AABE1CAF847B6247
AC50DCFA5CB1E5D8D7EE59D17083DCF438F18E6A24EA0C813B752732E5C9198DD
9FF07ADFA8543D5808F3E6B827B1C90EDE5CA94DFAFA85159D19E2D55DDF64970
8989377B43456352E8D181791C5E9D00046A41B0D2625ED54720EA455041161C9
C827F463B843CE646A0E4581638960AC3DD54CB4AB5C4D3E28556051AB0857F02
9602C863CFADD38268D86118B99494E79A31EBC06F33215C1120AD723415AC5C5
3DD33B7946FE038EFC98864CFBAE599208603083F3BF9A733C6484DC0741A17E2
D40FDFE167#0100010001020304050607080909A0B0C0D0E0F1011121314151617
18191A1B1C1D1E1F202122232425262728292A2B2C2D2E2F3031323334353637 3
8393A3B3C3D3E3F404142434445464748494A4B4C4D4E4F5051525354555657 58
595A5B5C5D5E5F606162636465666768696A6B6C6D6E6F7071727374757677787
97A7B7C7D7E7F808182838485868788898A8B8C8D8E8F909192939495969798 99
9A9B9C9D9E9FA0A1A2A3A4A5A6A7A8A9AAABACADAEAFB0B1B2B3B4B5B6B7B8B9B
ABBBCBDBEBFC0C1C2C3C4C5C6C7C8C9CACBCCCDCECFD0D1D2D3D4D5D6D7D8D9DA
DBDCDDDEDFE0E1E2E3E4E5E6E7E8E9EAEBECEDEEEFF0F1F2F3F4F5F6F7F8F9FAF
BFCFDFEFFC5C2AF49A4550F48EBC98537A19C4D0C88790F0EC6FC9372C40C8379
ECDABB8EBB2B9DCA9397193AE306E58E8775A304AABE1CAF847B6247AC50DCFA5
CB1E5D8D7EE59D17083DCF438F18E6A24EA0C813B752732E5C9198DD9FF07ADFA
8543D5808F3E6B827B1C90EDE5CA94DFAFA85159D19E2D55DDF649708989377B4
3456352E8D181791C5E9D00046A41B0D2625ED54720EA455041161C9C827F463B
```

```
843CE646A0E4581638960AC3DD54CB4AB5C4D3E28556051AB0857F029602C863C
FADD38268D86118B99494E79A31EBC06F33215C1120AD723415AC5C53DD33B794
6FE038EFC98864CFBAE599208603083F3BF9A733C6484DC0741A17E2D40FDFE16
7#7A81514F9E535A0D89068E969E5F46DBDC17830B60C768135F916D6AED12FAB
137796B7606C3DB0F027A726BF1AFE21783150F31AE97FA7F9442F6EE26359E68
F78D9164310494B22FED2085401FFDC59FE33E00F1324F22FC06F593DAA9CACA5
445BD09AA89267C5DCE69683CB2304AD443ADDAA8B73F11A324E62A7124D2BD08
CD9F718421EF95673CCE5B39E5CD252DA9050A4DED56F0624CAC74CDA4ECD6593
4B7D9095B6F8A6A458A6C1C4E709ED71508B9F97B5970C5E603AB7798BC1784FF
52DF256C35D79A0B4B9EC404DCFA477EC588076F07A2F28EC4FCAD9FB41B50481
A23BA732356C607F992A65BBA085C4BB01647CA194EFA908CAE434EB71D#1SRVE
000,00030100010100AF79E1E223A5DDC8EE7599D4817FDFAD04AAA3911D82D61
5C8F166E4C6391A9E5C8724628E62D9075836535642ABD14EAA681CD7EC8A4216
6870FB366E0863FC2C3A3B4969B9F46F1DA0C31D1B313B404ECC73AE15677D13F
E086CE0758890FDF7455905948D4FD827655FD96908EA4180611A2ED38D876CF3
1B23EC739F6F3F34048BFA3AEDA6090AFA205E1C19D0E3252366D3DC2C6BECBA6
ABDDA79C899150302B3192F986AEF29674BA56A648A928A9ECB84DE2E4600D805
38811AA7475CB6B559BA6D5D3C0D0F7F91A393B30266AAD1FB27DF6BDC7A938B6
4D96F9132501E5320CE0C93BD244C9B6A41172ABF9B9D927AB23313B17677293E
9E15DB9ABB,F7929A1B295C6A00#>
```

The Network Security Processor returns the following response:

```
<223#1SRVE000,00030100010100C5C2AF49A4550F48EBC98537A19C4D0C88790
F0EC6FC9372C40C8379ECDABB8EBB2B9DCA9397193AE306E58E8775A304AABE1C
AF847B6247AC50DCFA5CB1E5D8D7EE59D17083DCF438F18E6A24EA0C813B75273
2E5C9198DD9FF07ADFA8543D5808F3E6B827B1C90EDE5CA94DFAFA85159D19E2D
55DDF649708989377B43456352E8D181791C5E9D00046A41B0D2625ED54720EA4
55041161C9C827F463B843CE646A0E4581638960AC3DD54CB4AB5C4D3E2855605
1AB0857F029602C863CFADD38268D86118B99494E79A31EBC06F33215C1120AD7
23415AC5C53DD33B7946FE038EFC98864CFBAE599208603083F3BF9A733C6484D
C0741A17E2D40FDFE167,863C9BA6985D1E00#>
```

Below is a sample Diebold EPP Encryption Certificate that has been base 64 decoded. This certificate was not used in the example above. It is included here for reference only, the *italicized comments* show the data required in fields 4 through 7 of the command 123.

Start of Certificate

```
30 82 03 2B
   06 09 2A 86 48 86 F7 0D 01 07 02
   A0 82 03 1C
      30 82 03 18
         02 01 01
         31 00
            30 0B
          06 09 2A 86 48 86 F7 0D 01 07 01
         A0 82 03 00
       30 82 02 FC
```

Start of Signed certificate data (Field 6 in the command 123)

```
               30 82 01 E4
                  A0 03
                     02 01 02
```

```
02 11
   00 D0 1E 47 3E 00 00 00 F4
   6C 47 94 38 00 00 25 A6
30 0D
   06 09
      2A 86 48 86 F7 0D 01 01 05
      05 00
   30 2B
      31 0B
         30 09
            06 03 55 04 06
            13 02 55 53
      31 0C
         30 0A
            06 03 55 04 0A
            13 03 44 53 54
      31 0E
         30 0C
            06 03 55 04 03
            13 05 43 41 20 41 37
   30 1E
      17 0D
         30 33 30 34 30 38 30 37 35 37 33 30 5A
      17 0D
         31 33 30 34 30 35 30 38 35 37 33 30 5A
   30 34
      31 13
         30 11
            06 03 55 04 03 13 0A 30 35
            30 31 38 35 35 39 20 45
      31 10
         30 0E
            06 03 55 04 0A
            13 07 44 69 65 62 6F 6C 64
      31 0B
         30 09
            06 03 55 04 06
            13 02
               55 53
   30 82 01 22
      30 0D
         06 09
            2A 86 48 86 F7 0D 01 01 01
         05 00
      03 82 01 0F
         00
         30 82 01 0A
            02 82 01 01
                  00
```

Start of modulus (Field 5 in the command 123)

```
AB C1 68 86 83 1E EA 3F 79 86 33 39 94 28 67 84
C6 B9 E7 A6 75 9E 9D 65 D2 33 47 22 AC 38 D8 4D
F3 51 C8 20 04 ED 2B 30 38 59 A1 A5 79 47 39 9A
```

```
                        19 2F 5F 8B 35 5B 7F CE B4 60 61 C8 0C 48 D0 97
                        AD 43 43 64 77 D4 39 11 19 BD 26 8E 7D 79 CB D6
                        2D 71 49 9F 89 C3 E1 0F 0A 6E 98 C3 D0 66 C6 96
                        17 88 C8 44 A3 4D 8C 59 BB C6 C4 A3 A2 A3 F4 E3
                        8E C8 6C 93 ED 48 92 59 4E 54 AC C4 CC F1 91 4D
                        FA 3E 41 E2 22 96 20 8D 27 0F 50 A9 0F 53 98 D1
                        4D BA D1 15 1A 5F D0 44 42 09 43 75 BC A6 13 D6
                        77 70 FC D7 0A D3 10 79 1A 9D B0 2E 3C 01 D9 6A
                        FA FD 8B 9B 0E A8 06 D1 6C 69 88 45 C1 DA E6 F3
                        62 9D C2 16 01 B9 3A CC E0 59 EC E2 A6 8E A2 26
                        5B 79 65 58 17 23 78 F4 14 6C C6 AF 2C 8C 7D 9B
                        0B C4 39 FA 29 6A 3C 55 9A D4 C3 18 77 5E B2 85
                        20 F7 DF 0B AD 46 3A EB 1F 42 3B 87 CC 78 11 C5
```

End of Modulus (Field 5 in the command 123)

```
                        02 03
```

Start of Public Exponent (Field 4 in the command 123)

```
                          01 00 01
```

End of Public Exponent (Field 4 in the command 123)

```
                      A3 12
                         30 10
                            30 0E
                               06 03 55 1D 0F
                               01 01 FF
                               04 04
                                    03 02 04 30
```

End of signed certificate data (Field 6 in the command 123)

```
                 30 0D
                    06 09
                       2A 86 48 86 F7 0D 01 01 05
                    05 00
                 03 82 01 01
                    00
```

Start of Digital Signature (Field 7 in the command 123)

```
                    06 46 D9 79 64 55 9E 80 38 49 19 90 F7 A1 B7 F6
                    4F FB 13 40 BD 72 7F 98 0D 2D 36 39 56 7B E1 C9
                    E7 B7 F3 BF 77 4B 81 74 7B 7E 5E 61 2B 95 DC 88
                    D8 BF F9 FF 33 32 DC 0A 97 C7 A5 56 BA 53 82 89
                    BE 25 8B 8C 3C 01 04 1F 8A 67 06 79 AA EE A9 2A
                    81 C7 64 7C 81 CB 5F 86 9B FD F1 E4 BA F3 62 3E
                    97 D7 72 BF 4F 7C 55 57 33 83 8A CF 0C C2 5E 05
                    43 E7 F5 6C CE C6 64 B6 66 01 E3 8F 77 06 DB 73
```

```
CD F0 AC DC 75 38 96 7A FB 54 6C 85 86 9C 7E 84
63 37 56 BC 9E A4 23 B0 68 E5 6F 33 F7 C9 8E AA
7A B7 0B 48 E0 91 CB 02 3C 78 A2 5A E0 56 EA 3B
EA 48 EB BC C3 7C 02 EA CD EB 37 65 D1 FC E1 42
CC 66 37 8D 31 FE 50 22 39 BA 21 B1 42 29 6A 00
17 F6 A7 A6 F1 DC 2B C7 5E C8 A6 1C 8B D2 45 40
22 EE CA 05 9A C4 4D 15 2F DA 17 6E 39 3E A1 81
F5 77 2B BF 38 49 36 58 4C 8E 50 A2 5E 00 64 11
```

End of Digital Signature (Field 7 in the command 123)

```
31 00
```

End of Certificate

Generate Digital Signature (Command 124)

Command 124 – This command generates a digital signature using either the SHA1, SHA-256, or MD5 hash algorithm.

Prior to being encrypted by the RSA private key, the message digest (hash) is left-padded per PKCS#1 version 1.5.

```
0x00||0x01||Array of 0xFF||00||hash
```

where the length of `Array of 0xFF` = sizeof(modulus) - sizeof(hash) - 3.

In version 1.40 and above, support for SHA-256 has been added.

This command is a premium value command it is not enabled in the Network Security Processor’s default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor’s security policy.

Command

```
<124#Reserved#Algo#Hash Encoding#Data-length#Data#Reserved#
Key Slot or AKBPRV#>
```

Response

```
<224#Reserved#Digital Signature#[Check Digits#]>[CRLF]
```

Calling Parameters

`124`

Field 0, the command identifier.

`Reserved`

Field 1, this field must contain the number zero.

`Algo`

Field 2, the algorithm that will be used to generate the message digest.

Value	Description
2	MD5
3	SHA1
4	pre-calculated hash is provided in field 5
5	SHA-256

`Hash Encoding`

Field 3, this field specifies how the message digest will be encoded when field 2 contains the value 2, 3, or 5.

Value	Description
0	No encoding
1	Basic Encoding Rules (BER) encoding

When field 2 contains the value 4, the Network Security Processor will not perform BER encoding even if this field contains the value 1. If necessary, it is the host application's responsibility to encode the pre-calculated hash.

`Data-length`

Field 4, the number of bytes of data or pre-calculated hash to be signed. The value in this field must be the number of hexadecimal characters in field 5 divided by 2. For example, if field 5 contains 128 hexadecimal characters, this field must contain the value 64.

When field 2 contains the value 4, this field must be empty. When field 2 contains a value other than 4, the value in this field must be in the range of 1 through 15000.

`Data`

Field 5, the data or the pre-calculated hash, in unpacked ASCII-hexadecimal format, to be signed. Each byte of data or pre-calculated hash is represented as two hexadecimal characters.

When field 2 contains the value 4, the minimum number of hexadecimal characters allowed in this field is 32. The maximum number of hexadecimal characters is computed as follows:

```
(length of modulus in bits / 4) - 22
```

For example, when the modulus is 2048 bits, the maximum number of hexadecimal characters allowed in this field is 490.

When field 2 contains a value other than 4, the number of hexadecimal characters allowed in this field must be in the range of 2 through 30000.

`Reserved`

Field 6, this field must be empty.

`Key Slot`

Field 7, a key slot in the RSA key table where the RSA private key, that will be used to generate the signature, is stored. Command 120 or 121 must be used to store the private key in the RSA key table.

Or

`AKB`$_{PRV}$

Field 7, the private key, in AKB format, that will be used to generate the signature. The header of the AKB must be 1SRGE000. In version 1.50 and above, these additional headers are supported: 1kRDE000 and 1kRDN000.

Table 10-11 Command 124: Generate Digital Signature

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	124
1	Reserved	1	0
2	Algo	1	2, 3, 4, or 5
3	Hash Encoding	1	0 or 1
4	Data-length	1-5	1-15000
5	Data	2-30000	0 - 9, A - F
6	Reserved	0	none
7	Key Slot	1-3	0 - 199
or			
7	AKB$_{PRV}$	Variable	Printable characters

Responding Parameters

`224`

Field 0, the response identifier.

`Reserved`

Field 1, this field will contain the number zero.

`Digital Signature`

Field 2, the generated digital signature. The length of this field will be twice the size of the modulus (2*SOM).

`[Check Digits#]`

Field 3, the check digits of the RSA private key. Check digits are the leftmost 8 bytes of the SHA-1 message digest of the RSA key. This field is present only when field 7 of the command contains a value in the range of 0 - 199.

Table 10-12 Response 224: Generate Digital Signature

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	3	224
1	Reserved	1	0
2	Digital Signature	2*SOM	0 - 9, A - F
3	[Check Digits#]	0, 8	0 - 9, A - F

Usage Notes

- An error message that starts with <00#0707 indicates that the key slot specified in field 7 of the command is empty or is invalid.
- If the prime exponent 1 (p) is less than the prime exponent 2 (q) the processing time for this command will be longer. This condition can only happen if the signing key has been imported.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Sign the certificate of the public key using the private key stored in slot 10

The command looks like this:

```
<124#0#2#0#515#0100010001020304050607080900A0B0C0D0E0F101112131415
161718191A1B1C1D1E1F202122232425262728292A2B2C2D2E2F3031323334353
63738393A3B3C3D3E3F404142434445464748494A4B4C4D4E4F50515253545556
5758595A5B5C5D5E5F606162636465666768696A6B6C6D6E6F707172737475767
778797A7B7C7D7E7F808182838485868788898A8B8C8D8E8F909192939495969 7
98999A9B9C9D9E9FA0A1A2A3A4A5A6A7A8A9AAABACADAEAFB0B1B2B3B4B5B6B7B
8B9BABBBCBDBEBFC0C1C2C3C4C5C6C7C8C9CACBCCCDCECFD0D1D2D3D4D5D6D7D8
D9DADBDCDDDEDFE0E1E2E3E4E5E6E7E8E9EAEBECEDEEEFF0F1F2F3F4F5F6F7F8F
9FAFBFCFDFEFFC5C2AF49A4550F48EBC98537A19C4D0C88790F0EC6FC9372C40C
8379ECDABB8EBB2B9DCA9397193AE306E58E8775A304AABE1CAF847B6247AC50D
CFA5CB1E5D8D7EE59D17083DCF438F18E6A24EA0C813B752732E5C9198DD9FF07
ADFA8543D5808F3E6B827B1C90EDE5CA94DFAFA85159D19E2D55DDF6497089893
77B43456352E8D181791C5E9D00046A41B0D2625ED54720EA455041161C9C827F
463B843CE646A0E4581638960AC3DD54CB4AB5C4D3E28556051AB0857F029602C
863CFADD38268D86118B99494E79A31EBC06F33215C1120AD723415AC5C53DD33
B7946FE038EFC98864CFBAE599208603083F3BF9A733C6484DC0741A17E2D40FD
FE167##10#>
```

The Network Security Processor returns the following response:

```
<224#0#7A81514F9E535A0D89068E969E5F46DBDC17830B60C768135F916D6AED1
2FAB137796B7606C3DB0F027A726BF1AFE21783150F31AE97FA7F9442F6EE26359
E68F78D9164310494B22FED2085401FFDC59FE33E00F1324F22FC06F593DAA9CAC
A5445BD09AA89267C5DCE69683CB2304AD443ADDAA8B73F11A324E62A7124D2BD0
8CD9F718421EF95673CCE5B39E5CD252DA9050A4DED56F0624CAC74CDA4ECD6593
4B7D9095B6F8A6A458A6C1C4E709ED71508B9F97B5970C5E603AB7798BC1784FF5
2DF256C35D79A0B4B9EC404DCFA477EC588076F07A2F28EC4FCAD9FB41B50481A2
3BA732356C607F992A65BBA085C4BB01647CA194EFA908CAE434EB71D#>
```

Verify Digital Signature (Command 125)

Command 125 – This command verifies a digital signature that was generated using either the MD5, SHA1 or SHA-256 hash algorithm.

In version 1.40 and above, support for SHA-256 has been added.

This command is a premium value command. It is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<125#Reserved#Algo#Hash Encoding#Data-length#Data#Reserved# Digital
Signature#Key Slot or AKB_PUB#>
```

Response

```
<225#Reserved#Verification Indicator#[Check Digits#]>[CRLF]
```

Calling Parameters

`125`

Field 0, the command identifier.

`Reserved`

Field 1, this field must contain the number zero.

`Algo`

Field 2, the algorithm used to generate the message digest:

Value	Description
2	MD5
3	SHA1
4	pre-calculated hash is provided in field 5
5	SHA-256

`Hash Encoding`

Field 3, this indicates how the message digest in the signature was encoded:

Value	Description
0	No encoding
1	Basic Encoding Rules (BER) encoding

`Data-length`

Field 4, the length of data or pre-calculated hash to be verified. When field 2 contains the value 4, this field must be empty. When field 2 contains a value other than 4, the value in this field must be in the range of 1 through 15000.

`Data`

Field 5, the data or the pre-calculated hash to be verified. When field 2 contains the value 4, the minimum number of hexadecimal characters allowed in this field is 32. The maximum number of hexadecimal characters is computed as follows:

```
(length of modulus in bits / 4) - 22
```

For example, when the modulus is 2048 bits, the maximum number of hexadecimal characters allowed in this field is 490.

When field 2 contains a value other than 4, the number of hexadecimal characters allowed in this field must be in the range of 2 through 30000.

`Reserved`

Field 6, this field must be empty.

`Digital Signature`

Field 7, the digital signature, in ASCII-hexadecimal format, to be verified. The length of this field will be twice the size of the modulus (2*SOM).

`Key Slot`

Field 8, the key slot in the RSA key table where the public key, used to verify the signature, is stored. Command 121 must be used to store the public key in the RSA key table.

Or

$\texttt{AKB}_{\texttt{PUB}}$

Field 8, the public key, in AKB format, that will be used to verify the signature. The header of the AKB must be 1SRVE000 or 1SRVN000. In version 1.40 and above, this additional header is supported: 1RRVN000. In version 1.50 and above, these additional headers are supported: 1wRVE000 and 1RRVN00k.

Table 10-13 Command 125: Verify Digital Signature

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	125
1	Reserved	1	0
2	Algo	1	2, 3, 4 or 5
3	Hash Encoding	1	0 or 1
4	Data-length	1-5	1-15000
5	Data	2-30000	0 - 9, A - F
6	Reserved	0	none
7	Digital Signature	2*SOM	0 - 9, A - F
8	Key Slot	1-3	0 - 199
	or		
8	AKB$_{PUB}$	Variable	Printable characters

Responding Parameters

`225`

Field 0, the response identifier.

`Reserved`

Field 1, this field will contain the number zero.

`Verification Indicator`

Field 2, the verification indicator.

Value	Description
N	The signature did not verify.
Y	The signature did verify.

`[Check Digits#]`

Field 3, the check digits of the RSA public key. Check digits are the leftmost 8 bytes of the SHA-1 message digest of the RSA key. This field is present only when field 8 of the command contains a value in the range of 0 through 199.

Table 10-14 Response 225: Verify Digital Signature

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	3	225
1	Reserved	1	0
2	Verification Indicator	1	Y or N
3	[Check Digits#]	8	0 - 9, A - F

Usage Notes

An error message that starts with <00#0708 indicates that the key slot specified in field 8 of the command is empty or is invalid.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Verify the digital signature using the public key passed in the command

The command looks like this:

```
<125#0#2#0#515#0100010001020304050607080900A0B0C0D0E0F101112131415
161718191A1B1C1D1E1F202122232425262728292A2B2C2D2E2F3031323334353
63738393A3B3C3D3E3F404142434445464748494A4B4C4D4E4F50515253545556
5758595A5B5C5D5E5F606162636465666768696A6B6C6D6E6F707172737475767
778797A7B7C7D7E7F808182838485868788898A8B8C8D8E8F9091929394959697
98999A9B9C9D9E9FA0A1A2A3A4A5A6A7A8A9AAABACADAEAFB0B1B2B3B4B5B6B7B
8B9BABBBCBDBEBFC0C1C2C3C4C5C6C7C8C9CACBCCCDCECFD0D1D2D3D4D5D6D7D8
D9DADBDCDDDEDFE0E1E2E3E4E5E6E7E8E9EAEBECEDEEEFF0F1F2F3F4F5F6F7F8F
9FAFBFCFDFEFFC5C2AF49A4550F48EBC98537A19C4D0C88790F0EC6FC9372C40C
8379ECDABB8EBB2B9DCA9397193AE306E58E8775A304AABE1CAF847B6247AC50D
CFA5CB1E5D8D7EE59D17083DCF438F18E6A24EA0C813B752732E5C9198DD9FF07
ADFA8543D5808F3E6B827B1C90EDE5CA94DFAFA85159D19E2D55DDF6497089893
77B43456352E8D181791C5E9D00046A41B0D2625ED54720EA455041161C9C827F
463B843CE646A0E4581638960AC3DD54CB4AB5C4D3E28556051AB0857F029602C
863CFADD38268D86118B99494E79A31EBC06F33215C1120AD723415AC5C53DD33
B7946FE038EFC98864CFBAE599208603083F3BF9A733C6484DC0741A17E2D40FD
FE167##7A81514F9E535A0D89068E969E5F46DBDC17830B60C768135F916D6AED
12FAB137796B7606C3DB0F027A726BF1AFE21783150F31AE97FA7F9442F6EE263
59E68F78D9164310494B22FED2085401FFDC59FE33E00F1324F22FC06F593DAA9
CACA5445BD09AA89267C5DCE69683CB2304AD443ADDAA8B73F11A324E62A7124D
2BD08CD9F718421EF95673CCE5B39E5CD252DA9050A4DED56F0624CAC74CDA4EC
D65934B7D9095B6F8A6A458A6C1C4E709ED71508B9F97B5970C5E603AB7798BC1
784FF52DF256C35D79A0B4B9EC404DCFA477EC588076F07A2F28EC4FCAD9FB41B
50481A23BA732356C607F992A65BBA085C4BB01647CA194EFA908CAE434EB71D#
1SRVE000,00030100010100AF79E1E223A5DDC8EE7599D4817FDFAD04AAA3911D
82D615C8F166E4C6391A9E5C8724628E62D9075836535642ABD14EAA681CD7EC8
```

```
A42166870FB366E0863FC2C3A3B4969B9F46F1DA0C31D1B313B404ECC73AE1567
7D13FE086CE0758890FDF7455905948D4FD827655FD96908EA4180611A2ED38D8
76CF31B23EC739F6F3F34048BFA3AEDA6090AFA205E1C19D0E3252366D3DC2C6B
ECBA6ABDDA79C899150302B3192F986AEF29674BA56A648A928A9ECB84DE2E460
0D80538811AA7475CB6B559BA6D5D3C0D0F7F91A393B30266AAD1FB27DF6BDC7A
938B64D96F9132501E5320CE0C93BD244C9B6A41172ABF9B9D927AB23313B1767
7293E9E15DB9ABB,F7929A1B295C6A00#>
```

The Network Security Processor returns the following response:

```
<225#0#Y#>
```

Generate Message Digest (Command 126)

Command 126 – This command generates a message digest using the MD5, SHA1, or SHA-256 algorithm.

This command is a premium value command, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<126#Continue#Algo#Encoding Method#Data-len#Data#
[Intermediate Result]#>
```

Response

```
<226#Continue#Message Digest or Intermediate Result#>[CRLF]
```

Calling Parameters

`126`

Field 0, the command identifier.

`Continue`

Field 1, the continuation flag. This field must contain the number zero when the value of field 2 is either 2 or 3. When field 2 contains the number 4 (SHA-256), the following values are allowed:

Value	Description
0	One block only (all data included)
1	First block (more data to be processed)
2	Intermediate block (more data to be processed)
3	Last block (no more data to be processed)

`Algo`

Field 2, the algorithm used to generate the message digest:

Value	Description
2	MD5
3	SHA1
4	SHA-256

`Encoding Method`

Field 3, this indicates how the message digest will be encoded:

Value	Description
0	No encoding
1	Basic Encoding Rules (BER) encoding

`Data-len`

Field 4, the length of data used to generate the message digest. This field can contain a value in the range of 1 through 15000.

`Data`

Field 5, the data used to generate the message digest. The length of this field will be twice the content of field 4. The data must be supplied in ASCII-hexadecimal format.

`[Intermediate Result]`

Field 6, the intermediate result from processing the previous block of data. The intermediate result is encrypted under the MFK. When field 1 contains either 2 or 3, this field must contain 64 hexadecimal characters, otherwise it must be empty.

Table 10-15 Command 126: Generate Message Digest

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	126
1	Continue	1	0 - 3
2	Algo	1	2, 3 or 4
3	Encoding	1	0 or 1
4	Data-len	1-5	1-15,000
5	Data	2-30,000	0 - 9, A - F
6	[Intermediate Result]	0, 64	0 - 9, A - F

Responding Parameters

`226`

Field 0, the response identifier.

`Continue`

Field 1, this value is the same as field 1 of the command.

`Message Digest or Intermediate Result`

Field 2, the Message Digest or Intermediate Result.

This field contains the message digest when either MD5 or SHA-1 has been specified as the message digest algorithm. When SHA-256 has been specified, this field will contain the message digest when command field 1 (Continue) contains a value of either 0 or 3.

This field contains the intermediate result when SHA-256 has been specified and command field 1 contains a value of either 1 or 2. If present, the intermediate result will contain 64 hexadecimal characters.

note The intermediate result is encrypted under the Master File Key. If multiple Network Security Processors will be used to process data continuation commands, make sure that they all have the same Master File Key.

Table 10-16 Response 226: Generate Message Digest

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	3	226
1	Continue	1	0 - 3
2	Message Digest or Intermediate Result	32 (MD5) 40 (SHA1) 64 (SHA-256) 68 (MD5, BER-encoded) 70 (SHA1, BER-encoded) 102 (SHA256, BER-encoded) 64 (SHA-256)	0-9. A - F 0 - 9, A - F

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

A SHA-1 example is provided in Generate AKB for Root Public Key (Command 122).

The following series of three command and response examples demonstrate data continuation for SHA-256.

Example 1: First block

Continuation flag = 1 (first block)

Algorithm = 4 (SHA-256)

Encoding = 0 (no encoding)

Immediate Result = empty

Data length = 448

Data =

```
2006032212234344400000020290A99762FE4D81738B0B0873744B32FD181266F
DC70E0050C07AA796BA0B15B50993A2928FA8D3F5071F3E11AD7495FC0C960DE5
5C7EF2CC8799E179968A295CB250729CAAA37196029F35461173F45F2F9E1C16D
7FD3711787DE4C5E45B46F4DD90953F3E19FA0D386BF08E408DEC6FB533CB54FA
E4CE38E228282DC5FF05669D35C4AAC98B7AE16257CCD4C2A74F7A6482DDD79B3
A8F2ED7BB8326F16F0082D4C8596B6683EC13A7C5EE99627B802C626958465D5D
D6C7BAF928C2B9C96A2A392AF44563DDE03C1AD26AAB0B16B056A0BDA4E9916F5
60D8A5B03972C77AA9B38E07275CA538EE1BE4A6C4AE7B84DCD1231FA36F938C2
9FB7B2ED982E7664146896F779DFF2B5C3A459DF634A6D79247110FC71CFF76C5
4B189A37EEE5A1544BFA724FFA94CF0BC8466513EEAB585FF4377191490C01630
5ADE31E4EED2A99C6DAFCF4FAD73B9683BE8038100658A8C090CC3B1364C29F27
7081A5F2DFC91093B465FFC5941FA2DB37AC09EF5BEE9E9A36297752B6C22B942
82652977DDB28E7C924825DA24459BF27833BC2C06C231FBFA08D317B4FEDA232
8AF8843178F300692B8757FF476E3F0C26C00C55D9C79BA2F1B
```

The command looks like this:

```
<126#1#4#0#448#2006032212234344400000020290A99762FE4D81738B0B0873
744B32FD181266FDC70E0050C07AA796BA0B15B50993A2928FA8D3F5071F3E11A
D7495FC0C960DE55C7EF2CC8799E179968A295CB250729CAAA37196029F354611
73F45F2F9E1C16D7FD3711787DE4C5E45B46F4DD90953F3E19FA0D386BF08E408
DEC6FB533CB54FAE4CE38E228282DC5FF05669D35C4AAC98B7AE16257CCD4C2A7
4F7A6482DDD79B3A8F2ED7BB8326F16F0082D4C8596B6683EC13A7C5EE99627B8
02C626958465D5DD6C7BAF928C2B9C96A2A392AF44563DDE03C1AD26AAB0B16B0
56A0BDA4E9916F560D8A5B03972C77AA9B38E07275CA538EE1BE4A6C4AE7B84DC
D1231FA36F938C29FB7B2ED982E7664146896F779DFF2B5C3A459DF634A6D7924
7110FC71CFF76C54B189A37EEE5A1544BFA724FFA94CF0BC8466513EEAB585FF4
377191490C016305ADE31E4EED2A99C6DAFCF4FAD73B9683BE8038100658A8C09
0CC3B1364C29F277081A5F2DFC91093B465FFC5941FA2DB37AC09EF5BEE9E9A36
297752B6C22B94282652977DDB28E7C924825DA24459BF27833BC2C06C231FBFA
08D317B4FEDA2328AF8843178F300692B8757FF476E3F0C26C00C55D9C79BA2F1
B##>
```

The Network Security Processor returns the following response:

```
<226#1#2EA425708581015A0410248A72B46A22D744FFC5F9F1F3A00DF90AB1C9
2C9A2A0C56BC535F5A64FF#>
```

Example 2: Intermediate block

Continuation flag = 2 (intermediate block)

Algorithm = 4 (SHA-256)

Encoding = 0 (no encoding)

Immediate Result (from example 1 response) =

```
2EA425708581015A0410248A72B46A22D744FFC5F9F1F3A00DF90AB1C92C9A2A0
C56BC535F5A64FF
```

Data length = 448

Data =

```
D58C8DA934A97689621030CB73B0946BA2AB549BC2E781DE69A7262337CBE2FB3
154021E900B96C7D867D16788526329C5FD76CA9D041D582AD2EF8425E22B1C7F
E96A4CA2C759B445D1994E995DDEEC963A0F767BDDC884AF7D36A2411A67CA0FB
7AD1C8C22C3E522527ACD32BA76C8923A5764D3873BAC40EAE7D112EF23D500AB
E5AB506ECE5FE1AA661F1D43F8DDAAD74CF2C742EE413A78CC66BDC1155130111
70306B3C1C6B29870881B35688C3EA299A5EF460C528CED12558869FC28E0832E
E2368F855104453BA96F25026B646C1340FB99D18F61A6D5E4932FD50F63CD0D7
41EC4A8899B1DEEA1C72A9F5D3CE8D813788BDF9F16CDE782C3A47EF5DA75E90A
955EF853181C81C367ACB0CA31DFFBF4BE497E868B61E7E77C851B385A5BD2BAF
23DB22BDE395FB07A128F96D92A8E44A1172493CB31444EB72A063AEA664DD510
ADF845159E1C6670FFB534802CD3AE56C8659766439659B4691FCA88436C02482
90050BE37DF0B400646B53FA52DEBDA7D94C476F17CB2830C3123B4AF383B6075
552ED80620453F65684394FC0AA0C2034A72BF09A846BE93B3B3AEB11A490AE7F
830916663770734D47D2DAB522DFC57049C77AF30B140178B18
```

The command looks like this:

```
<126#2#4#0#448#D58C8DA934A97689621030CB73B0946BA2AB549BC2E781DE69
A7262337CBE2FB3154021E900B96C7D867D16788526329C5FD76CA9D041D582AD
2EF8425E22B1C7FE96A4CA2C759B445D1994E995DDEEC963A0F767BDDC884AF7D
36A2411A67CA0FB7AD1C8C22C3E522527ACD32BA76C8923A5764D3873BAC40EAE
7D112EF23D500ABE5AB506ECE5FE1AA661F1D43F8DDAAD74CF2C742EE413A78CC
66BDC115513011170306B3C1C6B29870881B35688C3EA299A5EF460C528CED125
58869FC28E0832EE2368F855104453BA96F25026B646C1340FB99D18F61A6D5E4
932FD50F63CD0D741EC4A8899B1DEEA1C72A9F5D3CE8D813788BDF9F16CDE782C
3A47EF5DA75E90A955EF853181C81C367ACB0CA31DFFBF4BE497E868B61E7E77C
851B385A5BD2BAF23DB22BDE395FB07A128F96D92A8E44A1172493CB31444EB72
A063AEA664DD510ADF845159E1C6670FFB534802CD3AE56C8659766439659B469
1FCA88436C0248290050BE37DF0B400646B53FA52DEBDA7D94C476F17CB2830C3
123B4AF383B6075552ED80620453F65684394FC0AA0C2034A72BF09A846BE93B3
B3AEB11A490AE7F830916663770734D47D2DAB522DFC57049C77AF30B140178B1
8#2EA425708581015A0410248A72B46A22D744FFC5F9F1F3A00DF90AB1C92C9A2
A0C56BC535F5A64FF#>
```

The Network Security Processor returns the following response:

```
<226#2#093487039C850EF4F522BC732EB84A8CD4D2EA2B85E8B3D6F131CC6397
C214E0FB46DA6F66C386A9#>
```

Example 3: Last block

Continuation flag = 3 (last block)

Algorithm = 4 (SHA-256)

Encoding = 0 (no encoding)

Immediate Result (from example 2 response) =
```
093487039C850EF4F522BC732EB84A8CD4D2EA2B85E8B3D6F131CC6397C214E0F
B46DA6F66C386A9
```

Data length = 527

Data =
```
3B72F63E9CE12AC8E5FABA11891F848F9DED3800B2E9BBDC5469BCD4F15624F92
A0ECAF742EEEFBE24D922E6E444D311B11B4889A4FCDFE963ECF9312C7C468DA7
58585A5C2D722F686E6DE07A0BAB464A3F17A95BBBB6BA910BD18B5C506262F4C
2B721E1F48922A3D6C9D296421CB0854EC3D977AF411CF4AA37E90258D8413ECD
EC1025CEAEBB2C279AE8818F545A1D80901798B53EB2A6D355D7AB5EB9C66E1DC
B4D58351546A9A6E8F6465EB2B5C6BAAFC09AC390E0EB595EE142FCEA720943BA
185066D4CA59BEF5E8CB8B97FE17BEB21619AF0FFE29003FEF8D808F0281E320C
DE7721596127EE12CB5BEC37576757135B556D12040C87B21CDFDCA366F442465
9BD46448F455F51A676A6669DF75FB9C46C00522B4972A537B9FEF10BB260E54B
04076196B743E08C90AB1E081D2373A018B31BA9FF9F8AB36BBFC71027031C1B9
05AE08F19F2FFB57EA1D85ABB32959633071EE9A59922968156D51EB896253AF8
E9A31A5DBB3D3B6B1D9F0217974681A3DA12853D99E65CCA614D695C8DEEC80FE
9C1A305CFE1EB1F5C893E6A705C64803EFB9931CE1CD375AAC34A0E1E5C20F60F
3BD83FBECC78134B0AFCA04120E002BE5D5E8112DE6A336E0ABABA777AABF5796
3D9705EC0B07C9FBA45F1CD2E24356E0BD6686E50F00F8EDD69B88E8E31D0E2D6
00DFD89AB0827ECCE7788B97986B206DEF1B56ACE6BE10EBA04F0F1657981B08F
8FC0C70962FA8E
```

The command looks like this:
```
<126#3#4#0#527#3B72F63E9CE12AC8E5FABA11891F848F9DED3800B2E9BBDC54
69BCD4F15624F92A0ECAF742EEEFBE24D922E6E444D311B11B4889A4FCDFE963E
CF9312C7C468DA758585A5C2D722F686E6DE07A0BAB464A3F17A95BBBB6BA910B
D18B5C506262F4C2B721E1F48922A3D6C9D296421CB0854EC3D977AF411CF4AA3
7E90258D8413ECDEC1025CEAEBB2C279AE8818F545A1D80901798B53EB2A6D355
D7AB5EB9C66E1DCB4D58351546A9A6E8F6465EB2B5C6BAAFC09AC390E0EB595EE
142FCEA720943BA185066D4CA59BEF5E8CB8B97FE17BEB21619AF0FFE29003FEF
8D808F0281E320CDE7721596127EE12CB5BEC37576757135B556D12040C87B21C
DFDCA366F4424659BD46448F455F51A676A6669DF75FB9C46C00522B4972A537B
9FEF10BB260E54B04076196B743E08C90AB1E081D2373A018B31BA9FF9F8AB36B
BFC71027031C1B905AE08F19F2FFB57EA1D85ABB32959633071EE9A5992296815
6D51EB896253AF8E9A31A5DBB3D3B6B1D9F0217974681A3DA12853D99E65CCA61
4D695C8DEEC80FE9C1A305CFE1EB1F5C893E6A705C64803EFB9931CE1CD375AAC
34A0E1E5C20F60F3BD83FBECC78134B0AFCA04120E002BE5D5E8112DE6A336E0A
BABA777AABF57963D9705EC0B07C9FBA45F1CD2E24356E0BD6686E50F00F8EDD6
9B88E8E31D0E2D600DFD89AB0827ECCE7788B97986B206DEF1B56ACE6BE10EBA0
4F0F1657981B08F8FC0C70962FA8E#093487039C850EF4F522BC732EB84A8CD4D
2EA2B85E8B3D6F131CC6397C214E0FB46DA6F66C386A9#>
```

The Network Security Processor returns the following response:
```
<226#3#7B33603F5BF8DC373B165C69E97A72B15BF29BF908F5873A0D86520048
E58C2A#>
```

Generate AKB of Root Public Key (Command 12A)

Command 12A – This command generates an AKB of the root public key. To generate an AKB of the root public key you must send this command twice to the same Network Security Processor. In the first command, the challenge field is empty and the Network Security Processor returns a random challenge value in the response. Using the SCA, and a minimum of 2 security administrator smart cards, you must encrypt this challenge under variant 30 of the Master File Key. In the second command, you supply the same public exponent and modulus, and the encrypted challenge value. The Network Security Processor returns the AKB of the root public key.

This command is a premium value command, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<12A#Header#Public Exponent#Modulus#[Encrypted Challenge]#>
```

Response

```
<22A#Status#[AKBPUB]#>[CRLF]
```

Calling Parameters

`12A`

Field 0, the command identifier.

`Header`

Field 1, the header value must be 1RRVN000. In version 1.50 and above, this additional header is supported: 1RRVN00k.

`Public Exponent`

Field 2, the public exponent in ASCII-hexadecimal format. The value of the public exponent must be odd, cannot be 01, and must be less than the value of the modulus.

`Modulus`

Field 3, the modulus in ASCII-hexadecimal format.

`[Encrypted Challenge]`

Field 4, the challenge must be encrypted under variant 30 of the Master File Key. This field must be empty for the first command, it must contain 16 hexadecimal characters in the second command.

Table 10-17 Command 12A: Generate AKB of Root Public Key

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	12A
1	Header	8	1RRVN000, 1RRV00k
2	Public Exponent	Variable	0 - 9, A - F
3	Modulus	Variable	0 - 9, A - F
4	[Encrypted Challenge]	0, 16	0 - 9, A - F

Responding Parameters

`22A`

Field 0, the response identifier.

`Status`

Field 1, when the Network Security Processor receives the first command of the two command sequence, this field will contain a random 16 hexadecimal character challenge. When the Network Security Processor receives the second command of the two command sequence, this field will contain one of these values:

Value	Description
NO	The encrypted challenge is incorrect. If the encrypted challenge is incorrect, the stored transaction is deleted. You must repeat the two command sequence again.
OK	The encrypted challenge is correct.

[AKB$_{PUB}$]

Field 2, the root public key in AKB format. When the Network Security Processor receives the first command of the two command sequence, this field will be empty. It will also be empty if response field 1 (Status) contains the value NO.

Table 10-18 Response 22A: Generate AKB of Root Public Key

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	3	22A
1	Status	16, 2	0 - 9, A - F, OK, NO
2	[AKB$_{PUB}$]	0, Variable	Printable characters

Usage Notes

- To encrypt the challenge value, refer to section 5 of the *Atalla Secure Configuration Assistant User Guide*.
- JPS fix this. To prevent aOption 43 should be enabled

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

The public exponent: `010001`

The modulus:

```
C5C2AF49A4550F48EBC98537A19C4D0C88790F0EC6FC9372C40
C8379ECDABB8EBB2B9DCA9397193AE306E58E8775A304AABE1CAF847B6247AC50
DCFA5CB1E5D8D7EE59D17083DCF438F18E6A24EA0C813B752732E5C9198DD9FF0
7ADFA8543D5808F3E6B827B1C90EDE5CA94DFAFA85159D19E2D55DDF649708989
377B43456352E8D181791C5E9D00046A41B0D2625ED54720EA455041161C9C827
F463B843CE646A0E4581638960AC3DD54CB4AB5C4D3E28556051AB0857F029602
C863CFADD38268D86118B99494E79A31EBC06F33215C1120AD723415AC5C53DD3
3B7946FE038EFC98864CFBAE599208603083F3BF9A733C6484DC0741A17E2D40F
DFE167
```

The first command looks like this:

```
<12A#1RRVN000#010001#C5C2AF49A4550F48EBC98537A19C4D0C88790F0EC6FC
9372C40C8379ECDABB8EBB2B9DCA9397193AE306E58E8775A304AABE1CAF847B6
247AC50DCFA5CB1E5D8D7EE59D17083DCF438F18E6A24EA0C813B752732E5C919
8DD9FF07ADFA8543D5808F3E6B827B1C90EDE5CA94DFAFA85159D19E2D55DDF64
9708989377B43456352E8D181791C5E9D00046A41B0D2625ED54720EA45504116
1C9C827F463B843CE646A0E4581638960AC3DD54CB4AB5C4D3E28556051AB0857
F029602C863CFADD38268D86118B99494E79A31EBC06F33215C1120AD723415AC
5C53DD33B7946FE038EFC98864CFBAE599208603083F3BF9A733C6484DC0741A1
7E2D40FDFE167##>
```

The Network Security Processor returns a random response:

```
<22A#7562CB08290BEFBF##>
```

The challenge value encrypted under variant 30 of the MFK is: 54A6DC2879E10207

The second command looks like this:

```
<12A#1RRVN000#010001#C5C2AF49A4550F48EBC98537A19C4D0C88790F0EC6FC
9372C40C8379ECDABB8EBB2B9DCA9397193AE306E58E8775A304AABE1CAF847B6
247AC50DCFA5CB1E5D8D7EE59D17083DCF438F18E6A24EA0C813B752732E5C919
8DD9FF07ADFA8543D5808F3E6B827B1C90EDE5CA94DFAFA85159D19E2D55DDF64
9708989377B43456352E8D181791C5E9D00046A41B0D2625ED54720EA45504116
1C9C827F463B843CE646A0E4581638960AC3DD54CB4AB5C4D3E28556051AB0857
```

```
F029602C863CFADD38268D86118B99494E79A31EBC06F33215C1120AD723415AC
5C53DD33B7946FE038EFC98864CFBAE599208603083F3BF9A733C6484DC0741A1
7E2D40FDFE167#54A6DC2879E10207#>
```

The Network Security Processor returns the AKB of the root public key:

```
<22A#OK#1RRVN000,00030100010100C5C2AF49A4550F48EBC98537A19C4D0C88
790F0EC6FC9372C40C8379ECDABB8EBB2B9DCA9397193AE306E58E8775A304AAB
E1CAF847B6247AC50DCFA5CB1E5D8D7EE59D17083DCF438F18E6A24EA0C813B75
2732E5C9198DD9FF07ADFA8543D5808F3E6B827B1C90EDE5CA94DFAFA85159D19
E2D55DDF649708989377B43456352E8D181791C5E9D00046A41B0D2625ED54720
EA455041161C9C827F463B843CE646A0E4581638960AC3DD54CB4AB5C4D3E2855
6051AB0857F029602C863CFADD38268D86118B99494E79A31EBC06F33215C1120
AD723415AC5C53DD33B7946FE038EFC98864CFBAE599208603083F3BF9A733C64
84DC0741A17E2D40FDFE167,DF726815A5105618#>
```

Translate Public or Private Key AKB from MFK to PMFK (Command 12B)

Command 12B – This command translates an RSA key, in AKB format, from encryption under the Master File Key to encryption under the Pending Master File Key. The AKB header is not changed.

This command is a premium value command, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

$<12B\#AKB_{PUB}$ or $AKB_{PRV}\#>$

Response

$<22B\#AKB_{PUB}$ or $AKB_{PRV}\#>$[CRLF]

Calling Parameters

`12B`

Field 0, the command identifier.

AKB_{PUB}

Field 1, the AKB for the public key. The header for public key must be one of the following values: 1KREE000, 1SRVE000, or 1pRVE000. In version 1.50 and above, these additional headers are supported: 1kREE000, 1kREN000, 1wRVE000 or 1wRVN000.

Or

AKB_{PRV}

Field 1, the AKB for the private key. The header for the private key must be one of the following values: 1KRDE000, 1SRGE000, or 1pRGE000. In version 1.50 and above, these additional headers are supported: 1kRDE000, 1kRDN000, 1wRGE000, 1wRGN000, 1wRSE000 and 1wRSN000.

Table 10-19 Command 12B: Translate Public or Private Key AKB from MFK to PMFK

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	12B
1	AKB_{PUB}	Variable	Printable characters
	or		
1	AKB_{PRV}	Variable	Printable characters

Responding Parameters

`22B`

Field 0, the response identifier.

`AKB`$_{PUB}$

Field 1, the AKB for the public key.

Or

`AKB`$_{PRV}$

Field 1, the AKB for the private key.

Table 10-20 Response 22B: Translate Public or Private Key AKB from MFK to PMFK

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	3	22B
1	AKB$_{PUB}$	Variable	Printable characters
	or		
1	AKB$_{PRV}$	Variable	Printable characters

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Translate an AKB of a public key from MFK to pending MFK

The command looks like this:

```
<12B#1SRVE000,00030100010100AF79E1E223A5DDC8EE7599D4817FDFAD04AAA
3911D82D615C8F166E4C6391A9E5C8724628E62D9075836535642ABD14EAA681C
D7EC8A42166870FB366E0863FC2C3A3B4969B9F46F1DA0C31D1B313B404ECC73A
E15677D13FE086CE0758890FDF7455905948D4FD827655FD96908EA4180611A2E
D38D876CF31B23EC739F6F3F34048BFA3AEDA6090AFA205E1C19D0E3252366D3D
C2C6BECBA6ABDDA79C899150302B3192F986AEF29674BA56A648A928A9ECB84DE
2E4600D80538811AA7475CB6B559BA6D5D3C0D0F7F91A393B30266AAD1FB27DF6
BDC7A938B64D96F9132501E5320CE0C93BD244C9B6A41172ABF9B9D927AB23313
B17677293E9E15DB9ABB,F7929A1B295C6A00#>
```

The Network Security Processor returns the following response:

```
<22B#1SRVE000,00030100010100AF79E1E223A5DDC8EE7599D4817FDFAD04AAA
3911D82D615C8F166E4C6391A9E5C8724628E62D9075836535642ABD14EAA681C
D7EC8A42166870FB366E0863FC2C3A3B4969B9F46F1DA0C31D1B313B404ECC73A
E15677D13FE086CE0758890FDF7455905948D4FD827655FD96908EA4180611A2E
```

```
D38D876CF31B23EC739F6F3F34048BFA3AEDA6090AFA205E1C19D0E3252366D3D
C2C6BECBA6ABDDA79C899150302B3192F986AEF29674BA56A648A928A9ECB84DE
2E4600D80538811AA7475CB6B559BA6D5D3C0D0F7F91A393B30266AAD1FB27DF6
BDC7A938B64D96F9132501E5320CE0C93BD244C9B6A41172ABF9B9D927AB23313
B17677293E9E15DB9ABB,85C75ED0FABAC842#>
```

Verify or Delete RSA Key Table Entry (Command 12C)

Command 12C – This command verifies or deletes an RSA key stored in the RSA key table. When used to verify an RSA key, the check digits of the key are returned in the response.

This command is a premium value command, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<12C#Operation#Key Slot#>
```

Response

```
<22C#Available Key Slots#[Check Digits#]>[CRLF]
```

Calling Parameters

`12C`

Field 0, the command identifier.

`Operation`

Field 1, the operation type can be either "VERIFY" or "DELETE".

`Key Slot`

Field 2, the slot in the RSA key table that will be verified or deleted. When the operation type is "DELETE" and this field contains the value "ALL", every RSA key stored in the RSA key table will be deleted.

Table 10-21 Command 12C: Verify or Delete RSA Key Table Entry

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	12C
1	Operation	6	VERIFY or DELETE
2	Key Slot	1-3	0 - 199 or ALL

Responding Parameters

`22C`

Field 0, the response identifier.

```
Available Key Slots
```

Field 1, the number of empty key slots in the RSA key table.

```
[Check Digits#]
```

Field 2, the check digits of RSA key. Check digits are the leftmost 8 bytes of the SHA-1 message digest of the RSA key. This field is returned only when the operation is “VERIFY”. When the specified key slot is empty, this field will contain the text “NO KEY IN KEY SLOT”.

Table 10-22 Response 22C: Verify or Delete RSA Key Table Entry

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	3	22C
1	Available Key Slots	1-3	0 - 9
2	[Check Digits#] or NO KEY IN KEY SLOT	0, 8 or 18	0 - 9, A - F, or NO KEY IN KEY SLOT

Example

Verify the RSA key in slot 1

The command looks like this:

```
<12C#VERIFY#1#>
```

The Network Security Processor returns the following response:

```
<22C#198#8F6D7481#>
```

Delete the RSA key in slot 7

The command looks like this:

```
<12C#DELETE#7#>
```

The Network Security Processor returns the following response:

```
<22C#199#>
```

Confirm key slot 7 is empty

The command looks like this:

```
<12C#VERIFY#7#>
```

The Network Security Processor returns the following response:

```
<22C#199#NO KEY IN KEY SLOT#>
```

Generate ATM Master Key and encrypt with public key (Command 12F)

Command 12F – This command generates an odd parity 3DES ATM Master Key (MK) and encrypts it under an Encrypting PIN Pad's public key. The ATM Master Key is also returned in AKB format encrypted under the MFK.

For NDC+ applications, the clear text of the ATM Master Key is left-padded per PKCS #1, as follows, before being encrypted by the EPP public key.

```
0x00||0x02||random binary data||00||MK (8 or 16 bytes)
```

where the length of random binary data = sizeof(modulus) - sizeof(MK) - 3.

For Diebold applications, the clear text of the ATM Master Key is embedded in a key block structure provided in field 3 of the command. The rightmost 16 bytes of the key block is a placeholder and will be replaced by the clear ATM Master Key before being encrypted by the public key. The Diebold key block is padded per OAEP.

For Wincor-Nixdorf Remote Key Loading (RKL) with signatures, the ATM Master Key is padded exactly as in NDC+ application.

In version 1.32 and above, Wincor Nixdorf Remote Key Loading (RKL) using certificates is supported. The clear text of the ATM Master Key is embedded in a key block structure provided in field 3 of the command. The rightmost 16 bytes of the key block is a placeholder and will be replaced by the clear ATM Master Key before being encrypted by the public key. The key block is padded per PKCS #1, as follows, before being encrypted using the public key.

```
0x00||0x02||random binary data||00||Key Block
```

where the length of random binary data = sizeof(modulus) - sizeof(Key Block) - 3.

In version 1.46 and above, the ability to specify the header, applied to the AKB of the generated master key, can be specified, see [Master Key Header#].

This command is a premium value command, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<12F#Master Key Length#Key Block Data length#Key Block Data#
Key Slot or AKB_PUB#[Master Key Header#]>
```

Response

```
<22F#Master Key Length#AKB_MK#Master Key Check Digits#
E_RSA-PUB(MK or Key Block Data)#[Check Digits#]>[CRLF]
```

Calling Parameters

`12F`

Field 0, the command identifier.

`Master Key Length`

Field 1, the length of the ATM Master Key, MK.

Value	Description
D	2key-3DES (double-length) key.
S	1key-3DES (single-length) key.

`Key Block Data length`

Field 2, the length of the key block data. For NDC+ and Wincor Nixdorf RKL with signatures, this field must be zero. For Diebold and Wincor Nixdorf RKL using certificates, this field must contain a value in the range of 20 through 245 (the minumum length has been decreased from 21 to 20 in version 1.51). For Wincor Nixdorf RKL with certificates, this field must contain the letter P followed by the length of the key block (20 to 245). The letter P indicates PKCS #1 version 1.5 padding, instead of OAEP.

The length of the key block structure is limited by the length of the RSA key and the padding method. This command has a maximum length of 245 bytes, however smaller key sizes and OAEP will be limited to smaller key block structures. For convenience, a list of key sizes and padding schemes is provided.

Key Size in bits	Padding	Maximum Key Block Data Length (bytes)
2296 - 4096	any	245
2048	PKCS #1 v1.5	245
2048	OAEP	214
1024	PKCS #1 v1.5	117
1024	OAEP	86

`[Key Block Data]`

Field 3, the key block data in ASCII-hexadecimal format. For NDC+ and Wincor Nixdorf RKL with signatures, this field must be empty. For Diebold and Wincor Nixdorf RKL with certificates, the length of this field will be twice the length specified in field 2.

`Key Slot`

Field 4, the key slot in the RSA key table where the EPP's public key is stored. Command 121 must be used to store the public key in the RSA key table.

Or

`AKB`$_{PUB}$

Field 4, the AKB of the EPP's public key. The header for this public key must be either 1KREE000 or 1KREN000.

`[Master Key Header#]`

Field 5, this field is optional. If present, it contains the header to be applied to the AKB of the generated ATM Master Key returned in field 2 of the response. The header must be either 1kDNE000 or 1KDNE000. If this field is not present, the header 1KDNE000 will be used.

Table 10-23 Command 12F: Generate ATM Master Key and encrypt with public key

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	12F
1	Master Key Length	1	S or D
2	Key Block Data Length	1 - 3	0, P, 20 - 245
3	Key Block Data	0, or 40-490	0 - 9, A - F
4	Key Slot	1 - 3	0 - 199
Or			
4	AKB$_{PUB}$	Variable	Printable characters
5	[Master Key Header#]	0, 8	1kDNE000 or 1KDNE000

Responding Parameters

`22F`

Field 0, the response identifier.

`Master Key Length`

Field 1, the length of the ATM Master Key.

`AKB`$_{MK}$

Field 2, the DES key AKB of the generated ATM Master Key. The header for this ATM Master Key is the value specified in [Master Key Header#], or 1KDNE00 if field 5 is not present.

`Master Key Check Digits`

Field 3, the check digits of the ATM Master Key. Check digits are the first four digits that result from encrypting zeros using the ATM Master Key.

If option 88 is enabled, this field will contain six check digits.

`E`$_{\texttt{RSA-PUB}}$`(Key Block)`

Field 4, the ATM Master Key or key block encrypted by the RSA public key. The length of this field will be twice the size of the modulus (2*SOM).

`[Check Digits#]`

Field 5, the check digits of the RSA public key. Check digits are the leftmost 8 bytes of the SHA-1 message digest of the RSA key. This field is present only when field 4 of the command contains a value in the range of 0 through 199.

Table 10-24 Response 22F: Generate ATM Master Key and encrypt with public key

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	3	22F
1	Master Key Length	1	S or D
2	AKB$_{MK}$	74	Printable characters
3	Master Key Check Digits	4	0 - 9, A - F
4	E$_{RSA-PUB}$(Key Block or Master Key)	2*SOM	0 - 9, A - F
5	[Check Digits#]	0, 8	0 - 9, A - F

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

NCR - include EPP's public key in the command

The command looks like this:

```
<12F#D#0##1KREE000,00030100010100BE481D015C2FE8ADACFBCC69057DDAEE
F3DA7554A83073660AF2E624523B24710C3187AA2A2494969A282C006427AB239
0E0F5D19D249305141346F57557B5F120E30A2DBBE415DFB32C6A4B2F194450FE
25AAA56F53EA71716DDFE0C53632035E15CA65DA414C7CE4213F3E107FE8B6911
DD999222D7F2D9AB77794BAFC68255F4949171624C754268827984B47FEBEAF47
A6772938613E8C130F76753B82CA123C47603771E30E9FF86CC15642486A45E51
4E01122589377A49DA300FDBA4F2EDA9FF02DA64C6D6D8DF5CBB567C4CFBD8C19
05AF09758ED2C926628375383000E67CFD7EAC282F8866107E91B1FEBC76A5055
6CFA4C112B4AA6D9B880C0967,4767BE0ACD16F4DB#>
```

The Network Security Processor returns a response similar to this:

```
<22F#D#1KDNE000,D8897E87628DBBF1CB8D9362168604E57235127556850008,
414EC14970D29050#91D1#200BD168BCA7F7D6E8A8CD0A158D6AB6CA04CF16A20
F7FA43F9652EF6B2EB29057A294DE123AEE9AAD8BB641DF60C4D91504C1AA8C4A
```

```
E2A413F9FE9F7BAEEAD8156D4011E26ED7E5302D556FA9474121706A4C96807EC
82CED4787B3D999B20781A36F9DBB2EACADF9E70E05CB4FFA2A1C958E7B3F5832
AACBCC9EC8C7A8975B184881EC740FF2222C65E1285A7D3698AA186D94836730D
BE37D8EF9DAC6E6A341E9F4E5AFE72532ED1CD1AEB15E8C8A47A32EE0B8F47092
4BFDC00D70F29826E4BC893439A43C8BD911745F9E96BA580AD83D03B85A59ACD
C91DDCB106BB3E2D2EBBF0C0AD32DAB015C5CA9BA8BE1C6387361F8A30E089D8F
2BC6C26115BC6F#>
```

Diebold - use EPP’s public key stored in slot 15

The command looks like this:

```
<12F#D#36#12345678901234567890123456789012345678901234567890123456789012345
67890123456789012#15#>
```

The Network Security Processor returns a response similar to this:

```
<22F#D#1KDNE000,3870C4D0F8D9B5E18F96F726F680B80A09A18375D9EAA30A,
0CABC1D50A57D84A#09D77D#A6784C8DDDA26123B0FB63F3AB4E0156C2C8CBC2D
8E450A9369D3B4A7FEAD5B4FD703620A6C2600D94175F12C3473394D0BD7F3F04
55FAFA661660C32A26F8F34F590F2DDCCBE0B8F72F24651E81DBF28A4168B1E3B
3AC06B6D95201B8710B5C824685F003BEB3C70DE8DDEC88FD3469161DC0DDDBD0
663EA908F0B9D88C327E87FDAFF961D3644F19E51F607C98F427A33577ADD690B
15ED700E9DA8CBBC9554732E8518FA445E9BAB439949F582858AD1E0458A1B51C
261822801383C4DEEF86F9AA1C41835152ED3F40B98020E6BC25FD3C47C73D982
A88FD351A44633E86668962D47A582344B32E49981FF3FDA237D1DC638502DCEE
AEEE4D265E3B53EF#E66C7C60#>
```

Wincor Nixdorf RKL with certificates

The command looks like this:

```
<12F#D#P32#6EEA924FBA8EC827637BEAF0DB877CF01C3EE094E115173211F200
56F2EC725B#1KREE000,0001030100DE486CF7BD8E75B64F702A1CC7D209373C3
9E569B4F3D36636C9C7342410E3C97F2F1ABF630AB827D8CED14F77CDBF609201
4E642B82C02CBBCE52C76CA781BA694B5F9C114F5FBA67504DF3BA51AFA8D7305
2B9B042C01E6C099FFB4230AF9E6F157A374FF6766F0A34FF00DE25B8AE65C884
6A247F102FC0FE1EC66A489BF3A97A2940CF185E7FEE15F4750E2C987C514DEE8
3E1790B940AB9DEF5374D77F1C4ABCD14187EAFB2B3144AB6D0FA86148444AA56
55659C4B3CEA3073CA9C6E8B3D26F37382FD1FE61132D653AB12854FA61B3D9C8
6188DA964C000296B644566126351620BF6F27AAF85D27703315071BAF354A07F
87B2E79350DB6788401EC3,5F19153D2C096625#>
```

The Network Security Processor returns a response similar to this:

```
<22F#D#1KDNE000,88481AC413B828AAE8C9466EFCDDC21B804285D0CA9001DD,
B6A6894FD8978951#73C6#38F312A8CD0F413BF33CC2C7EAAA8E9579D896C52DE
E4C16E8A0DA7CE3B7206EA4DA0A6D051EC80DF9D1064E093BDD5221555418CB11
9562F99D4B5CFBA76461ED8767249A57EEE0CAC384C4405B80049C9FE4E43E47E
2702EB2D83B2B0E93A1B96B87C835CEAD26DB17361C544B99268F8028FB2709C3
6F393CA5E8EA563E77403817DAF3DE93A2522537B76504EB310E18E9ECA405738
A14F4204E4D4DBCD630FCA5C97BAFB1F1B06B34AB1D0E6220D50FAA7C71C80C97
79BA47F2108EAD573B5A1FEC2DF4B5E6FCB151C47BF421C5DE29500AC3BE6C139
0A4285A2A6309850255C99CA9F8B5D34789773516F70C79653A0E79CC681B86C3
52624919208D3A#>
```

Export RSA Private Key in ECB encrypted format (Command 135)

Command 135 – This command will decrypt an AKB formatted RSA private key, which is encrypted under the Network Security Processor's MFK, and then encrypt each RSA private key component under a Key Exchange Key using 3DES ECB. Use this command to export an RSA private key.

This command is a premium value command. It is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<135#AKB_PRV#Header,E_MFK.E(KEK),MAC#Reserved#>
```

Response

```
<235#e#n#E_KEK(d)#E_KEK(p)#E_KEK(q)#E_KEK(dp)#E_KEK(dq)#E_KEK(u)#
Check Digits#>[CRLF]
```

Calling Parameters

`135`

Field 0, the command identifier.

$\texttt{AKB}_{\texttt{PRV}}$

Field 1, the AKB of the private key to be exported. The header must be 1pRGE000.

$\texttt{Header,E}_{\texttt{MFK.E}}\texttt{(KEK),MAC}$

Field 2, the AKB of the Key Exchange Key that will be used to encrypt the components of the private key. This key must be either a 2key- or 3key-3DES key. The following headers are supported: 1pRXN000 or 1pRXE000.

`Reserved`

Field 3, this field must be empty, it is reserved for future use.

Table 10-25 Command 135: Export RSA private key in ECB encrypted format

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	135
1	AKB_{PRV}	variable	Printable characters
2	$\text{Header,E}_{\text{MFK.E}}\text{(KEK),MAC}$	74	Printable characters
3	Reserved	0	none

Responding Parameters

`235`

Field 0, the response identifier.

`e`

Field 1, the RSA public exponent in clear form.

`n`

Field 2, the RSA public modulus in clear form.

$E_{KEK}(d)$

Field 3, the 3DES ECB encrypted form of the RSA private exponent.

$E_{KEK}(p)$

Field 4, the 3DES ECB encrypted form of prime p.

$E_{KEK}(q)$

Field 5, the 3DES ECB encrypted form of prime q.

$E_{KEK}(dp)$

Field 6, the 3DES ECB encrypted form of exponent 1, d mod (p-1).

$E_{KEK}(dq)$

Field 7, the 3DES ECB encrypted form of exponent 2, d mod (q-1).

$E_{KEK}(u)$

Field 8, the 3DES ECB encrypted form of the Chinese Remainder Theorem coefficient, (inverse of q) mod p.

`Check Digits`

Field 9, the RSA private key check digits. Check digits are the leftmost 8 bytes of the SHA-1 message digest of the RSA key.

Table 10-26 Response 235: Export RSA private key in ECB encrypted format

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	3	235
1	e	1-1024	0 - 9, A - F
2	n	192-1024	0 - 9, A - F
3	$E_{KEK}(d)$	variable	0 - 9, A - F

Table 10-26 Response 235: Export RSA private key in ECB encrypted format (continued)

Field	Contents	Length (bytes)	Legal Characters
4	$E_{KEK}(p)$	112-528	0 - 9, A - F
5	$E_{KEK}(q)$	112-528	0 - 9, A - F
6	$E_{KEK}(dp)$	variable	0 - 9, A - F
7	$E_{KEK}(dq)$	variable	0 - 9, A - F
8	$E_{KEK}(u)$	variable	0 - 9, A - F
9	Check Digits	8	0 - 9, A - F

Usage Notes

- Use of this format is not recommended, as it does not provide integrity protection for the private key.
- Your node and the remote node must have the same Key Exchange Key value.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

The command looks like this:

```
<135#1pRGE000,DDC2AF3F360A8920A39123A2BD49012A77C4D382DF2F45CAB67
354F75ED64C72FAE09CEBE577AE4485AB105394DE1728BC358802B841E3FA0516
885DDA90073D4668165BCA6F76011374B59928903ED1E0A39A3B001F7B6E8C65B
4F0C7D7DEFF220FE441A4BAAF1B906391FD1582A5EA59E875AD33739C61888416
3F12FF148053C40A968E5714295B8849D09B77C0B555BE4A5B34F18F732F45606
3D7AFF558DEDA1A843B6034A5A86EACC85AC6C61C96A223AB43ABFAA7DA0A07A3
27233AEBC15F1621159DB55C2A4CC18040E52B9CFA71D567760B7AA978A4C5B1A
D9E913980B9D6B965E466542CAF778113975A3CD37EF803BB166CB004252579EF
BDFF8F74249A313B386FC9EE53DB251356D4442B1FFB622A9296156C0FBDE5554
9DC470A8F4DC3B82664703838319 9F6AACC3F70BF8C6EFD07C651CDE851119DCF
8FF17F70FCF9AE29414F216C399EA65DE0C21C8AF7AF93C489934F2772AABCD76
7E941A5AF2BB94E07C660363D747AD95D6303270159E9026B25FC7EA5BE4525C4
D15D49717E612A2AD3BF7820848F4F208331C2483EB4784ADE472CEB0009AE2F0
5DC5A48CE37BBBC54C2EA5BC4E740A352040FAF0904152A5E11B88E8F9EB3B42D
76390FA426A22A6CFA3D3E79BB4998141EDA303E96A38F7D2B1D98D2AF88A9856
84385FC7E1224CB5076CC6226D9D9D89191797A55477493169524CDE38D6B197B
B73F34C0781C9ECFC0D82B22C32BA9302731BC63D137B2A302C78E1D3BDABC829
A6E1A68680205024F5143E78BE579129240035D9D9395CCB66847F382A6833CA5
EB8DCCA6A50DF4D1C9E3182A676605B9C95C0D2B174C,3E96DCC4769F0C23#1pR
XE000,5A292C546F517B39FD91ACF82D34B27BC9180333E544F11E,BEEB9C28C1
E36EB3##>
```

The Network Security Processor returns a response similar to this:

```
<235#010001#D85851316ED742172133AB631AAEABADB5FDC8309195897C113E7
424D2C5269FEC4836635AFA11DA844A2AE8A41445D66D58B326C929E0DDE9D598
3644C78DF7BE7CC9C2A34C79DB4E82C4A043956D15D348B2D6A7137BBA39F9A7B
D78AE8E14BFE23E006D334146BF46E6ADDC2198FF711FC2A2E1E8F13FAA86243B
64135B03#485A9D6E979475752FD5413EF25BAE5C165550C4ABD01F6D6C47AB15
B48642401AB47911B7BC06A5877170F06447BFEA7AF6937212498C8AD5AF8E808
58AA2DDD7961C4AB4B951E59F19FB5D31C4BFC5017D781369D6C051FE68628001
5FA93927750EFC5F8F6E80DD4901AC71C076D4914BF28E622D8231F5F7A085BE9
6FAFDD286EB72912D7D0C#40438AC3A50E0BC55CA7E057881C1D8A468B771FEEB
7736B68CEF5E80CDBBB5F538FDC2C720FF57D3CC4081B2FC110EF0E131B93F88F
103FEDECBF8F481151F7D286EB72912D7D0C#EC027FCEDB31AC8994800F19C769
2C37E98D9836C7D26B9824FF30AB78B851823AC6D4677EA5F38E174FF4DB33352
EF7F962F8277982B179EA0E6A301356D089D286EB72912D7D0C#865B05E216C5F
399D33B96E3EE581DCE7ACE29AE68FFAD77FBA657E88063E4C7C08146836084F2
15766A6A3CEF2B8CDD4ABAC7408F72FA60E4E0DDE7FA1DA375D286EB72912D7D0
C#624F867696A1359F1DEF0E9ECD03507E2EC194B3570FE2A970D690FA2334C1D
052D87438E3EEE98024897F444E3954D7E2F1827C3EC23B77C742F2FEC9C3F92B
D286EB72912D7D0C#7708309C60D00476AA8752D7A3284DD0A07CE1F42C2DC0F5
B0A0D9C1A7BD3602CCBB215EAA68B5DB87799B64EE535E99FC72BD4B69147B286
A2EF0E1F836FEFDD286EB72912D7D0C#73A7721A#>
```

Generate a TR-34 Key Block (Command 136)

Command 136 – This command generates a random odd parity 3DES key, an initialization vector, and an odd parity 3key-3DES ephemeral key. The generated 3DES key is formatted into a TR-34 key block. It is also returned in AKB format encrypted under the MFK. The ephemeral key is encrypted under an RSA public key. The TR-34 key block is 3DES-CBC encrypted under the ephemeral key using the generated initialization vector.

In addition, this command creates a signing token. The signing token binds this command and command 139 together. Without this binding, a malicious insider could encrypt a known key under the Key Receiving Device's public key, and then use command 139 to sign the known key blob. Command 139 verifies the signing token before generating the signature. The signing token is single use. It should not need to be stored or translated to the pending MFK.

For more information on using this command, see Command Usage Scenario for sending TR-34 messages.

This command is a premium value command, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<136#Key Length#Key Block Header#Reserved#BE Data#
Ephemeral Algorithm#Hash Algorithm#AKB_PUB#>
```

Response

```
<236#Header,E_MFK.E(K_N),MAC#K_N Check Digits#RSA(Ke)#IV#E_Ke(BE)#
Header,Signing Token,MAC#>[CRLF]
```

Calling Parameters

`136`

Field 0, the command identifier.

`Key Length`

Field 1, the length of the 3DES key to be generated.

Value	Description
D	2key-3DES (double-length) key
T	3key-3DES (triple-length) key

`Key Block Header`

Field 2, the TR-34 Key Block Header to be applied to the generated 3DES key. This field must contain a 16 ASCII character value. Refer to the *X9 TR-31 2010, Interoperable Secure Key Exchange Key Block Specification for Symmetric Algorithms* for valid header values. The Usage Notes contains information on how the Network Security Processor will map the TR-34 Key Block Header to the AKB Header.

`Reserved`

Field 3, this field must be empty, it is reserved for future use.

`BE Data`

Field 4, this field must contain the BE data. This field must contain an even number of hexadecimal characters, the minimum length is 10 and the maximum length is 400 (increased from 300 in version 1.51). The BE data is DER encoded, it consists of the Version concatenated with the ID_{KDH_CRED}. The underlined and bolded text in the following line is the BE data.

(**<u>Version||ID$_{KDH\_CRED}$</u>**||K_n|| KBH)

Using the sample from section B.2.2.2.1 of the *X9 TR34-2012 Interoperable Method for Distribution of Symmetric Keys using Asymmetric Techniques: Part 1 – Using Factoring-Based Public Key Cryptography Unilateral Key Transport*, the BE data would be:

```
020101304A3041310B30090603550406130255533115301306035504 0A130C54523
3342053616D706C6573311B301906035504031312545233342053616D706C652043
41204B444802053400000006
```

`Ephemeral Algorithm`

Field 5, this field must contain the letter D, which indicates 3key-3DES CBC.

`Hash Algorithm`

Field 6, this field must contain the number 4, which indicates SHA-256.

`AKB`$_{PUB}$

Field 7, this field contains the recipient's public key. The header for this public key must be 1kREE000. The public key modulus be 2048 bits or greater, and the public exponent must be 65537 (0x010001). This AKB_{PUB} is created using command 123.

Table 10-27 Command 136: Generate TR-34 Key Block

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	136
1	Key Length	1	D, T
2	Key Block Header	16	0 - 9, A - F, a - z

Table 10-27 Command 136: Generate TR-34 Key Block (continued)

Field	Contents	Length (bytes)	Legal Characters
3	Reserved	0	none
4	BE Data	10 - 400	0 - 9, A - F
5	Ephemeral Algorithm	1	D
6	Hash Algorithm	1	4
7	AKB_{PUB}	variable	Printable ASCII

Responding Parameters

`236`

Field 0, the response identifier.

`Header,`$E_{MFK.E}(K_N)$`,MAC`

Field 1, the AKB of the generated 3DES key. The header for this AKB depends on the value specified in field 2 of the command.

K_N `Check Digits`

Field 2, the check digits of the generated 3DES key. Check digits are the first six hexadecimal characters that result from encrypting zeros using Kn.

`RSA(Ke)`

Field 3, the ephemeral key encrypted under RSA public key using OAEP_SHA256.

`IV`

Field 4, the generated Initialization Vector used the 3DES-CBC encrypt BE. This field will contain 16 hexadecimal characters.

E_{Ke}`(BE)`

Field 5, the key block BE, encrypted under the ephemeral key.

`Header,Signing Token,MAC`

Field 6, the signing token in AKB format. The header will be 1bRVN000. It must be supplied as field 8 in command 139.

Table 10-28 Response 236: Generate a TR-34 Key Block

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	3	236
1	Header,$E_{MFK.E}(K_N)$, MAC	74	Printable ASCII
2	K_N Check Digits	6	0 - 9, A - F
3	RSA(Ke)	variable	0 - 9, A - F
4	IV	16	0 - 9, A - F
5	E_{Ke}(BE)	variable	0 - 9, A - F
6	Signing Token	110	Printable ASCII

Usage Notes

The following table shows the list of supported TR-34 headers, and the relationship between TR-34 header bytes and the AKB header.

- The first character of the TR-34 header can be A, B, or C without changing the meaning of the header. The table uses '$' to indicate A, B, or C.
- The value ‘xxxx’ in the TR-34 header bytes 1-4 is the length of the key block. The Network Security Processor will not check this value.
- The value ‘yy’ in the TR-34 header bytes 9-10 can have any alphanumeric value, except the first character cannot be a 'c'.
- The value 'N' in the TR-34 header byte 11 means the key will not be exportable after this initial message.
- Optional blocks in the key header are not supported.
- The mode of use will be opposite for the TR-34 and the AKB header. For example, an encrypt-only KEK for an ATM would be a decrypt only KEK for the Network Security Processor.

TR-34 Header	AKB Header	Description
$xxxxK0TByyE0000	1KDNE000	KEK, exportable
$xxxxK0TByyS0000	1KDNE000	KEK- sensitive exportability
$xxxxK0TByyN0000	1KDNN000	KEK- non-exportable
$xxxxK0TDyyE0000	1KDEE000	KEK- encrypt only for NSP, decrypt only for recipient, exportable

TR-34 Header	AKB Header	Description
$xxxxK0TDyyS0000	1KDEE000	KEK- encrypt only for NSP, decrypt only for recipient, sensitive exportability
$xxxxK0TDyyN0000	1KDEN000	KEK- encrypt only for NSP, decrypt only for recipient, non-exportable
$xxxxK0TEyyE0000	1KDDE000	KEK- decrypt only for NSP, encrypt only for recipient, exportable
$xxxxK0TEyyS0000	1KDDE000	KEK- decrypt only for NSP, encrypt only for recipient, sensitive exportability
$xxxxK0TEyyN0000	1KDDN000	KEK-decrypt only for NSP, encrypt only for recipient, non-exportable
$xxxxK1TByyE0000	1kDNE000	TR31 Key Block Protection Key, exportable
$xxxxK1TByyS0000	1kDNE000	TR31 Key Block Protection Key, sensitive exportability
$xxxxK1TByyN0000	1kDNN000	TR31 Key Block Protection Key, non-exportable
$xxxxK1TDyyE0000	1kDEE000	TR31 Key Block Protection Key - encrypt only for NSP, decrypt only for recipient, exportable
$xxxxK1TDyyS0000	1kDEE000	TR31 Key Block Protection Key - encrypt only for NSP, decrypt only for recipient, sensitive exportability
$xxxxK1TDyyN0000	1kDEN000	TR31 Key Block Protection Key - encrypt only for NSP, decrypt only for recipient, non-exportable
$xxxxK1TEyyE0000	1kDDE000	TR31 Key Block Protection Key - decrypt only for NSP, encrypt only for recipient, exportable
$xxxxK1TEyyS0000	1kDDE000	TR31 Key Block Protection Key - decrypt only for NSP, encrypt only for recipient, sensitive exportability
$xxxxK1TEyyN0000	1kDDN000	TR31 Key Block Protection Key - decrypt only for NSP, encrypt only for recipient, non-exportable

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

The command looks like this:

```
<136#D#A0256K0TB00E0000##020101304A3041310B300906035504061302 5553
```

```
<136#D#A0256K0TB00E0000##020101304A3041310B30090603550406130255 53
```

```
<136#D#A0256K0TB00E0000##020101304A3041310B3009060355040613025553
3115301306035504 0A130C5452333420536 16D706C6573311B301906035504031
```

```
<136#D#A0256K0TB00E0000##020101304A3041310B3009060355040613025553
311530130603550 40A130C545233342053616D706C6573311B301906035504031
```

```
<136#D#A0256K0TB00E0000##020101304A3041310B3009060355040613025553
31153013060355040A130C545233342053616D706C6573311B301906035504031
312545233342053616D706C65204341204B444802053400000006#D#4#1kREE00
0,00030100010100CF888CCC2A70199C89DB5ECEBBDC5C126767204C6781D8D1C
5A79FFCE5032665AD1596F91B82F315FFC1A7F2FADB6649379D83B63A5ECCD8F2
E8A6870F9FD0B439538E1D366BB59D2D5FA6EEE8171B8811F0E4ED956A52069B8
630209C8AA65B2879FFE3015C6D8C6DE2071AADB7FF36574F966CDABB5311E1CB
EE2EBE5E141156D0EDF0E24A4D5082E52F48DB0D5BF9A4B39BABF3E832070814B
733B392EF6232BA77F94A20719BD6D8F50DC251B78CE0BB2EFCEFBD6028FD4AEB
```

```
3592A2010D0D5A1C721A2AD850B672B9826B57B1E5431BE8B791EF60A14E1152F
00E4A33E1E7958D2CC3C3E5B40D03CD1319C2121F2E64B2AFE3C2B708092D0BD0
1DA9D50B,0269B31AB0D6F30A#>
```

The Network Security Processor returns a response similar to this:

```
<236#1KDNE000,98F667F6292F7A367C9762FBEE20348BDBCE1B92DB0E1079,9B
BE924794E0396B#814DE0#3DA98910DA88EDA70D2F969D80FC7418FA1BC607E63
D232B041B01A8ACA63FD465B1E121458075656C9DA56DE2120F3E2D72BEC59F0D
FCD419333AD1F72C672B73EF7400AF832113848C9CA9C48AE129964A881DEC64A
C0524A8391D757E635DAAE91CC8702EAEEEFCF5372B27843ED5CFFC23C381E013
3C17379BCB05EC33B9AA2A9D0ADBFDE5ED1FEA10AEBCFDE26105472B93B0BC7CE
268674E0DC947FE1DD4536D31337856560BE190ED95A5F406BFA1E8B9D8A12FD3
45A1E54F644680DF1AB35AA91AA22C02A5450B472BDE58DD0332F393B5840B7D4
5AF332193D50B109E624AE4C98622F924BC7C48D30BC862D9996C3D21EF8C9A45
6A1657C9376C2C#2198B53DACC36011#746AFB855BF9B6AC3BC844AE8705BD051
6BD969F5BCC44E113271AB551ADF8F79B4DE30EC07FB01D326EC7EF9063FB762F
520EE30B880EC5C0D2D313ED90F6EC3BB442BD93385F44CD27607B93A395F6954
55B25720BB8E5FA1E3A5C2E09627EFDAA4EA770AA03349041FF8EE697289D2DB4
EDCED956EACBADB1F97410BA1830FCCD23901F5986A6#1bRVN000,00060100000
8008800203DCC9C7019CCB93F6AD3B1B57F9DC720418E19F9D65611911F412C27
FBE44F16,DF04823151AB6177#>
```

Sign TR-34 Message (Command 139)

Command 139 – This command is used to sign TR-34 messages. The Network Security Processor can sign messages that contain a TR-34 key block, and also messages that do not, such as rebind or unbind messages.

This command should only be used for signing TR-34 messages. Command 124 should be used for any other signing applications.

For more information on using this command, see Command Usage Scenario for sending TR-34 messages.

This command is a premium value command, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<139#Hash Algorithm#Signing Algorithm#To Be Signed Data#
[Enveloped Data]#[Ephemeral Key Offset]#[IV Offset]#[BE Offset]#
[Signing Token]#AKB_PRV#>
```

Response

```
<239#Input Validity Indicator#[Signature]#>[CRLF]
```

Calling Parameters

`139`

Field 0, the command identifier.

`Hash Algorithm`

Field 1, this field must contain the number 5, which indicates SHA-256.

`Signing Algorithm`

Field 2, this field must contain the number 1, which indicates PKCS #1 version 1.5 signatures with the BER encoded hash algorithm identifier.

`To Be Signed Data`

Field 3, the data to be hashed and signed. This field must contain an even number of hexadecimal characters, the minimum length is 2 and the maximum length is 10,000.

`[Enveloped Data]`

Field 4, this field contains the enveloped data. It must be empty when the TR-34 message does not contain a key token. When the TR-34 message contains a key token, this field contains the enveloped data contents that are to be protected by the signature. Do not include any tag or length that should not be hashed. This field can contain an even number of hexadecimal characters, the minimum length is 2 and the maximum length is 5,000.

`[Ephemeral Key Offset]`

Field 5, this field contains the offset location of the RSA-encrypted ephemeral key, Ke. This field must be empty if field 4 is empty. Otherwise, this field can contain a decimal value in the range of 0000 - 9999.

`[IV Offset]`

Field 6, this field contains the offset location of the IV used to encrypt BE. This field must be empty if field 4 is empty. Otherwise, this field can contain a decimal value in the range of 0000 - 9999.

`[BE Offset]`

Field 7, this field contains the offset location of the encrypted BE value. This field must be empty if field 4 is empty. Otherwise, this field can contain a decimal value in the range of 0000 - 9999.

`[Signing Token]`

Field 8, this field contains the signing token create by command 136 when the TR34 key block was generated. This field must be empty if field 4 is empty. Otherwise, this field can contain an AKB with header 1bRVN000.

$\mathrm{AKB}_{\mathrm{PRV}}$

Field 9, this field contains the private key that will sign the message. The header for this private key must be either 1wRGE000 or 1wRGN000. The private key must be 2048 bits or greater, and the public exponent must be 65537 (0x010001).

Table 10-29 Command 139: Sign TR-34 Message

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	139
1	Hash Algorithm	1	5
2	Signing Algorithm	1	1
3	To Be Signed Data	2 - 10000	0 - 9, A - F
4	[Enveloped Data]	0 - 5000	0 - 9, A - F

Table 10-29 Command 139: Sign TR-34 Message (continued)

Field	Contents	Length (bytes)	Legal Characters
5	[Ephemeral Key Offset]	0 - 4	0 - 9
6	[IV Offset]	0 - 4	0 - 9
7	[BE Offset]	0 - 4	0 - 9
8	[Signing Token]	0, 110	Printable ASCII
7	AKB_{PRV}	variable	Printable ASCII

Responding Parameters

`239`

Field 0, the response identifier.

`Input Validity Indicator`

Field 1, this field indicates if the input data fields are being used correctly.

Indicator	Description
Y	All inputs are valid, signature computed.
KT	To be signed data contains a key token/enveloped data and at least one of the fields 4-8 are empty.
MD	The enveloped data field does not match the message digest in the to be signed field.
ST	The RSA encrypted block, IV, or encrypted data field do not match the signing token. Either the wrong signing token was used, or one of the offsets in fields 5-7 were incorrect.

`[Signature]`

Field 2, the digital signature. This field will be empty if field 1 does not contain 'Y'.

Table 10-30 Response 239: Sign TR-34 Message

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	3	239
1	Input Validity Indicator	1-2	Y, KT, MD, ST
2	[Signature]	0, 2*SOM	0 - 9, A - F

Usage Notes

Messages that do not contain a key block are signed much like in command 124. The host application builds the to-be-signed data, and then sends it in this command for hashing and signing.

- Per section 9.3 of PKCS #7, when hashing the to be signed data, (field 3), which starts with the IMPLICIT [0] tag, 0xA0, the 0xA0 must be converted to 0x31. For example, using the example data supplied in the KDH Rebind Token shown is section B.11 of the *X9 TR34-2012 Interoperable Method for Distribution of Symmetric Keys using Asymmetric Techniques: Part 1 – Using Factoring-Based Public Key Cryptography Unilateral Key Transport*, the converted data would be as follows:
  3165301806092A864886F70D010903310B06092A864886F70D0107023018060A2A
  864886F70D01091903310A04087DEA1C00894E246A302F06092A864886F70D010
  9043122042091734FC741020043061596D961F12810BBF38F282E8B8A8A1201EC1438
  7D0990

OnlyTR34 signing keys may be used in this command, they may not be used in command 124. The difference between command 124 and this command are:

- This command always uses "BER encoding", the hash identifier is included in the signature calculation.
- This command checks the data to be signed to see if it includes a TR-34 encrypted key block. If a key block is present, the command will only sign the message if the enveloped data is authenticated with a signing token generated by command 136. This command will sign other messages (e.g. bind or unbind) without a signing token.
- This command does not support a pre-computed hash.
- This command does not support deprecated hashes (MD5 and SHA1).

Messages that contain a key block require additional information to verify that the key block was created by command 136. This requires the following additional input fields:

- The signing token created when command 136 created the encrypted fields encoded in this message.
- The enveloped data. Include only the data that is hashed for inclusion in the signed attributes.
- The offset within the enveloped data where the octet string containing the ephemeral key encrypted by RSA is located. This encrypted key offset value should be the character of the string within the enveloped data (the first character is position 1), not the location of the DER encoded tag or length.
- The offset within the enveloped data of the IV. The offset value should be the character of the string within the enveloped data (the first character is position 1), not the location of the DER encoded tag or length.

- The offset within the enveloped data of the encrypted BE. The offset value should be the character of the string within the enveloped data (the first character is position 1), not the location of the DER encoded tag or length.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Sign message which does not contain a key block

- To be signed data:
  3165301806092A864886F70D010903310B06092A864886F70D0107023018060A2A864886F70D0109
  1903310A04087DEA1C00894E246A302F06092A864886F70D01090431220420917 34FC74102004306
  1596D961F12810BBF38F282E8B8A8A1201EC14387D0990

- Private key:
  1wRGE000,8D103BB1CEDFB183FD570F38ECC308D092C98034263A637984DB6DF68D55C28A1BF06
  0679F14EB6995FF3252C461BAAFE521A609C6E96389163B83557B84DBDE02EAB4287FB5B4950E013
  FFE714BE61ECD641AC006622626CB177D4B803706EFE821724ADCE0363560563E9BC4AF6ACB0056
  6880F995BB850557915DBD1BEE2EFB39BB71126318B32B2DD6BF2EFDA5E79612E35D0DF308B6980B
  8C6E0063E796DAA81878DBEF7BE1707FF351A0D32C59572658847C8D35A5284C86E89EB15E9208C
  BA7BD7C8E1AD90BEE578FA2CDAD61F607B305FE777723D58AF3C143086AF7A889BDD5BC9642C71
  D7B736BEB20C6BB3D5731F917DF3F20988AE743EAC90B668AF95810FBEC41DBB5A5BDDE16D1C625
  1BD3FC616E9B9B3083C6DCC933B07E2B2298A7C57FD7703E057C0200F2F56FC46F93D184A611E12F1
  94545FD00375F4ED2926B57681AD65FEE70DCCDBF298E12AFF499D0B6E2682B24AF2554A3281A58
  5B89667FE73A53EA3116E0278D7410A538400BC9D0FA5E0FFF63D36D642B0701F5DD0AE0D000611
  2F7CE56694D020EA7E25C65266118A5193B587AA6DCC431D29F484CAE7FF5B7A7E5074C87A85A81
  A6CA69F0593BBCFB251BE138D20AC94D76F376B0C7C579BA8D91472DBC167D57BCB74EAC93F936
  D4E4F8672D2BA9021DD574F42FED8DC6E63617A348BD080C9F032EEA23C07D4444AB2E69801F93
  A7CC06F444D9AB5F77DEA318A26720BB57DF6C5F940577B856309FE33E488C229B6A119582285601
  26D13D11E0B67CE2D5665B466D2B1CF27649F92203D6E69BD30395A68693121241E8BE96C70594C44
  EDB22E6DA182812C882E51F027B9A79174CF7F38D07198033372FE7D737EB61EF5C36C9A66C7D66BB
  EC180C552672660255B87F9F91DFC7E14185A2EFC65C201481599770A54849DAC3197D2677F7E96F4
  A0117BCD1B42CD4447234A71AEED5554AB166BA23156CBE5E61C8FBB3A0598C719C6F296D20D5CC
  D381897B4660DCE48A7E5592CB644C6F5E03594D7248F6877AB90B3A3846EAADB6CA3D17F42A54
  B6F21D33CBEC426B9A9FB2B2D5FA44894628BD1F0AE339DD341FE547F73C69F7C2E6F292BC1E40D
  50D417622C918CE587AF5F4B945A58BE4CDF8F169D150A8E024173B04B99A135BD3AD2DE35640E5
  FFAACFD4C6AF98581780A315AF80BC026155F0BEED947AD52E05BC50BB4A20D96B690496C2DF8
  7691291AD8CBC8B415C8C51F9D53819C6A5FF5AFDDB2F78469511E5F9E9FD95C868A8D0D5EDEDA6
  3EF940D77436E7F8058A67BF721AC1F903FAF693A33B8C616A2BE5BE3AC492BFE8417E0AEA556D3
  85F9132E12BE720FB514F46BE540D68F9C2F028D6CBCF1802EC90174E90C4B985D6FD0AA4EE861A
  73AE1BBF6BFF82AF0E88B126EC102CFC5D9CC870061501FEF40E9BF4CC07B430A29968BD1165820
  52AD69D99270BD1681D82D19679AC5B08882BBDA109BA893AE2A9CF3398FEC8FBBF089EF192B23
  C0DEFA09A32016BA013A47C08216BC38F4B3EBD85567B561891481B02908AC74A42E9775ECEE2B7
  FE1B3B5C757082226518EE7092DB6D5C1F7DE0A758B4503E7CA2E70956E0FE1C21094D9C0F012F48
  53615AECDC5B86267D96E7F6546252DC8EBDB3857939F55878993CA3BF2A598E3E,9A7A6923D2994
  7DB

  The command looks like this:

```
<139#5#1#3165301806092A864886F70D010903310B06092A864886F70D010702
3018060A2A864886F70D01091903310A04087DEA1C00894E246A302F06092A864
886F70D01090431220420917 34FC741020043061596D961F12810BBF38F282E8B
8A8A1201EC14387D0990######1wRGE000,8D103BB1CEDFB183FD570F38ECC308
D092C98034263A637984DB6DF68D55C28A1BF060679F14EB6995FF3252C461BAA
FE521A609C6E96389163B83557B84DBDE02EAB4287FB5B4950E013FFE714BE61E
CD641AC006622626CB177D4B803706EFE821724ADCE0363560563E9BC4AF6ACB0
0566880F995BB850557915DBD1BEE2EFB39BB71126318B32B2DD6BF2EFDA5E796
12E35D0DF308B6980B8C6E0063E796DAA81878DBEF7BE1707FF351A0D32C59572
658847C8D35A5284C86E89EB15E9208CBA7BD7C8E1AD90BEE578FA2CDAD61F607
B305FE777723D58AF3C143086AF7A889BDD5BC9642C71D7B736BEB20C6BB3D573
1F917DF3F20988AE743EAC90B668AF95810FBEC41DBB5A5BDDE16D1C6251BD3FC
616E9B9B3083C6DCC933B07E2B2298A7C57FD7703E057C0200F2F56FC46F93D18
4A611E12F194545FD00375F4ED2926B57681AD65FEE70DCCDBF298E12AFF499D0
B6E2682B24AF2554A3281A585B89667FE73A53EA3116E0278D7410A538400BC9D
0FA5E0FFF63D36D642B0701F5DD0AE0D0006112F7CE56694D020EA7E25C652661
18A5193B587AA6DCC431D29F484CAE7FF5B7A7E5074C87A85A81A6CA69F0593BB
CFB251BE138D20AC94D76F376B0C7C579BA8D91472DBC167D57BCB74EAC93F936
D4E4F8672D2BA9021DD574F42FED8DC6E63617A348BD080C9F032EEA23C07D444
4AB2E69801F93A7CC06F444D9AB5F77DEA318A26720BB57DF6C5F940577B85630
9FE33E488C229B6A11958228560126D13D11E0B67CE2D5665B466D2B1CF27649F
92203D6E69BD30395A68693121241E8BE96C70594C44EDB22E6DA182812C882E5
1F027B9A79174CF7F38D07198033372FE7D737EB61EF5C36C9A66C7D66BBEC180
C552672660255B87F9F91DFC7E14185A2EFC65C201481599770A54849DAC3197D
2677F7E96F4A0117BCD1B42CD4447234A71AEED5554AB166BA23156CBE5E61C8F
BB3A0598C719C6F296D20D5CCD381897B4660DCE48A7E5592CB644C6F5E03594D
7248F6877AB90B3A3846EAADB6CA3D17F42A54B6F21D33CBEC426B9A9FB2B2D5F
A44894628BD1F0AE339DD341FE547F73C69F7C2E6F292BC1E40D50D417622C918
CE587AF5F4B945A58BE4CDF8F169D150A8E024173B04B99A135BD3AD2DE35640E
5FFAACFD4C6AF98581780A315AF80BC026155F0BEED947AD52E05BC50BB4A20D9
6B690496C2DF87691291AD8CBC8B415C8C51F9D53819C6A5FF5AFDDB2F7846951
1E5F9E9FD95C868A8D0D5EDEDA63EF940D77436E7F8058A67BF721AC1F903FAF6
93A33B8C616A2BE5BE3AC492BFE8417E0AEA556D385F9132E12BE720FB514F46B
E540D68F9C2F028D6CBCF1802EC90174E90C4B985D6FD0AA4EE861A73AE1BBF6B
FF82AF0E88B126EC102CFC5D9CC870061501FEF40E9BF4CC07B430A29968BD116
582052AD69D99270BD1681D82D19679AC5B08882BBDA109BA893AE2A9CF3398FE
C8FBBF089EF192B23C0DEFA09A32016BA013A47C08216BC38F4B3EBD85567B561
891481B02908AC74A42E9775ECEE2B7FE1B3B5C757082226518EE7092DB6D5C1F
7DE0A758B4503E7CA2E70956E0FE1C21094D9C0F012F4853615AECDC5B86267D9
6E7F6546252DC8EBDB3857939F55878993CA3BF2A598E3E,9A7A6923D29947DB#
>
```

The Network Security Processor returns the following response:

```
<239#Y#9F23CDECA25F53E629C3ADAFC896FE6EC6898407A9CA1E367CF04E2A0A
A5D00C20DD20F40F24BDF1CFA4570A1B9D231C1969C1C1F325096B6673795CF32
A6B08869C9E66B90672507CE7573557EE98EDE4D8943AA8ED975A57514D8E380B
16A8DED01D6EA1BC871045C04D724FF3D7929D69CA2D342590FF270496D4F5385
92423570DF235C19E5A97495A4FE3CBF285A1237D4F64A60FF8783B376A31294E
CBB9C05A172201146861BB96A02A634D5000B15ADBD8813CAA89A145D3AA38EA3
E45229CA63FBE98FDE8CB4A354C72719B06D6D6929B46CE80E203EA94901B2437
27923DEB932C7C9B2C02D4E267FF75B7EA0D58F07077461AB19633ECD9AA2DFF#
>
```

Sign message which does contain a key block

- To be signed data:
  31818E301806092A864886F70D010903310B06092A864886F70D0107033020060A2A864886F70D01
  09190331120410167EB0E72781E4940112233445566778301F06092A864886F70D010701311204104130
  3235364B30544230304530303030302F06092A864886F70D010904312204205D98145E22FCB7F6751
  B1A453A30C52487F924BC75EF46DB7974C7AA6C4BC72D

- Enveloped Data:
  0201003182019E3082019A020100304A3041310B300906035504061302555331153013060355040A130
  C545233342053616D706C6573311B301906035504031312545233342053616D706C65204341204B5244
  02053400000007304506092A864886F70D0101073038300D0609608648016503040201050030180
  6092A864886F70D010108300B0609608648016503040201300D06092A864886F70D01010904000
  48201002CBD086DC723286D97AA617C1E94980E539AE8BF51A926C55FE4858BE480856506F08F00
  9327E2EAC813D77C7B24A0AE52325C56452F750466CD5781EF1CB4B573A60724106D1252F18C27422
  9599B7887BA379C5081782198DC9A094493D389CA83D6F08D58D88E954806F7B00A620B20CA607
  90674727C79D74B79E039C985E2F107BEC30A7FC82E5CD4268A1A2CD579FC822CC366A572DC6995
  26A1B1CC0CE3F6830AE7FEAA2BE1464F1BFB4814D978F6007646407F224B75840F961127EF0E2347
  26A36A36564A32D2C74605A50849C9BF9F93F727D1AC68FED720DEEED2AD0A064B30AC01D2BEE
  B3CCD03D46315D40F1FFD6D260DC6F6537D70E02818A73081AD06092A864886F70D01070130819F
  06082A864886F70D030704080123456789ABCDEF8081885332A1F84521DE2D3B23EBE3CB2D674B1
  6114EC598214102C3DEE175C2A669400EB039136E632E4A32140AAB5546AC478799F7B7A025335F45
  CCA3CD1894314FF513E3E02573ADB5135DF8B1DB3277D9DE273DC6A8B5E79D215F63B93A52137DBA
  FBE5CC3FF472919D86D2409762370FA80A77AED183E1ED597BF9BFDC9D286934C7C1E1E8D003FB

- Encrypt key offset: 331

- IV offset: 901

- BE offset: 923

- Signing Token:
  1bRVN000,000601000008008800201D66E11E07A026E6EDA02E4B86C4057F02B1DAE3965C94342
  7AC0E11BEA252D0,9CCBE32350B1C3F5

- Private Key:
  1wRGE000,8D103BB1CEDFB183FD570F38ECC308D092C98034263A637984DB6DF68D55C28A1BF06
  0679F14EB6995FF3252C461BAAFE521A609C6E96389163B83557B84DBDE02EAB4287FB5B4950E013
  FFE714BE61ECD641AC006622626CB177D4B803706EFE821724ADCE0363560563E9BC4AF6ACB0056
  6880F995BB850557915DBD1BEE2EFB39BB71126318B32B2DD6BF2EFDA5E79612E35D0DF308B6980B
  8C6E0063E796DAA81878DBEF7BE1707FF351A0D32C59572658847C8D35A5284C86E89EB15E9208C
  BA7BD7C8E1AD90BEE578FA2CDAD61F607B305FE777723D58AF3C143086AF7A889BDD5BC9642C71
  D7B736BEB20C6BB3D5731F917DF3F20988AE743EAC90B668AF95810FBEC41DBB5A5BDDE16D1C625
  1BD3FC616E9B9B3083C6DCC933B07E2B2298A7C57FD7703E057C0200F2F56FC46F93D184A611E12F1
  94545FD00375F4ED2926B57681AD65FEE70DCCDBF298E12AFF499D0B6E2682B24AF2554A3281A58
  5B89667FE73A53EA3116E0278D7410A538400BC9D0FA5E0FFF63D36D642B0701F5DD0AE0D000611
  2F7CE56694D020EA7E25C65266118A5193B587AA6DCC431D29F484CAE7FF5B7A7E5074C87A85A81
  A6CA69F0593BBCFB251BE138D20AC94D76F376B0C7C579BA8D91472DBC167D57BCB74EAC93F936
  D4E4F8672D2BA9021DD574F42FED8DC6E63617A348BD080C9F032EEA23C07D4444AB2E69801F93
  A7CC06F444D9AB5F77DEA318A26720BB57DF6C5F940577B856309FE33E488C229B6A119582285601
  26D13D11E0B67CE2D5665B466D2B1CF27649F92203D6E69BD30395A68693121241E8BE96C70594C44
  EDB22E6DA182812C882E51F027B9A79174CF7F38D07198033372FE7D737EB61EF5C36C9A66C7D66BB
  EC180C552672660255B87F9F91DFC7E14185A2EFC65C201481599770A54849DAC3197D2677F7E96F4
  A0117BCD1B42CD4447234A71AEED5554AB166BA23156CBE5E61C8FBB3A0598C719C6F296D20D5CC
  D381897B4660DCE48A7E5592CB644C6F5E03594D7248F6877AB90B3A3846EAADB6CA3D17F42A54

```
B6F21D33CBEC426B9A9FB2B2D5FA44894628BD1F0AE339DD341FE547F73C69F7C2E6F292BC1E40D
50D417622C918CE587AF5F4B945A58BE4CDF8F169D150A8E024173B04B99A135BD3AD2DE35640E5
FFAACFD4C6AF98581780A315AF80BC026155F0BEED947AD52E05BC50BB4A20D96B690496C2DF8
7691291AD8CBC8B415C8C51F9D53819C6A5FF5AFDDB2F78469511E5F9E9FD95C868A8D0D5EDEDA6
3EF940D77436E7F8058A67BF721AC1F903FAF693A33B8C616A2BE5BE3AC492BFE8417E0AEA556D3
85F9132E12BE720FB514F46BE540D68F9C2F028D6CBCF1802EC90174E90C4B985D6FD0AA4EE861A
73AE1BBF6BFF82AF0E88B126EC102CFC5D9CC870061501FEF40E9BF4CC07B430A29968BD1165820
52AD69D99270BD1681D82D19679AC5B08882BBDA109BA893AE2A9CF3398FEC8FBBF089EF192B23
C0DEFA09A32016BA013A47C08216BC38F4B3EBD85567B561891481B02908AC74A42E9775ECEE2B7
FE1B3B5C757082226518EE7092DB6D5C1F7DE0A758B4503E7CA2E70956E0FE1C21094D9C0F012F48
53615AECDC5B86267D96E7F6546252DC8EBDB3857939F55878993CA3BF2A598E3E,9A7A6923D2994
7DB
```

The command looks like this:

```
<139#5#1#31818E301806092A864886F70D010903310B06092A864886F70D0107
033020060A2A864886F70D0109190331120410167EB0E72781E49401122334455
66778301F06092A864886F70D01070131120410413032353364B30544230304530
303030302F06092A864886F70D010904312204205D98145E22FCB7F6751B1A453
A30C52487F924BC75EF46DB7974C7AA6C4BC72D#020100318201 9E3082019A020
100304A3041310B30090603550406130255533115301306035504 0A130C545233
342053616D706C6573311B301906035504031312545233342053616D706C65204
341204B524402053400000000730450609 2A864886F70D0101073038300D060960
864801650304020105003018060 92A864886F70D010108300B060960864801650
3040201300D06092A864886F70D0101090400048201002CBD086DC723286D97AA
617C1E94980E539AE8BF51A926C55FE4858BE480856506F08F009327E2EAC813D
77C7B24A0AE52325C56452F750466CD5781EF1CB4B573A60724106D1252F18C27
4229599B7887BA379C5081782198DC9A094493D389CA83D6F08D58D88E954806F
7B00A620B20CA60790674727C79D74B79E039C985E2F107BEC30A7FC82E5CD426
8A1A2CD579FC822CC366A572DC699526A1B1CC0CE3F6830AE7FEAA2BE1464F1BF
B4814D978F6007646407F224B75840F961127EF0E234726A36A36564A32D2C746
05A50849C9BF9F93F727D1AC68FED720DEEED2AD0A064B30AC01D2BEEB3CCD03D
46315D40F1FFD6D260DC6F6537D70E02818A73081AD06092A864886F70D010701
30819F06082A864886F70D030704080123456789ABCDEF8081885332A1F84521D
E2D3B23EBE3CB2D674B16114EC598214102C3DEE175C2A669400EB039136E632E
4A32140AAB5546AC478799F7B7A025335F45CCA3CD1894314FF513E3E02573ADB
5135DF8B1DB3277D9DE273DC6A8B5E79D215F63B93A52137DBAFBE5CC3FF47291
9D86D2409762370FA80A77AED183E1ED597BF9BFDC9D286934C7C1E1E8D003FB#
331#901#923#1bRVN000,00060100000800880020 1D66E11E07A026E6EDA02E4B
86C4057F02B1DAE3965C943427AC0E11BEA252D0,9CCBE32350B1C3F5#1wRGE00
0,8D103BB1CEDFB183FD570F38ECC308D092C98034263A637984DB6DF68D55C28
A1BF060679F14EB6995FF3252C461BAAFE521A609C6E96389163B83557B84DBDE
02EAB4287FB5B4950E013FFE714BE61ECD641AC006622626CB177D4B803706EFE
821724ADCE0363560563E9BC4AF6ACB00566880F995BB850557915DBD1BEE2EFB
39BB71126318B32B2DD6BF2EFDA5E79612E35D0DF308B6980B8C6E0063E796DAA
81878DBEF7BE1707FF351A0D32C59572658847C8D35A5284C86E89EB15E9208CB
A7BD7C8E1AD90BEE578FA2CDAD61F607B305FE777723D58AF3C143086AF7A889B
DD5BC9642C71D7B736BEB20C6BB3D5731F917DF3F20988AE743EAC90B668AF958
10FBEC41DBB5A5BDDE16D1C6251BD3FC616E9B9B3083C6DCC933B07E2B2298A7C
57FD7703E057C0200F2F56FC46F93D184A611E12F194545FD00375F4ED2926B57
681AD65FEE70DCCDBF298E12AFF499D0B6E2682B24AF2554A3281A585B89667FE
73A53EA3116E0278D7410A538400BC9D0FA5E0FFF63D36D642B0701F5DD0AE0D0
006112F7CE56694D020EA7E25C65266118A5193B587AA6DCC431D29F484CAE7FF
5B7A7E5074C87A85A81A6CA69F0593BBCFB251BE138D20AC94D76F376B0C7C579
BA8D91472DBC167D57BCB74EAC93F936D4E4F8672D2BA9021DD574F42FED8DC6E
63617A348BD080C9F032EEA23C07D4444AB2E69801F93A7CC06F444D9AB5F77DE
```

```
A318A26720BB57DF6C5F940577B856309FE33E488C229B6A11958228560126D13
D11E0B67CE2D5665B466D2B1CF27649F92203D6E69BD30395A68693121241E8BE
96C70594C44EDB22E6DA182812C882E51F027B9A79174CF7F38D07198033372FE
7D737EB61EF5C36C9A66C7D66BBEC180C552672660255B87F9F91DFC7E14185A2
EFC65C201481599770A54849DAC3197D2677F7E96F4A0117BCD1B42CD4447234A
71AEED5554AB166BA23156CBE5E61C8FBB3A0598C719C6F296D20D5CCD381897B
4660DCE48A7E5592CB644C6F5E03594D7248F6877AB90B3A3846EAADB6CA3D17F
42A54B6F21D33CBEC426B9A9FB2B2D5FA44894628BD1F0AE339DD341FE547F73C
69F7C2E6F292BC1E40D50D417622C918CE587AF5F4B945A58BE4CDF8F169D150A
8E024173B04B99A135BD3AD2DE35640E5FFAACFD4C6AF98581780A315AF80BC02
6155F0BEED947AD52E05BC50BB4A20D96B690496C2DF87691291AD8CBC8B415C8
C51F9D53819C6A5FF5AFDDB2F78469511E5F9E9FD95C868A8D0D5EDEDA63EF940
D77436E7F8058A67BF721AC1F903FAF693A33B8C616A2BE5BE3AC492BFE8417E0
AEA556D385F9132E12BE720FB514F46BE540D68F9C2F028D6CBCF1802EC90174E
90C4B985D6FD0AA4EE861A73AE1BBF6BFF82AF0E88B126EC102CFC5D9CC870061
501FEF40E9BF4CC07B430A29968BD116582052AD69D99270BD1681D82D19679AC
5B08882BBDA109BA893AE2A9CF3398FEC8FBBF089EF192B23C0DEFA09A32016BA
013A47C08216BC38F4B3EBD85567B561891481B02908AC74A42E9775ECEE2B7FE
1B3B5C757082226518EE7092DB6D5C1F7DE0A758B4503E7CA2E70956E0FE1C210
94D9C0F012F4853615AECDC5B86267D96E7F6546252DC8EBDB3857939F5587899
3CA3BF2A598E3E,9A7A6923D29947DB#>
```

The Network Security Processor returns the following response:

```
<239#Y#97BFCE9F17F1D3BA795ABF2453280A3DA08FF79A280786044CE04EFEC6
D4C1F54F6F0E0938221A7405ACCE21191FAC86B99810323BB73B4004FB5DDDE0A
AAC2AD0431F0DED3E8BCF95750A3D5C26B333AEF830E7EE9A24B4700525372E0D
2D9F240AC6A76D20294BA7A651B628822E9A1A6483A65C931128043C71C69DC2A
75181D4787920CC835D8C21109895BA2955958603361 1F78EDC3D4856D2C1FB7C
6D25C1C9B3FD0A4E5C2B4E606D614012B978A8B58C288FD0E166FF8D561074B4F
3497AFDB468BB0B27753AA35FFE7CA3FA7C6A47BACC63F1688C9BA86DD6520612
4286794CBC0EFD1FB9BC3E909163F17D8978D09469397A867F6ECF6CD9CDEB6E#
>
```

Generate ISO/IEC 9796-2 Digital Signature (Command 358)

Command 358 – This command generates a digital signature for use with EMV smart cards.

This command is a premium value command, it is not enabled in the Network Security Processor's default security policy. You must purchase this command in the form of a command 105, and then enable it in the Network Security Processor's security policy.

Command

```
<358#Reserved#Hash Algorithm#Header#Msg1#Msg2#Trailer#AKBPRV#>
```

Response

```
<458#Reserved#Digital Signature#Check Digits#Hashed Data#>[CRLF]
```

Calling Parameters

`358`

Field 0, the command identifier.

`Reserved`

Field 1, this field must contain the value 0.

`Hash Algorithm`

Field 2, this field must contain the value 3, indicating that the SHA-1 algorithm is used to generate the message digest.

`Header`

Field 3, this field must contain the value 6A.

`Msg1`

Field 4, the most significant bits of the message to be signed. The length of this field is based on the number of bytes in the modulus. The formula to determine the length is:

2 * (modulus size -22)

For example, a 1024-bit RSA key has a 128 byte modulus, 128 - 22 =106, 106 x 2 = 212.

When using a 1024-bit key to generate a digital signature, the length of MSG1 must be 212 hexadecimal characters. The following table contains some common RSA key sizes and their corresponding Msg1 lengths.

Key Size (in bits)	Modulus Size (in bytes)	2*(Modulus size-22)	Msg1 Length (in hexadecimal characters)
1024	128	2 * (106)	212
2048	256	2 * (234)	468
4096	512	2 * (490)	980

`Msg2`

Field 4, the remainder of the message to be signed. The number of hexadecimal characters allowed in this field must be in the range of 2 through 16,384.

`Trailer`

Field 6, this field must contain the value BC.

AKB$_{PRV}$

Field 7, the AKB of the signing key. The header for this private key can be either 1sRGN000 or 1sRGE000. In version 1.42 and above, this additional header value is supported: 1pRGE000.

The size of the RSA private key must be in the range of 1024-bits through 4096-bits. If option 8F is enabled, the minimum RSA private key size can be 768 bits.

Table 10-31 Command 358: Generate ISO/IEC 9796-2 Digital Signature

Field	Contents	Length (bytes)	Legal Characters
0	Command Identifier	3	358
1	Reserved	1	0
2	Hash Algorithm	1	3
3	Header	2	6A
4	Msg1	Variable	0 - 9, A - F
5	Msg2	2-16,384	0 - 9, A - F
6	Trailer	2	BC
7	AKB$_{PRV}$	Variable	Printable characters

Responding Parameters

`458`

Field 0, the response identifier.

`Reserved`

Field 1, this field will contain the value 0.

`Digital Signature`

Field 2, the generated digital signature. The length of this field will be twice the size of the modulus (2*SOM), in bytes. For example, the modulus of a 2048 bit RSA key is 256 bytes, therefore the digital signature will contain 512 hexadecimal characters.

`Check Digits`

Field 3, the check digits of the RSA key. Check digits are the leftmost 8 bytes of the SHA-1 message digest of the RSA key.

`Hash Data`

Field 4, the SHA-1 result of hashing the Msg1||Msg2 data.

Table 10-32 Response 458: Generate ISO/IEC 9796-2 Digital Signature

Field	Contents	Length (bytes)	Legal Characters
0	Response Identifier	3	458
1	Reserved	1	0
2	Digital Signature	Variable	0 - 9, A - F
3	Check digits - RSA key	8	0 - 9, A - F
4	Hash Data	40	0 - 9, A - F

Usage Notes

- The NSP does not validate the contents of the Msg1 or Msg2 fields. It only checks that the lengths are correct.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Msg1 (468 characters) =

```
1234567890123456789012345678901234567890123456789012345678901234567
8901234567890123456789012345678901234567890123456789012345678901234
5678901234567890123456789012345678901234567890123456789012345678901
2345678901234567890123456789012345678901234567890123456789012345678
9012345678901234567890123456789012345678901234567890123456789012345
6789012345678901234567890123456789012345678901234567890123456789012
3456789012345678901234567890123456789012345678901234567890123456789012345678
```

Msg2 (22 characters - PAN + PSN + Expiration Date) =

```
1234567890123456121234
```

2048 bit RSA Private Key =

```
1sRGE000,4DA78A1E081F1A09124CBA74DDE3F0EB2A5242CA4A9B1C6C207B03DF00
D13130E47A0C26DC582631E9870127F2953FA98EAC5E143857278128A084762A93A
45FA9DEB6F207D265F0A71441F9F66073B95A9DF36AAA414F6C4097A3D946B3414D
16C7DF6A87ADAF073473D37900093CFFD2DA976E117B312E4C9D1CE8E79ECA0989C
7E6570B8FE46ED69859C616E14BD33205E7D3E8E23B7E21139A7F989EF8E09DD824
B195B956636AB08264579C6931A48D294BF9C42E28505B9B4B23440BCCBBA5C4F85
C5A6A8F6B33DB8015A740DF8FB7C9D599143D540CDCC375CDBAA951CFD06E400586
E68CBF9F2E7B95919B84C0E8FDD66B3C74728E57ABD1A3C90C06924287EA538AD58
1EB4F565D74D23D221ECCCB7D3D333159158ED6741468D5753E64BB4037575A1938
E2568DF1440BE66F779CF3332793D3DF6E493103AB0E7179BD5460B9765056A8EC7
1168EC885683ADFE81C75FBC3F294B7059830BBE82709FE25B658B4F59B6DDC5671
D7478DC8246559C6A7C5C405ADFAA49D34831A02B64BA342B06EC24DAA38C39624A
99B48063A66B490C16486A75B62B7BD138630E35F42E713536FDB9220458610EB80
7EFBF506C4163EA5F7520D9018E7AEC8460D0E55D5C7F0894C3C36B6DC2ED2B937F
C6A68076FFD4BFB4FE0BD20E030F22FFFAAA660309C03F615083485A4956F6487F5
43B1665B96A10B44DAE84E6B8018F985912700BE677D70A14409A6AE53DCA5A0A58
7F4FA00E86A28F2622AFC6F9A7E75E443F08C3EA513FACC965551C3EBD58EB41112
88052F8B54F4ED41461CF61A6193B49846C5B9C12F1D2758401108A2819B9DFC7B2
FF4BF3A3925FB6DBA79814E692948F146ED4CCDA62893E38FDA032687D2FDB326B6
92A93A56BDDAA761B9D13D1720FB1F6775ED54958BA181C6245B2F472FECE2E01BA
4CA31286DCFCF95F606F6EDDAA99F1960CC85B85F4D5FDCAD5D1691066B3F17B1CE
0D77C5E440E43E4599CE2C51BFF113F9F63D7CFE1D45C2393645ECF0C006A85CD7B
0849ADB31624B51647AF65DCCDAF2E392782B0996F1C12F7EA0995752196F42771A
3530F14264E5EE471F80A42B12FDBF114E5BF5033BF7AC33D92C529974A7D486FD9
7983A76F2070978371DB749230F33B9B997BFB67112A7D90504BB3E94ACF4EBBEE1
417BE23EF50F2287ACFD57B5E8A3938F6CC585F6AA08E8548895B63B16058E8DE82
15CD5183F59172D36496F442CDE2BC60818FB462E9DD704CA981D081A183F98D2BC
9AB116450DF62E0ED6067DC65684A2CC69C2B290AF02D65C18B4AAB2A281B81C262
6F1AF24255C9B8DE616C8D0CCE299F9163D88D9AFD8AF06619AF36FECF2372FAE71
C756B269197CC92EC1C69AB157F5D27EF8D87ECCB416A5FAAE91DC5E3E4A96CE231
68A50D2CC924CC7DB8300379845C7493D384221CFC2CFDED2F139E2F2B3D9EE3477
2099A17E581B238E080303D5E0593C94647B4AE4849CB64CFBB71EBD9831B0DC6AC
75DA0B748FE32FFBA31541AA2F99C5C1B880D039C7777663FA1D042D81E8F03186A
4E9C6D9979BF02939022B126AF3A174B3777AF64002982D6A377F2D173C2C463DF2
DF422D0A12D4B5154A79D79436C0D36025C7CA2B086F343C8B1621FEF9674491A4F
A58C27EFFDE52C42,EDE3848DBDC93C85
```

The command looks like this:

```
<358#0#3#6A#1234567890123456789012345678901234567890123456789012345
6789012345678901234567890123456789012345678901234567890123456789012
3456789012345678901234567890123456789012345678901234567890123456789
0123456789012345678901234567890123456789012345678901234567890123456
7890123456789012345678901234567890123456789012345678901234567890123
4567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567
89012345678#1234567890123456121234#BC#1sRGE000,4DA78A1E081F1A09124C
BA74DDE3F0EB2A5242CA4A9B1C6C207B03DF00D13130E47A0C26DC582631E987012
7F2953FA98EAC5E143857278128A084762A93A45FA9DEB6F207D265F0A71441F9F6
6073B95A9DF36AAA414F6C4097A3D946B3414D16C7DF6A87ADAF073473D37900093
```

```
CFFD2DA976E117B312E4C9D1CE8E79ECA0989C7E6570B8FE46ED69859C616E14BD3
3205E7D3E8E23B7E21139A7F989EF8E09DD824B195B956636AB08264579C6931A48
D294BF9C42E28505B9B4B23440BCCBBA5C4F85C5A6A8F6B33DB8015A740DF8FB7C9
D599143D540CDCC375CDBAA951CFD06E400586E68CBF9F2E7B95919B84C0E8FDD66
B3C74728E57ABD1A3C90C06924287EA538AD581EB4F565D74D23D221ECCCB7D3D33
3159158ED6741468D5753E64BB4037575A1938E2568DF1440BE66F779CF3332793D
3DF6E493103AB0E7179BD5460B9765056A8EC71168EC885683ADFE81C75FBC3F294
B7059830BBE82709FE25B658B4F59B6DDC5671D7478DC8246559C6A7C5C405ADFAA
49D34831A02B64BA342B06EC24DAA38C39624A99B48063A66B490C16486A75B62B7
BD138630E35F42E713536FDB9220458610EB807EFBF506C4163EA5F7520D9018E7A
EC8460D0E55D5C7F0894C3C36B6DC2ED2B937FC6A68076FFD4BFB4FE0BD20E030F2
2FFFAAA660309C03F615083485A4956F6487F543B1665B96A10B44DAE84E6B8018F
985912700BE677D70A14409A6AE53DCA5A0A587F4FA00E86A28F2622AFC6F9A7E75
E443F08C3EA513FACC965551C3EBD58EB4111288052F8B54F4ED41461CF61A6193B
49846C5B9C12F1D2758401108A2819B9DFC7B2FF4BF3A3925FB6DBA79814E692948
F146ED4CCDA62893E38FDA032687D2FDB326B692A93A56BDDAA761B9D13D1720FB1
F6775ED54958BA181C6245B2F472FECE2E01BA4CA31286DCFCF95F606F6EDDAA99F
1960CC85B85F4D5FDCAD5D1691066B3F17B1CE0D77C5E440E43E4599CE2C51BFF11
3F9F63D7CFE1D45C2393645ECF0C006A85CD7B0849ADB31624B51647AF65DCCDAF2
E392782B0996F1C12F7EA0995752196F42771A3530F14264E5EE471F80A42B12FDB
F114E5BF5033BF7AC33D92C529974A7D486FD97983A76F2070978371DB749230F33
B9B997BFB67112A7D90504BB3E94ACF4EBBEE1417BE23EF50F2287ACFD57B5E8A39
38F6CC585F6AA08E8548895B63B16058E8DE8215CD5183F59172D36496F442CDE2B
C60818FB462E9DD704CA981D081A183F98D2BC9AB116450DF62E0ED6067DC65684A
2CC69C2B290AF02D65C18B4AAB2A281B81C2626F1AF24255C9B8DE616C8D0CCE299
F9163D88D9AFD8AF06619AF36FECF2372FAE71C756B269197CC92EC1C69AB157F5D
27EF8D87ECCB416A5FAAE91DC5E3E4A96CE23168A50D2CC924CC7DB8300379845C7
493D384221CFC2CFDED2F139E2F2B3D9EE34772099A17E581B238E080303D5E0593
C94647B4AE4849CB64CFBB71EBD9831B0DC6AC75DA0B748FE32FFBA31541AA2F99C
5C1B880D039C7777663FA1D042D81E8F03186A4E9C6D9979BF02939022B126AF3A1
74B3777AF64002982D6A377F2D173C2C463DF2DF422D0A12D4B5154A79D79436C0D
36025C7CA2B086F343C8B1621FEF9674491A4FA58C27EFFDE52C42,EDE3848DBDC9
3C85#>
```

The Network Security Processor returns the following response:

```
<458#0#0801D2421CBB37E7FAA3D2FDF4C281AB68B6051C55F4AFB7CA426C1EF0B6
92F3245B7CD0C9A0C0F6B9E3F6B5FC3A494D6BB0C9BA8CB9E0828B96C757A0891CD
E62C1FF74EBB40EEF039098E400DA2FD42866BAE91D6A18BC0DF7F8345607D22241
5972920C16F10A623E3218F109BC6856D4F0545C755C7E579ED1234A6630CAC2F44
9764FBC88A58C0DBB510AF1B34E016D33F35872772CA764977126CA072A03D97841
791A4D44787EB7DD2D226314D8AF4532B8A7DB5D5FB66F47AF4C86B4E097C5EE650
432AD7A41481C922C92FB301012A5708D2E585BAD8ECBAB6171F39A7C1071A33AD3
C87F1DDF8CFCE67DEA52B7ED64A9B197D967D85BA0A77500AF#A786EDA4#0BD8A94
F83205E09EAD182D6E4E3368BE022B3D0#>
```

11

Printing Commands

The following commands support printing of letters that contain either a cleartext PIN or key component:

- Combine Key Components (Command 15E)
- Print PIN Letter (Command 161)
- PIN Issuance: IBM 3624 Method (Command 162)
- PIN Issuance: Visa Method (Command 163)
- Reset Printing Command Counter (Command 16A)
- Divide a Key into Components (Command 16E)
- Print Component Letter (Command 16F)

These commands are disabled in the Network Security Processor's default security policy.

note These commands are only allowed on the NIC1 Print Command Port. For information on how to configure the Print Command Port, see section 4 of the *Installation and Operations Guide for the Atalla Ax160 NSP*.

Enable and disable printing commands

Your security administrators must perform multiple procedures to enable printing commands. The licensing procedure is performed once per Network Security Processor. In addition, the Network Security Processor's security policy must be configured each day that the host application will use the printing commands.

Licensing procedure

The licensing procedure is a three step process.

1. You must purchase premium value options E5 and 87 for each Network Security Processor that will use the printing commands 161 or 16F to print PIN letters or key components. Option E5 allows the Network Security Processor to process the printing

commands for eight hours of elapsed time. Option 87 enables the Network Security Processor's second Network Interface Connector (NIC2), this is the NIC that communicates with the printer. Be sure to include the serial number of the Network Security Processor when placing your order for these options. The serial number is returned in field one of the response to the <9A#ID#> command.

2. When you purchase these options, you will receive an email that contains the command 105 license string that is based on the Network Security Processor's serial number. Send the command 105 command to the appropriate Network Security Processor.

3. A minimum of two security administrator must then use the SCA to enable the appropriate printing commands and option 87 in the Network Security Processor's security policy. See sections titled **NSP Option Configuration** and **Enable Command Counting** in section 4 of the *SCA User Guide,* for the steps required to accomplish this task.

note If the Network Security Processor is reset to factory state the license is erased.

Daily printing procedure

note Command count values are stored in non-volatile memory; they are not erased when the Network Security Processor is powered off. They are erased when the Network Security Processor is reset to factory state.

Security Administrators use the SCA-3

A minimum of two security administrators must use the SCA-3 to enable option E5 and also to specify the count value for each of the printing commands that your host application will send to the Network Security Processor.

1. Generate a command 108 that contains the option E5 enable string and the commands and count values. For example, if you going to generate 50 PINs using the command 162, and then use command 161 to print a PIN letter for each PIN, and also use command 16F to print 200 key component letters, the command 108 would look like this;

```
<108#(E5)=”E”;162=n50;161=n50;16F=n200#>
```

note When you specify a command count it overwrites the current count value.

2. Send the command 108 from step 1 to the Network Security Processor.

3. Fields two and four of the Network Security Processor's 208 response will each contain a 16 hexadecimal character challenge value that you must then encrypt under the Network Security Processor's Master File Key. Fields three and five contain the check digits of the challenge values. In the example 208 response, the challenge values are underlined.
   ```
   <208##0DC2D50EF492E33D#30D6#DAF29816C8B96843#9EBC#1#
   0000000000000004#A7PV87#>
   ```

4. Encrypt the challenge values. See the subsection titled **Encrypting a challenge or hash value,** in section 5 of the *SCA-3 User Guide,* for the steps required to accomplish this task. Using the values shown in step 3 above and the Master File Key published in this manual, the encrypted challenge values are: `8CA79B7503C998C5` and `83C406915254D4C9`.

5. Concatenate the two 16 hexadecimal character encrypted challenge values, from step 4 above, generate a command 109, and then send it to the Network Security Processor. For example:
   ```
   <109##8CA79B7503C998C583C406915254D4C9#>
   ```
   If the challenge values were encrypted correctly, the 209 response will contain the modifications to the Network Security Processor's security policy. For example:
   ```
   <209#(E5)="E";162=n50;161=n50;16F=n200#0000000000000001#
   A7PV87#>
   ```

6. To confirm that the appropriate commands have been enabled with the correct count values, you can send this command to the Network Security Processor.
   ```
   <9A#COUNT#>
   ```
   The response will indicate the counted commands and their current count values. For example:
   ```
   <AA#A7PV87#1#0161-0000000050#0162-0000000050#
   016F-0000000200#>
   ```

Disable printing commands

These actions will disable printing commands and cause the Network Security Processor to return an error <00#0300xx#> indicating that it is unable to process the printing command.

- Send command 16A to the Network Security Processor. The Network Security Processor will immediately disable all of the printing commands and remove them from the command count table.
- Disable option E5. This action will cause the Network Security Processor to immediately stop processing all printing commands, however the printing command count values will be retained.
- When a count value for a specific printing command reaches zero, only that command will be disabled.

Letter template file

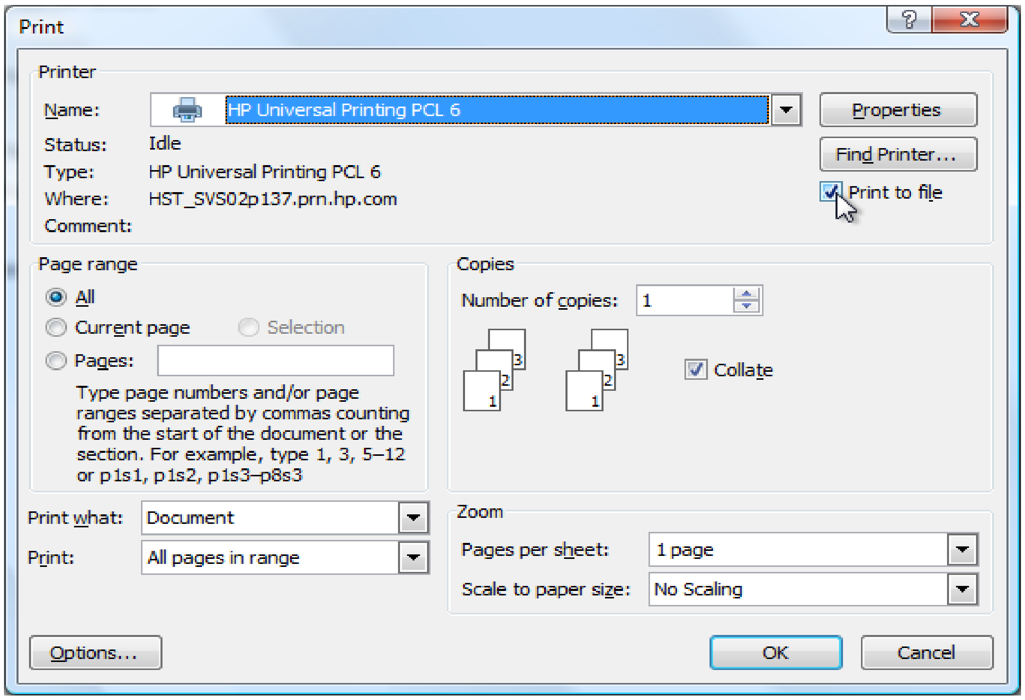
To print a PIN or key component letter, the host application must first create a letter template file. Once this file has been created, the host application can send it as binary data to the Network Security Processor in either the Print PIN Letter (Command 161) or Print Component Letter (Command 16F) command. The Network Security Processor will process the letter template file, and then create a print job which it will then send to the printer.

For maximum performance and efficiency, the letter template file should be a simple ASCII text file. You can use a text editor such as Windows Notepad to create the letter template file. The printer’s default values for font and size will be used to print the letter.

You can use Microsoft Word to create a complex letter template file. Typically, the template file is a large file that require multiple commands to send the entire letter template file to the Network Security Processor. If you use Microsoft Word to create a letter template file, all of these restrictions apply:

- The HP Universal Print Driver for Windows PCL6 is required on the PC/laptop that creates the letter template file. It is available for download from the HP.com website; search for HP Universal Print Driver for Windows PCL6.
- The font for these marker strings must be either "`Courier`" or "`Courier New`", the remainder of the document can utilize any font.
- The entire PIN, component, check digit, or reference marker string must be input without using the copy/paste features and cannot be modified once it has been input.
- All marker strings must have a leading and trailing space.
- The data encoding method specified in field 10 of the print letter command must contain the letter W.
- To create the letter template file you must print it to the HP Universal PCL 6 driver and specify the **Print to File** option as shown in the following screen shots.

  In Word 2007 the window looks like this.



Print

Printer

Name: HP Universal Printing PCL 6

Status: Idle

Type: HP Universal Printing PCL 6

Where: HST_SVS02p137.prn.hp.com

Comment:

[✓] Print to file

Page range: [● All] ○ Current page ○ Selection ○ Pages:

Type page numbers and/or page ranges separated by commas counting from the start of the document or the section. For example, type 1, 3, 5–12 or p1s1, p1s2, p1s3–p8s3

Copies

Number of copies: 1

[✓] Collate

Print what: Document

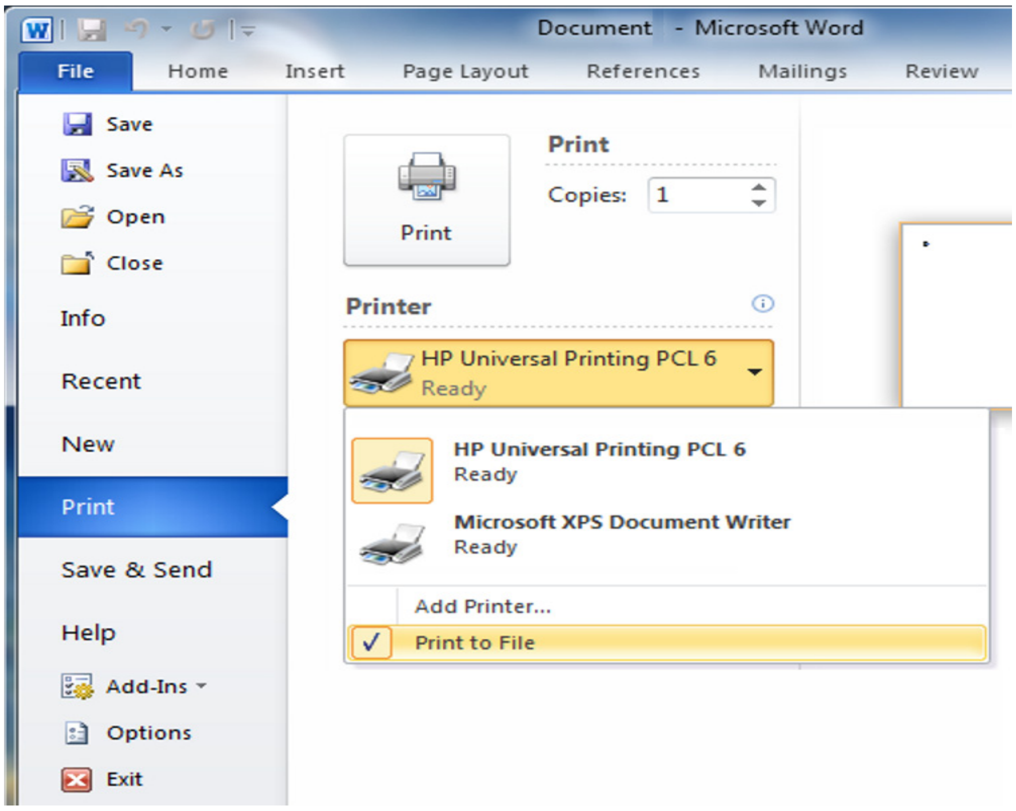
Print: All pages in range

Zoom

Pages per sheet: 1 page

Scale to paper size: No Scaling

In Word 2010 the relevant portion of the window looks like this.



Document - Microsoft Word

Print

Copies: 1

Printer

HP Universal Printing PCL 6
Ready

HP Universal Printing PCL 6
Ready

Microsoft XPS Document Writer
Ready

Add Printer...

✓ Print to File

You must specify the filename to save the output file. The binary data of this output file must be supplied in the Data Block field of the Print PIN Letter (Command 161) or Print Component Letter (Command 16F).

Marker strings

The letter template file must contain a marker string which is a unique value within the letter template file. The numbers 0-9 and the letters A-Z and a-z are allowed in a marker string.

There are four types of marker strings.

Marker Type	Command	Required	Length in characters
PIN	161	Yes	12
Component	16F	Yes	19
Check Digits	16F	Yes	varies based on the check digit method
Reference	16F	No	19

The Network Security Processor will replace the marker string with the PIN, component, check digits, or reference value prior to sending the print job to the printer. For a PIN letter template sent to the Network Security Processor in the Print PIN Letter (Command 161), the PIN marker string must be 12 characters. For a component letter template sent to the Network Security Processor in the Print Component Letter (Command 16F), there are two required marker strings and one optional marker string. The component marker string must be 19 characters. For components that are longer than 16 characters, the marker string must be repeated. For example, the component marker string must be present three times for a 3key-3DES key component. A maximum of four component marker strings may be present in a letter template file. If there are more component marker strings present in the letter template file than are needed (for example, when a letter template file contains 4 component marker strings but is printing only a 2-key 3DES component), the unused component marker strings will be filled with spaces. The length of the check digits marker string is based on the check digit method. The optional reference marker string, if present, must be 19 characters.

Print large letter template files

The maximum size of the letter template file is 1,048,576 bytes (1 megabyte). If the letter template file is larger than 30,000 bytes, the host application must split it into separate data blocks and send each data block as a separate command to the Network Security Processor. The maximum size of a data block is 30,000 bytes. When a letter template file is split into multiple data blocks, information about the PIN or component (i.e. PIN block type, PIN Encryption Key, encrypted PIN block, etc.) must be included in only the final command.

The Network Security Processor can receive a maximum of four concurrent multi-command letter template files. The Network Security Processor’s response to the first command in a multi-command sequence will include a continuation index that must be provided in the subsequent intermediate and final commands required to send the remainder of the letter template file. When the Network Security Processor receives the final data block of the letter template file, it will replace the marker strings with the clear PIN or component and check digit values, and then send the complete letter print job to the printer.

Print an encrypted PIN

The PIN printing command requires that the PIN be encrypted under a PIN Printing Key. As such, an ANSI or ISO-3 PIN block that is encrypted under a PIN Encryption Key must be re-encrypted under a PIN Printing Key. To obtain an encrypted PIN that can be printed perform these steps:

1. Use Generate 3DES Working Key, Any Type (Command 10) to generate a PIN Printing Key, the header must be either 1PUpN000 or 1PUpE000. The resulting value will be used as field 6 in the commands listed in step 2, and as field 7 in the command listed in step 3.
2. Use either PIN Issuance: IBM 3624 Method (Command 162) or PIN Issuance: Visa Method (Command 163) to produce an encrypted PIN block encrypted under the PIN Printing Key.
3. Use Print PIN Letter (Command 161) to print the encrypted PIN from step 2.

Print a key component

The key component printing command requires that the key component AKB have an appropriate header. There are two ways to obtain key components that can be printed.

Divide an existing key into key components

To obtain an encrypted key component that can be printed perform these steps:

1. Use Divide a Key into Components (Command 16E) to create encrypted key components from an existing key.
2. Use Print Component Letter (Command 16F) to print the encrypted key component from step 1.

Create new key components and combine them into a key

1. Use Print Component Letter (Command 16F) to generate a random key component and print it. Repeat this step to create the desired number of key components.
2. Use Combine Key Components (Command 15E) to combine the key components into a key in AKB format which can then be stored on the host application’s key database. The response to the command also returns the key encrypted under the Key Exchange Key.

Print a test page

Before attempting to print a batch of PIN or component letters it is highly recommended that the host application print a test page to ensure that the printer is on-line and operating properly. The test page feature can also be used to print operator instructions, job identifiers, start of job, and end of job pages.

Below are example commands that will print the text “Test Page” on a page.

The command looks like this:

```
<161#0#0########A#B#9#9#Test Page#>
```

The Network Security Processor returns the following response:

```
<261####>
```

The command looks like this:

```
<16F#0#0########A#B#9#9#Test Page#>
```

The Network Security Processor returns the following response:

```
<26F####>
```

HP Printers

Only printers that support Printer Command Language version 6 are supported.

Some HP printers do not allow you to configure the printer port number, if you encounter this situation, specify 9100 (PRINTER_PORT_2=9100) in the config.prm file.

Some HP printers have buffers and do not print or eject the printed page until the buffer is full. To resolve this issue, the host application must include a form feed character (0x0C) at the end of the print data. In the following example the symbol [0C] represents the form feed character.

```
<161#0#0########A#B#10#10#Test Page[0C]#>
```

Some HP printers will only print the page when the last two characters of the message to be printed are the carriage return and line feed characters. To resolve this issue, the host application must include both the carriage return character (0x0D) and line feed character (0x0A) at the end of the print data. In the following example, the symbols [0D] and [0A] represent the carriage return and line feed characters.

```
<161#0#0########A#B#11#11#Test Page[0D][0A]#>
```

Manage printer sockets

The Network Security Processor opens a socket on the printer after it has received all the print job data for a letter. After the socket is established, the Network Security Processor sends the print job to the printer. The printer will acknowledge receipt of the print job and print the letter. The Network Security Processor will then close the socket connection. The Network Security Processor will open one socket for each complete print job that it will send to the printer. The Network Security Processor can open a maximum of 16 sockets on the printer. The Network Security Processor will wait for 75 seconds to establish a socket connection on the printer. If it cannot establish the socket connection within this time, it will return an error 11 to the host application. The detailed error 11xx will indicate the cause of the error (see Detailed Errors for the specific detailed error values).

Printing errors

When the Network Security Processor receives an error from the printer, it will return an error 11 to the host application. The detailed error 11xx will indicate the cause of the error (see Detailed Errors for the specific detailed error values).

The Network Security Processor does not support status reporting from the printer, it only checks that the printer has received the print job. Once the printer acknowledges receipt of the print job, the Network Security Processor will return the response to the host application. If the printer is out of paper, the printer will buffer the job, and then print it once the operator loads the printer with paper.

Clear the printer’s buffer

Printers store print jobs in their memory. After printing a job of PIN or component letters, it is highly recommended that the print job be erased from the printer’s memory. The Network Security Processor does not perform this function, it must be performed by an operator.

Printing Commands

Quick Reference

Table 11-1 on page 11-9 identifies each command by number, name, and purpose.

.

Table 11-1 Printing Commands

Command	**Name**	**Purpose**
15E	Combine Key Components	Combines key components into a key.
161	Print PIN Letter	Prints a clear text PIN in a letter.
162	PIN Issuance: IBM3624 Method	Issues a PIN using the IBM 3624 method.
163	PIN Issuance: Visa Method	Issues a PIN using the Visa method.
16A	Reset Printing Command Counter	Resets the print command counter.

Table 11-1 Printing Commands (continued)

Command	Name	Purpose
16E	Divide a Key into Components	Splits a key into multiple key components.
16F	Print Component Letter	Prints a clear text key component in a letter.

Combine Key Components (Command 15E)

Command 15E combines key components, which are in AKB format, into a Working Key. The Working Key is returned, in AKB format, encrypted under the Key Exchange Key (KEK) and the Master File Key (MFK). The minimum number of key components is two, and the maximum number of key components is four. This command supports either 3DES or AES key components.

This command is not enabled in the Network Security Processor's default security policy. It is only allowed on the NIC1 Print Command Port.

note It is highly recommended that this command be enabled for a specific number of executions. For information on how to configure the Network Security Processor to limit how many times this command can be executed, see the Command Count feature which is documented in section 4 of the *Atalla Secure Configuration Assistant-3 User Guide*.

Command

<15E#Header,$E_{MFK.E}$(KEK),MAC#Header,$E_{MFK.E}$(Comp-1),MAC#
Header,$E_{MFK.E}$(Comp-2),MAC#[Header,$E_{MFK.E}$(Comp-3),MAC]#
[Header,$E_{MFK.E}$(Comp-4),MAC]#Reserved#[AES Check Digits Method]#>

Response

<25E#Header,$E_{KEK.E}$(WK),MAC#Header,$E_{MFK.E}$(WK),MAC#
Working Key Check Digits#>[CRLF]

Calling Parameters

`15E`

Field 0, the command identifier.

Header,$E_{MFK.E}$(KEK),MAC

Field 1, the Key Exchange Key (KEK) encrypted under the MFK. After combining the key components into a Working Key, the Network Security Processor uses the KEK to encrypt the Working Key. This field contains a 74 byte value. The KEK can be a 2key-3DES key, it can be a 3key-3DES key only if the MFK is also a 3key-3DES key. The length of the KEK must be equal to or greater than the length of the key components. The following headers are supported: 1KDEE000,1KDEN000, 1KDNE000, and 1KDNN000.

`Header,`$E_{MFK.E}$`(Comp-1),MAC`

Field 2, the first key component encrypted under the MFK. Byte 6 of the key component header must contain one of the following values: C, 1, 2, 3, 4, i, e, k, or n. The length of key component can be 74 bytes. When byte 2 of the header contains the capital letter J, it can be 58, 74 or 90 bytes in length.

`Header,`$E_{MFK.E}$`(Comp-2),MAC`

Field 3, the second key component encrypted under the MFK. The header and component length must match the key component present in field 2.

`[Header,`$E_{MFK.E}$`(Comp-3),MAC]`

Field 4, the third key component encrypted under the MFK. This field can be empty. If present, the header and component length must match the key component present in field 2.

`[Header,`$E_{MFK.E}$`(Comp-4),MAC]`

Field 5, the fourth key component encrypted under the MFK. This field can be empty. If present, the header and component length must match the key component present in field 2.

`Reserved`

Field 6, this field is reserved for future use. This field must be empty.

`[AES Check Digits Method]`

Field 7, the method used to calculate check digits for an AES key. When field 2 contains a 3-DES key component, this field should be empty. For AES keys, one of the following values must be present.

Value	**Description**
C	The key is used to calculate a MAC of an all zero block using the CMAC algorithm. The leftmost 10 hexadecimal digits are the check digits.
H	SHA-256 hash of the key (64 hexadecimal digits)

Table 11-2 Command 15E: Combine Key Components

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Command identifier	3	15E
1	Header,$E_{MFK.E}$(KEK),MAC	74	printable ASCII
2	Header,$E_{MFK.E}$(Comp-1),MAC	58, 74, 90	printable ASCII
3	Header,$E_{MFK.E}$(Comp-2),MAC	58, 74, 90	printable ASCII

Table 11-2 Command 15E: Combine Key Components (continued)

Field	Contents	Length (bytes)	Legal Characters
4	[Header,$E_{MFK.E}$(Comp-3),MAC]	0, 58, 74, 90	printable ASCII
5	[Header,$E_{MFK.E}$(Comp-4),MAC]	0, 58, 74, 90	printable ASCII
6	Reserved	0	n/a
7	[AES Check Digits Method]	0, 1	C, H

Responding Parameters

`25E`

Field 0, the response identifier.

Header,$E_{KEK.E}$(Working Key),MAC

Field 1, the Working Key encrypted under the KEK. When the Working Key is either a 3DES or an AES-192 bit key, this field contains a 74 byte value. When the working key is an
AES-128 bit key, this field contains a 58 byte value. When the working key is an AES-256 bit key, this field contains a 90 byte value.

Header,$E_{MFK.E}$(Working Key),MAC

Field 2, the Working Key encrypted under the MFK. When the Working Key is either a 3DES or an AES-192 bit key, this field contains a 74 byte value. When the working key is an
AES-128 bit key, this field contains a 58 byte value. When the working key is an AES-256 bit key, this field contains a 90 byte value.

`Working Key Check Digits`

Field 3, the check digits of the Working Key. For 3DES keys, the check digits are the first four digits that result from encrypting zeros using the working key. If option 88 is enabled, this field will contain six check digits. For AES keys, the check digits will be generated based on the value specified in field 7 of the command.

Table 11-3 Response 25E: Combine Key Components

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	25E
1	Header,$E_{KEK.E}$(Working Key),MAC	58, 74, 90	printable ASCII
2	Header,$E_{MFK.E}$(Working Key),MAC	58, 74, 90	printable ASCII
3	Working Key Check Digits	4, 6, 10, or 64	0 - 9, A - F

Usage Notes

- Generate the KEK in AKB format.
- All key components must be the same length. An error <03xx...#> will be returned which points to the field in the command that contains a component whose length is not equal to the length of component 1. The detailed error code is 209.
- The length of the KEK must be equal to or greater than the length of the key components. If the length of any of the key components is greater than the length of the KEK, an error <00#0301....#> will be returned. The detailed error code is 209.
- All key components supplied in the command must have the same header. An error <07xx...#> will be returned which points to the field in the command that contains a header that does match the header supplied in component 1. The detailed error code is 607.
- If the combination of the key components produces a weak or semi-weak key (for a list of these keys, see Table A-1 on page A-8), an error<00#0600....#> will be returned. The detailed error is 513.
- If the key component contains an invalid value in header byte 6, an error <00#07...#> will be returned. The error points to the field in the command that contains the invalid value. The detailed error code is 607.
- The resulting Working Key will not be adjusted to odd parity.
- Certain commands require Working Keys have specific values in header byte 6. The key component header byte 6 value determines the Working Key AKB header byte 6 value, per the following table:

Description	**Component Header Byte 6 Value**	**Key Header Byte 6 Value**
Generic key component	C	0
Number of key components required	2, 3, 4	0
KEK for use in command 11B	i (lowercase i)	I (capital I)
KEK for use in command 11B	k (lowercase k)	K
Component of a KEK with E in byte 6	e (lowercase e)	E
Component of a key with N in byte 6	n (lowercase n)	N
Component of a derivation key	1	d (lowercase d)

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Combine two 2key-3DES key components into a key

- The 2key-3DES KEK:0123 4567 89AB CDEF FEDC BA98 7654 3210, check digits = 08D7
  The KEK in AKB format:
  1KDEE000,00B12295FA353767B3E2F92C51F8C4799EE072B479C8C889,815074BF95E27120

- The key component 1: 50530C9BA2576382 12C626AF058B433B, check digits = BD27
  The key component 1 in AKB format:
  1V3NE020,78D3AFA8EB3C65A9D565BCE7332623E0E70A8B3DF39BA6D4,FBAAF95D9C00D0AC

- The key component 2: 517049FC2BFCAE6D EC1A9C3773DF712B, check digits = 2F73
  The key component 2 in AKB format:
  1V3NE020,2070D8EED085BC71CBA8942BC3EBB59DA237FEDA00048B93,8424EFF3C6E3652A

The command looks like this:

```
<15E#1KDEE000,00B12295FA353767B3E2F92C51F8C4799EE072B479C8C889,8150
74BF95E27120#1V3NE020,78D3AFA8EB3C65A9D565BCE7332623E0E70A8B3DF39BA
6D4,FBAAF95D9C00D0AC#1V3NE020,2070D8EED085BC71CBA8942BC3EBB59DA237F
EDA00048B93,8424EFF3C6E3652A#####>
```

The Network Security Processor returns the following response:

```
<25E#1V3NE000,C7109E831DA566D3A5001C1F1C3522D659090D0D49FA34EF,37A5
160CED7C9435#1V3NE000,1C33AD320C016E5DE2F175AD10450BE007E0FDF5A97A5
1D6,86218BFF2ECD9D25#08D7#>
```

Combine four AES-128 key components into a key

- The 2key-3DES KEK: FEDC BA98 7654 3210 0123 4567 89AB CDEF, check digits = 7B83
  The KEK in AKB format:
  1KDEN000,37643D400AA790817C5D389B33C1E5D63F82A1E6DCA16B56,441968350015092A

- AES-128 Key Component 1: C0771E7B8A06AFD7 5C86BA5B18A8E47D
  Key component 1 in AKB format:
  1DJNE0n0,4202B00D386722BF044373E3BEBBD486,4AEFA5847A9B499C

- AES-128 Key Component 2: 91A496C91E896C03 77AA3C2D7FE98B12
  Key component 2 in AKB format:
  1DJNE0n0,FDAC89588ACF24997C8E2413AE79AD02,A313AFF7ED74A338

- AES-128 Key Component 3: C24B0D15B853F4FA F5F2A56688F45B14
  Key component 3 in AKB format:
  1DJNE0n0,AC2A85447E53336B2087CEFE4F0B598C,7DB038095945A97A

- AES-128 Key Component 4: 89BB4172CA2BBE3E DFFD6677661EF994
  Key component 4 in AKB format:
  1DJNE0n0,B5F79FF03357B2201FB8141BB7B24618,048AE4B3CFE01A7D

CMAC - check digits

The command looks like this:

```
<15E#1KDEN000,37643D400AA790817C5D389B33C1E5D63F82A1E6DCA16B56,4419
68350015092A#1DJNE0n0,4202B00D386722BF044373E3BEBBD486,4AEFA5847A9B
499C#1DJNE0n0,FDAC89588ACF24997C8E2413AE79AD02,A313AFF7ED74A338#1DJ
NE0n0,AC2A85447E53336B2087CEFE4F0B598C,7DB038095945A97A#1DJNE0n0,B5
F79FF03357B2201FB8141BB7B24618,048AE4B3CFE01A7D##C#>
```

The Network Security Processor returns the following response:

```
<25E#1DJNE0N0,1D4BA18A3FB3516F45ED9ACE7BD2322D,560153D5633A6CCC#1DJNE
0N0,8EB3BC6D992B383180D1564FED32C533,5A9D0434E5DC5E5E#75C6F11844#>
```

SHA256 - check digits

The command looks like this:

```
<15E#1KDEN000,37643D400AA790817C5D389B33C1E5D63F82A1E6DCA16B56,4419
68350015092A#1DJNE0n0,4202B00D386722BF044373E3BEBBD486,4AEFA5847A9B
499C#1DJNE0n0,FDAC89588ACF24997C8E2413AE79AD02,A313AFF7ED74A338#1DJ
NE0n0,AC2A85447E53336B2087CEFE4F0B598C,7DB038095945A97A#1DJNE0n0,B5
F79FF03357B2201FB8141BB7B24618,048AE4B3CFE01A7D##H#>
```

The Network Security Processor returns the following response:

```
<25E#1DJNE0N0,1D4BA18A3FB3516F45ED9ACE7BD2322D,560153D5633A6CCC#1DJ
NE0N0,8EB3BC6D992B383180D1564FED32C533,5A9D0434E5DC5E5E#BC50FEC934B
262D29388760B701B857A01D3EF87037D1DBD4542F565D4E41E9B#>
```

Print PIN Letter (Command 161)

This command is used to send a PIN letter print job to the printer.

warning The print job will contain the cleartext PIN. Appropriate security measures are required to ensure that only authorized personnel have access to the printer, and that communications between the Network Security Processor and the printer are not monitored.

This command is not enabled in the Network Security Processor's default security policy. It is only allowed on the NIC1 Print Command Port.

note It is highly recommended that this command be enabled for a specific number of executions. For information on how to configure the Network Security Processor to limit how many times this command can be executed, see the Command Count feature which is documented in section 4 of the *Atalla Secure Configuration Assistant-3 User Guide*.

The host application creates a PIN letter template (as a standard ASCII text file or Microsoft Word document). The PIN letter template must contain a unique 12 character PIN marker string value, for example "123456789012" or "xxxxxxxxxxxx"). The PIN marker string indicates where the cleartext PIN will be inserted into the PIN letter template; it must be present only once.

The host application uses this command to send the PIN letter template to the Network Security Processor along with the encrypted PIN. The encrypted PIN is encrypted under a PIN Printing Key. The Network Security Processor decrypts the PIN, searches the PIN letter template for the PIN marker string, and then replaces the PIN marker string with the cleartext PIN value (right padded with spaces, if necessary). The Network Security Processor then sends the PIN letter print job to the printer.

The maximum size of the PIN letter template is 1,048,576 bytes (1 megabyte). If the PIN letter template is larger than 30,000 bytes, the host application must split it into separate data blocks, and sends each data block as a separate command to the Network Security Processor. When a PIN letter template is split into multiple data blocks, information about the PIN (i.e. PIN block type, PIN Encryption Key, encrypted PIN block, etc.), and the PIN marker string must be included in only the final command.

The Network Security Processor can receive a maximum of four concurrent multi-command PIN letter templates. The Network Security Processor's response to the first command in a multi-command sequence will include a continuation index that must be provided in the subsequent intermediate and final commands required to send the remainder of the PIN letter template to the Network Security Processor. When the Network Security Processor receives the final data block of the PIN letter template, it will replace the PIN marker string with the clear PIN, and then send the complete PIN letter print job to the printer.

To reduce the PIN letter template size, company logos and other graphics should be preprinted on the paper that is loaded into the printer.

Command

```
<161#Letter Type#Continuation Flag#[Continuation Index]#
[PIN Block Type]#[EKPP(PIN Block)]#Reserved#[Header,EMFK.E(KPP),MAC]#
[PIN Block Digits]#[PIN Marker String]#Data Encoding#Data Type#
Letter Template Size#Data Block Length#Data Block#>
```

Response

```
<261#[Continuation Index]#[KPP Check Digits]# [PIN Sanity Error]#>
[CRLF]
```

Calling Parameters

`161`

Field 0, the command identifier.

`Letter Type`

Field 1, this field is specifies the type of letter to be printed.

Specify a letter type of 0 (zero) to print a test page. The following restrictions apply to printing a test page: field 2 must be the number 0 (zero), fields 3 through 9 must be empty, and field 10 must be the letter A.

To print a PIN letter, specify a letter type value of 1.

`Continuation Flag`

Field 2, the continuation flag. The following table defines the allowed values:

Value	Description
0	Entire PIN letter template is included in this command
1	The command contains the first block of a multi-block PIN letter template
2	The command contains an intermediate block of a multi-block PIN letter template
3	The command contains the final block of a multi-block PIN letter template
4	Cancel current print job; removes a partial PIN letter template from Network Security Processor’s memory

`[Continuation Index]`

Field 3, this index specifies which of the four internal memory storage locations the Network Security Processor used to store the first PIN letter template data block. This field must be empty when the continuation flag (field 2) is set to a value of 0 or 1. This field must be empty if the command is used to send the first data block of the PIN letter template. For subsequent commands used to send intermediate and final data blocks, the value of this field must match the value returned in field 1 of the response to the command that was used to send the first data block of the PIN letter template. When the continuation flag (field 2) is set to a value of 2, 3, or 4, this field can contain the values 0, 1, 2, or 3.

`[PIN Block Type]`

Field 4, the incoming PIN block format. This field must contain one of these values:

Value	Description
1	ANSI (ISO-0) Format PIN Block
8	ISO-3 Format PIN Block

This field should be empty when the continuation flag (field 2) is set to a value of 1, 2, or 4.

`[E`$_{KPP}$`(PIN Block)]`

Field 5, the ANSI PIN Block or ISO-3 PIN Block encrypted under a PIN Printing Key. This field contains a 16 hexadecimal digit value. This field should be empty when the continuation flag (field 2) is set to a value of 1, 2, or 4.

`Reserved`

Field 6, this field is reserved for future use. This field must be empty.

`[Header,E`$_{MFK.E}$`(KPP),MAC]`

Field 7, the PIN Printing Key encrypted under the MFK. This field contains a 74 byte value. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. It can contain a 3key-3DES key only if the MFK is also a 3key-3DES key, otherwise it contains
a 2key-3DES key. The following headers are supported: 1PUpE000, and 1PUpN000. This field should be empty when the continuation flag (field 2) is set to a value of 1, 2, or 4.

`[PIN Block Digits]`

Field 8, the account number digits used to format the ANSI or ISO-3 PIN block. This field contains 12 numeric digits. This field should be empty when the continuation flag (field 2) is set to a value of 1, 2, or 4.

`[PIN Marker String]`

Field 9, the 12 character PIN marker string in the print letter template that identifies the location where the cleartext PIN will be printed. This field can contain upper and lower case letters (A-Z, a-z) and numeric digits (0-9). When printed in the letter, the PIN will be left justified and space filled. For example, a five digit PIN will print in the leftmost 5 positions followed by 7 spaces. This field should be empty when the continuation flag (field 2) is set to a value of 1, 2, or 4.

`Data Encoding`

Field 10, the encoding used for the PIN marker string in the letter template file. This field can contain one of these values:

Value	Description
A	ASCII encoding, where 1234 = 0x31323334
W	Windows encoding (16-char, little endian) where 1234 = 0x3100320033003400

`Data Type`

Field 11, only binary data is supported. This field must contain the letter B.

`Letter Template Size`

Field 12, the size of the complete PIN letter template. The maximum size of the PIN letter template is 1,048,576 bytes (1 megabyte).

`Data Block Length`

Field 13, the number of bytes of the data sent in this data block. The maximum value is 30000.

`Data Block`

Field 14, the binary data of the PIN letter template. The maximum amount of binary data is 30000 bytes.

Table 11-4 Command 161: Print PIN Letter

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	161
1	Letter Type	1	0, 1
2	Continuation Flag	0, 1	0 - 4
3	[Continuation Index]	0, 1	0 - 3
4	[PIN Block Type]	0, 1	1, 8
5	[E_{KPP}(PIN Block)]	0, 16	0 - 9, A - F
6	Reserved	0	n/a
7	[Header,$E_{MFK.E}$(KPP),MAC]	0, 74	printable ASCII
8	[PIN Block Digits]	0, 12	0 - 9
9	[PIN Marker String]	0, 12	0 - 9, A - Z, a - z
10	Data Encoding	1	A, W

Table 11-4 Command 161: Print PIN Letter (continued)

Field	Contents	Length (bytes)	Legal Characters
11	Data Type	1	B
12	Letter Template Size	1, 7	0 - 1048576
13	Data Block Length	1 - 5	0 - 30000
14	Data Block	1 - 30000	binary

Responding Parameters

```
261
```

Field 0, the response identifier.

```
[Continuation Index]
```

Field 1, this field will contain a value in the range of 0 through 3 if the continuation flag (command-field 2) contains the number 1. It will match field 3 of the command if the continuation flag (command-field 2) is 2, 3, or 4. It will be empty if the continuation flag is 0.

```
[KPP Check Digits]
```

Field 2, the PIN Printing Key check digits. The first four digits that result from encrypting zeros using the KPE. If option 88 is enabled, this field contains six check digits. This field will be empty when the continuation flag (command-field 2) is 1, 2, or 4.

```
[PIN Sanity Error]
```

Field 3, the PIN sanity error. This field will be empty when the continuation flag (command-field 2) is 1, 2, or 4. If the Network Security Processor is able to successfully decrypt the encrypted PIN block (command-field 5), this field will be empty. If the Network Security Processor is unable to correctly decrypt the encrypted PIN block, or if the length of the decrypted PIN is less than the value defined in option A0 or greater than 12, this field will contain either the letter S or L based on how option A1 is configured.

Table 11-5 Response 261: Print PIN Letter

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	261
1	[Continuation Index]	0, 1	0 - 3
2	[KPP Check Digits]	0,4,6	0 - 9, A - F
3	[PIN Sanity Error]	0,1	S, L

Usage Notes

When a letter template file requires multiple commands to process the entire letter template file, the Network Security Processor will clear the entire letter template file from its memory on any of the following error conditions. In this case, correct the error and send all of the commands required to process the entire letter template file again.

- An invalid letter template length. The total number of bytes received is not equal to the total number of bytes specified in the Letter Template Size (field 12) of the command.
- An invalid PIN Printing Key specified in field 7. The Network Security Processor will return an error code 07.
- A TCP/IP connection or send/receive error is detected. The Network Security Processor will return an error code 11.
- A Network Security Processor execution error. The Network Security Processor will return an error code 08.
- The marker string is not present in the template file. The Network Security Processor will return an error code 12.
- The decrypted PIN block fails the sanity test. The Network Security Processor will return either an S or L in field 3 of the response.

Command syntax errors, such as invalid number of fields in a command or an invalid character in a command, will not cause the Network Security Processor to erase the letter template file from its internal memory slot. In this case, correct the syntax error and send the command again to print the PIN letter.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

- 2key-3DES KPP:0123 4567 89AB CDEF FEDC BA98 7654 3210,
  check digits = 08D7
  The KPP in AKB format:1PUpE000,EEDCE81076897C9CBF94F42ED98B152A2
  133F725BC1E171E,E9FB28B06FE60D79
- Clear PIN to be printed: 1234
- Clear ANSI PIN Block: 041262876FEDCBA9
- Encrypted ANSI PIN Block: E08962A98076BF5C
- PIN Marker String: xxxxxxxxxxxx

ASCII PIN Letter Template Text

```
Mr John Smith

1234 Main Street
```

```
Anytown, CA, 123456



Dear Mr Smith,


Your new PIN is : xxxxxxxxxxxx


Please keep your PIN safe.


Regards,


AnyBank
```

The command looks like this. For visibility purposes, in this example the binary data in the data block field (field 14) is presented in hexadecimal format.

```
<161#1#0##1#E08962A98076BF5C##1PUpE000,EEDCE81076897C9CBF94F42ED98B
152A2133F725BC1E171E,E9FB28B06FE60D79#567890123456#xxxxxxxxxxxx#A#B
#162#162#4D72204A6F686E20536D6974680D0A31323334204D61696E2053747265
65740D0A416E79746F776E2C2043412C203132333435360D0A0D0A0D0A446561722
04D7220536D6974682C0D0A0D0A596F7572206E65772050494E206973203A207878
787878787878787878200D0A0D0A506C65617365206B65657020796F757220504
94E20736166652E0D0A0D0A526567617264732C0D0A0D0A416E7942616E6B0D0A#>
```

The Network Security Processor returns the following response:

```
<261##08D7##>
```

PIN Issuance: IBM 3624 Method (Command 162)

This command can generate or calculate a PIN and IBM 3624 offset. Three modes of operations are supported:

- Calculate an offset from an encrypted PIN block
- Generate a random PIN and calculate the offset
- Calculate the PIN from an offset

The response to the command will contain an encrypted PIN block encrypted under a PIN Printing Key. Use command 161 to print the encrypted PIN.

This command is not enabled in the Network Security Processor's default security policy. It is only allowed on the NIC1 print command port.

note It is highly recommended that this command be enabled for a specific number of executions. For information on how to configure the Network Security Processor to limit how many times this command can be executed, see the Command Count feature which is documented in section 4 of the *Atalla Secure Configuration Assistant-3 User Guide*.

Command

```
<162#Algorithm#Mode#PIN Block Format#[EKPE(PIN)]#
[Header,EMFK.E(KPE),MAC]#Header,EMFK.E(KPP),MAC#
Header,EMFK.E(KPV),MAC#[Offset]#[PIN Length]#Conversion Table#
Validation Data#Pad#PIN Block Data#>
```

Response

```
<262#EKPP(PIN Block)/Sanity Error#[Offset]#>[CRLF]
```

Calling Parameters

`162`

Field 0, the command identifier.

`Algorithm`

Field 1, the PIN algorithm. This field must contain the number 2.

`Mode`

Field 2, the mode of operation. This field must contain one of these values:

Value	Description
1	Calculate an offset from an encrypted PIN block
2	Generate a random PIN and calculate the offset
3	Calculate the PIN from an offset

`PIN Block Format`

Field 3, the PIN block format. This field must contain one of these values:

Value	Description
1	ANSI PIN Block (ISO-0)
8	ISO-3 PIN Block

[E_{KPE}(PIN Block)]

Field 4, the ANSI or ISO-3 PIN block encrypted under the PIN encryption key. This field must contain a 16 hexadecimal digit value when the mode (field 2) is 1. In all other cases, this field must be empty.

[Header,$E_{MFK.E}$(KPE),MAC]

Field 5, the PIN Encryption Key (KPE) used to encrypt the PIN supplied in field 4. This field must contain a 74 byte value when the mode (field 2) is 1. In all other cases, this field must be empty. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. It can be a 3key-3DES key only if the MFK is also a 3key-3DES key, otherwise it contains
a 2key-3DES key. The following headers are supported: 1PUNE000 and 1PUNN000.

Header,$E_{MFK.E}$(KPP),MAC

Field 6, the PIN Printing Key used to encrypt the PIN returned in the response. This field contains a 74 byte value. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. It can be a 3key-3DES key only if the MFK is also a 3key-3DES key, otherwise it contains a 2key-3DES key. The following headers are supported: 1PUpE000 and 1PUpN000.

Header,$E_{MFK.E}$(KPV),MAC

Field 7, the PIN Verification Key (KPV). This field contains a 74 byte value. When option 4A is enabled, this field can contain a 1key-3DES (single-length) key. It can be a 3key-3DES key only if the MFK is also a 3key-3DES key, otherwise it contains a 2key-3DES key. The following headers are supported: 1V3NE000, 1V3NN000, 1V3GN000 and 1V3GE000.

`[Offset]`

Field 8, the PIN offset. This field must be empty when the mode (field 2) is 1 or 2. When the mode is 3, this field can contain a 4 through 12 digit numeric value or if empty, the Network Security Processor will generate an offset of all zeros equal to the PIN length.

`[PIN Length]`

Field 9, the PIN length. This field must be empty when the mode (field 2) is 1. When the mode is 2 or 3, this field contains the length of the PIN or offset to be calculated. If present, this field must contain a numeric value in the range of 4 through 12.

`Conversion Table`

Field 10, a table that maps hexadecimal digits (0 through 9, A through F) to decimal digits (0 through 9). This field contains the 16 decimal digit value of the clear-text conversion table. When option 48 is enabled, this field contains the conversion table in AKB format (the header must be 1nCNE000). When option 4E is enabled, the conversion table must adhere to these rules:

- The conversion table must have at least eight unique digits.
- No single digit can occur more than four times.

`Validation Data`

Field 11, validation data. This value is unique for each card holder, and is typically the account number. This field contains a 4 to 16 byte hexadecimal value. When option 4C is enabled, the value supplied in this field must be 12 digits in length and match the PIN Block Data value supplied in field 13.

`Pad`

Field 12, the pad character which right-pads the validation data. This field contains a one byte hexadecimal value. The pad character is only applied when the validation data is less than 16 characters in length.

`PIN Block Data`

Field 13, the account number digits used to format the ANSI or ISO-3 PIN block. This field contains 12 numeric digits.

Table 11-6 Command 162: PIN Issuance: IBM 3624 Method

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	162
1	Algorithm	1	2
2	Mode	1	1 - 3
3	PIN Block Format	1	1, 8

Table 11-6 Command 162: PIN Issuance: IBM 3624 Method (continued)

Field	Contents	Length (bytes)	Legal Characters
4	[E_{KPE}(PIN)]	0,16	0 - 9, A - F
5	[Header,$E_{MFK.E}$(KPE),MAC]	0,74	printable ASCII
6	Header,$E_{MFK.E}$(KPP),MAC	74	printable ASCII
7	Header,$E_{MFK.E}$(KPV),MAC	74	printable ASCII
8	[Offset]	0, 4-12	0 - 9
9	[PIN Length]	0, 1-2	4-12
10	Conversion Table	16, 74	0 - 9, printable ASCII
11	Validation Data	4-16	0 - 9, A - F
12	Pad	1	0 - 9, A - F
13	PIN Block Data	12	0 - 9

Responding Parameters

`262`

Field 0, the response identifier.

`E`$_{KPP}$`(PIN Block)/Sanity Error`

If the PIN block (command-field 4) passes the PIN sanity test, this field will contain the ANSI or ISO-3 PIN block encrypted under the PIN Printing Key. If the PIN block fails the sanity test, this field will contain a sanity error. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. Sanity errors are:

- S – PIN failed the sanity test. Or the length of the PIN is out of range, and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
- L – the length of the PIN is out of range

`[Offset]`

Field 2, the offset. This field will contain the offset when the mode (command-field 2) is 1 or 2 and field 1 of the response does not contain a sanity error. When the mode is 3, this field will be empty.

Table 11-7 Response 262: PIN Issuance: IBM 3624 Method

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	262

Table 11-7 Response 262: PIN Issuance: IBM 3624 Method (continued)

Field	Contents	Length (bytes)	Legal Characters
1	E_{KPP}(PIN Block) / Sanity Error	16, 1	0 - 9, A - F, S, L
2	[Offset]	0, 4 - 12	0 - 9

Usage Notes

- Generate the PIN Printing Key.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Mode 1 - Calculate an offset from an encrypted ANSI PIN block

- PIN block type: ANSI (1)
- PAN: 5555557890123456
- ANSI PIN block: 041261A876FEDCBA, clear PIN = 1234
  The encrypted ANSI PIN block: F81BD4D6E8AC404E
- PIN Encryption Key (KPE): 0123456789ABCDEF FEDCBA9876543210
  The PIN Encryption Key (KPE) in AKB format:
  1PUNE000,27F0026930C71646A3ADED3356C36BB0CCC2D6874DE6AB43,83237B4AEA731054
- PIN Printing Key (KPP):FEDCBA9876543210 0123456789ABCDEF
  The PIN Printing Key in AKB format:
  1PUpE000,300863A6F83437CC88B92FDCE8F5270F4867F1A10EC6282B,02AAEE7033A5B27A
- PIN Verification Key (KPV): 1234123412341234 5678567856785678
  The PIN Verification Key in AKB format:
  1V3NE000,F3CED719DC65AC7F086FB746C11E30E7AE5F50D746C9FA7D,D90C6B8D04DB19EB
- Conversion table: 0123456789012345
- Validation data: 7890123456
- Pad character: F
- PIN block data: 555789012345

The command looks like this:

```
<162#2#1#1#F81BD4D6E8AC404E#1PUNE000,27F0026930C71646A3ADED3356C36B
B0CCC2D6874DE6AB43,83237B4AEA731054#1PUpE000,300863A6F83437CC88B92F
DCE8F5270F4867F1A10EC6282B,02AAEE7033A5B27A#1V3NE000,F3CED719DC65AC
7F086FB746C11E30E7AE5F50D746C9FA7D,D90C6B8D04DB19EB###0123456789012
345#7890123456#F#555789012345#>
```

The Network Security Processor returns the following response:

```
<262#F81BD4D6E8AC404E#3953#>
```

Mode 2 - Generate a 4 digit PIN and calculate the offset

- PIN block type: ANSI (1)
- PAN: 5555557890123456
- PIN Printing Key (KPP):FEDCBA9876543210 0123456789ABCDEF
  The PIN Printing Key in AKB format:
  1PUpE000,300863A6F83437CC88B92FDCE8F5270F4867F1A10EC6282B,02AAEE703 3A5B27A
- PIN Verification Key (KPV): 1234123412341234 5678567856785678
  The PIN Verification Key in AKB format:
  1V3NE000,F3CED719DC65AC7F086FB746C11E30E7AE5F50D746C9FA7D,D90C6B8D 04DB19EB
- PIN Length: 4
- Conversion table: 0123456789012345
- Validation data: 7890123456
- Pad character: F
- PIN block data: 555789012345

The command looks like this:

```
<162#2#2#1###1PUpE000,300863A6F83437CC88B92FDCE8F5270F4867F1A10EC62
82B,02AAEE7033A5B27A#1V3NE000,F3CED719DC65AC7F086FB746C11E30E7AE5F5
0D746C9FA7D,D90C6B8D04DB19EB##4#0123456789012345#7890123456#F#55578
9012345#>
```

The Network Security Processor generates a random PIN the response will be similar to this:

```
<262#11455BBB3ED429A0#3953#>
```

Mode 3 - Calculate the PIN from an offset

- PIN block type: ANSI (1)
- PAN: 5555557890123456

- PIN Printing Key (KPP):FEDCBA9876543210 0123456789ABCDEF
  The PIN Printing Key in AKB format:
  1PUpE000,300863A6F83437CC88B92FDCE8F5270F4867F1A10EC6282B,02AAEE703 3A5B27A

- PIN Verification Key (KPV): 1234123412341234 5678567856785678
  The PIN Verification Key in AKB format:
  1V3NE000,F3CED719DC65AC7F086FB746C11E30E7AE5F50D746C9FA7D,D90C6B8D 04DB19EB

- PIN Length: 4

- Conversion table: 0123456789012345

- Validation data: 7890123456

- Pad character: F

- PIN block data: 555789012345

The command looks like this:

```
<162#2#3#1###1PUpE000,300863A6F83437CC88B92FDCE8F5270F4867F1A10EC62
82B,02AAEE7033A5B27A#1V3NE000,F3CED719DC65AC7F086FB746C11E30E7AE5F5
0D746C9FA7D,D90C6B8D04DB19EB#3953#4#0123456789012345#7890123456#F#5
55789012345#>
```

The Network Security Processor returns the following response:

```
<262#11455BBB3ED429A0##>
```

PIN Issuance: Visa Method (Command 163)

This command can generate or calculate a PIN and Visa PIN Verification Value. Two modes of operations are supported:

- Calculate the PIN Verification Value (PVV) from an encrypted PIN block.
- Generate a random PIN and calculate the PIN Verification Value (PVV).

The response to the command will contain an encrypted PIN block encrypted under a PIN Printing Key. Use command 161 to print the encrypted PIN.

This command is not enabled in the Network Security Processor's default security policy. It is only allowed on the NIC1 print command port.

note It is highly recommended that this command be enabled for a specific number of executions. For information on how to configure the Network Security Processor to limit how many times this command can be executed, see the Command Count feature which is documented in section 4 of the *Atalla Secure Configuration Assistant-3 User Guide*.

Command

```
<163#Algorithm#Mode#PIN Block Format#[$E_{KPE}$(PIN)]#
[Header,$E_{MFK.E}$(KPE),MAC]#Header,$E_{MFK.E}$(KPP),MAC#
Header,$E_{MFK.E}$(KPV),MAC#[PIN Length]#PVKI#Validation Data#
PIN Block Data#>
```

Response

```
<263#$E_{KPP}$(PIN Block)/Sanity Error#[PVV]#>[CRLF]
```

Calling Parameters

`163`

Field 0, the command identifier.

`Algorithm`

Field 1, the PIN algorithm. This field must contain the number 3.

`Mode`

Field 2, the mode of operation. This field must contain one of these values:

Value	Description
1	Calculate the PIN Verification Value (PVV) from an encrypted PIN block
2	Generate a random PIN and calculate the PIN Verification Value (PVV)

`PIN Block Format`

Field 3, the PIN block format. This field must contain one of these values:

Value	Description
1	ANSI PIN Block (ISO-0)
8	ISO-3 PIN Block

[$\mathtt{E_{KPE}}$(PIN Block)]

Field 4, the ANSI or ISO-3 PIN block encrypted under the PIN encryption key. This field must contain a 16 hexadecimal digit value when the mode (field 2) is 1. If the decrypted PIN contains a PIN that is more than 4 digits in length, the Network Security Processor will use only the leftmost 4 digits to calculate the PVV. When the mode (field 2) is 2, this field must be empty.

[Header,$\mathtt{E_{MFK.E}}$(KPE),MAC]

Field 5, the PIN Encryption Key (KPE) used to encrypt the PIN supplied in field 4. This field must contain a 74 byte value when the mode (field 2) is 1. When the mode (field 2) is 2, this field must be empty. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. It can be a 3key-3DES key only if the MFK is also a 3key-3DES key, otherwise it contains a 2key-3DES key. The following headers are supported: 1PUNE000 and 1PUNN000.

Header,$\mathtt{E_{MFK.E}}$(KPP),MAC

Field 6, the PIN Printing Key used to encrypt the PIN returned in the response. This field contains a 74 byte value. When option 6C is enabled, this field can contain a 1key-3DES (single-length) key. It can be a 3key-3DES key only if the MFK is also a 3key-3DES key, otherwise it must be a 2key-3DES key. The following headers are supported: 1PUpE000 and 1PUpN000.

Header,$\mathtt{E_{MFK.E}}$(KPV),MAC

Field 7, the PIN Verification Key (KPV). This field contains a 74 byte value. When option 4A is enabled, this field can contain a 1key-3DES (single-length) key. It can be a 3key-3DES key only if the MFK is also a 3key-3DES key, otherwise it must be a 2key-3DES key. The following headers are supported: 1VVNE000, 1VVNN000, 1VVGE000 and 1VVGN000.

`[PIN Length]`

Field 8, the PIN length. This field must be empty when the mode (field 2) contains the number 1. This field must contain the number 4 when the mode (field 2) contains the number 2.

`PVKI`

Field 9, the PIN Verification Key Indicator (PVKI) used in the algorithm to calculate the PVV. This field contains a 1 byte decimal value in the range of 0 to 9.

`Validation Data`

Field 10, validation data. This value is unique for each card holder, and is typically a portion of the account number. This field contains an 11 digit numeric value. When option 4C is enabled, these 11 digits must be present in the PIN block data value supplied in field 11.

`PIN Block Data`

Field 11, the account number digits used to format the ANSI or ISO-3 PIN block. This field contains 12 numeric digits.

Table 11-8 Command 163: PIN Issuance: Visa Method

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	163
1	Algorithm	1	3
2	Mode	1	1, 2
3	PIN Block Format	1	1, 8
4	[E_{KPE}(PIN)]	0,16	0 - 9, A - F
5	[Header,$E_{MFK.E}$(KPE),MAC]	0,74	printable ASCII
6	Header,$E_{MFK.E}$(KPP),MAC	74	printable ASCII
7	Header,$E_{MFK.E}$(KPV),MAC	74	printable ASCII
8	[PIN Length]	0,1	4
9	PVKI	1	0 - 9
10	Validation Data	11	0 - 9
11	PIN Block Data	12	0 - 9

Responding Parameters

`263`

Field 0, the response identifier.

$\texttt{E}_{\texttt{KPP}}$`(PIN Block)/Sanity Error`

Field 1, if the PIN block (command-field 4) passes the PIN sanity test, this field will contain the ANSI or ISO-3 PIN block encrypted under the PIN Printing Key. If the PIN block fails the sanity test, this field will contain a sanity error. Option 4B specifies the type of PIN sanity test to be performed on the incoming PIN block. Sanity errors are:

- S – PIN failed the sanity test. Or the length of the PIN is out of range, and PIN-length error reporting has not been enabled. See PIN Sanity Error and option A1.
- L – the length of the PIN is out of range.

`[PVV]`

Field 2, the 4 numeric digit PIN Verification Value. This field will be empty if a sanity error is present in field 1 of the response.

Table 11-9 Response 263: PIN Issuance: Visa Method

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	263
1	E_{KPP}(PIN Block) / Sanity Error	16, 1	0 - 9, A - F, S, L
2	[PVV]	0,4	0 - 9

Usage Notes

- Generate the PIN Printing Key.

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Mode 1 - Calculate the PVV from an encrypted ANSI PIN block

- PIN block type: ANSI (1)
- PAN: 5555557890123456
- ANSI PIN block: 041261A876FEDCBA, clear PIN = 1234
  The encrypted ANSI PIN block: F81BD4D6E8AC404E
- PIN Encryption Key (KPE): 0123456789ABCDEF FEDCBA9876543210
  The PIN Encryption Key (KPE) in AKB format:
  1PUNE000,27F0026930C71646A3ADED3356C36BB0CCC2D6874DE6AB43,83237B4A
  EA731054

- PIN Printing Key (KPP):FEDCBA9876543210 0123456789ABCDEF
  The PIN Printing Key in AKB format:
  1PUpE000,300863A6F83437CC88B92FDCE8F5270F4867F1A10EC6282B,02AAEE703 3A5B27A

- PIN Verification Key (KPV): 1234123412341234 5678567856785678
  The PIN Verification Key in AKB format:
  1VVNE000,E38AA6361D2591C14D48F356C77E116C686F2F5BDF3DB14A,7BC06F42C 82FE0C2

- PIN Length: empty

- PVKI: 1

- Validation data: 55789012345

- PIN block data: 555789012345

The command looks like this:

```
<163#3#1#1#F81BD4D6E8AC404E#1PUNE000,27F0026930C71646A3ADED3356C36B
B0CCC2D6874DE6AB43,83237B4AEA731054#1PUpE000,300863A6F83437CC88B92F
DCE8F5270F4867F1A10EC6282B,02AAEE7033A5B27A#1VVNE000,E38AA6361D2591
C14D48F356C77E116C686F2F5BDF3DB14A,7BC06F42C82FE0C2##1#55789012345#
555789012345#>
```

The Network Security Processor returns the following response:

```
<263#F81BD4D6E8AC404E#0177#>
```

Mode 2 - Generate a 4 digit PIN and calculate the PVV

- PIN block type: ANSI (1)

- PAN: 5555557890123456

- PIN Printing Key (KPP): FEDCBA9876543210 0123456789ABCDEF
  The PIN Printing Key in AKB format:
  1PUpE000,300863A6F83437CC88B92FDCE8F5270F4867F1A10EC6282B,02AAEE703 3A5B27A

- PIN Verification Key (KPV): 1234123412341234 5678567856785678
  The PIN Verification Key in AKB format:
  1V3NE000,F3CED719DC65AC7F086FB746C11E30E7AE5F50D746C9FA7D,D90C6B8D 04DB19EB

- PIN Length: 4

- PVKI: 1

- Validation data: 55789012345

- PIN block data: 555789012345

The command looks like this:

```
<163#3#2#1###1PUpE000,300863A6F83437CC88B92FDCE8F5270F4867F1A10EC62
82B,02AAEE7033A5B27A#1VVNE000,E38AA6361D2591C14D48F356C77E116C686F2
F5BDF3DB14A,7BC06F42C82FE0C2#4#1#55789012345#555789012345#>
```

The Network Security Processor generates a random PIN the response will be similar to this:

```
<263#3B4196958319C64D#0439#>
```

Reset Printing Command Counter (Command 16A)

This command resets all the printing command counters to zero, and also disables these printing commands: 15E, 161, 162, 163, 16E, and 16F.

This command is not enabled in the Network Security Processor's default security policy.

This command is only allowed on the NIC1 Print Command Port.

Command

```
<16A#Operation#>
```

Response

```
<26A#Status#>[CRLF]
```

Calling Parameters

`16A`

Field 0, the command identifier.

`Operation`

Field 1, the operation to be performed. This field must contain the value 0 (zero) to reset printing command counters to zero.

Table 11-10 Command 16A: Reset Printing Command Counter

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	16A
1	Operation	1	0

Responding Parameters

`26A`

Field 0, the response identifier.

`Status`

Field 1, the result of the reset count operation. The values can be either:

Value	Description
OK	The printing command counters have been reset to zero.
FAILED	The printing counter has not been reset to zero.

Table 11-11 Response 26A: Reset Printing Command Counter

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	26A
1	Status	2 or 6	OK, FAILED

Usage Notes

- The host application can use this command to turn off and disable the printing commands.

Example

The command looks like this:

```
<16A#0#>
```

The Network Security Processor returns the following response:

```
<26A#OK#>
```

Divide a Key into Components (Command 16E)

This command divides a 3DES or AES key into multiple random key components. The key components are returned in AKB format. The minimum number of key components is 2 and the maximum number of key components is 4.

This command is not enabled in the Network Security Processor's default security policy. It is only allowed on the NIC1 print command port.

note It is highly recommended that this command be enabled for a specific number of executions. For information on how to configure the Network Security Processor to limit how many times this command can be executed, see the Command Count feature which is documented in section 4 of the *Atalla Secure Configuration Assistant-3 User Guide*.

Command

```
<16E#Header#Header,EMFK.E(Key),MAC#Number of Components#
AES Check Digit Method#>
```

Response

```
<26E#Key Check Digits# Header,EMFK.E(Comp-1),MAC#
Component 1 Check Digits#Header,EMFK.E(Comp-2),MAC#
Component 2 Check Digits#[Header,EMFK.E(Comp-3),MAC]#
[Component 3 Check Digits]# [Header,EMFK.E(Comp-4),MAC]#
[Component 4 Check Digits]#>[CRLF]
```

Calling Parameters

`16E`

Field 0, the command identifier.

`Header`

Field 1, the header to be applied to the generated components. The first 5 characters of this header must match the header of the key provided in field 2. Byte 6 of the header must contain one of the following values: C, 1, 2, 3, 4, i, e, k, or n.

`Header,EMFK.E(Key),MAC`

Field 2, the 3DES or AES key to be divided into components. When the key is either a 3DES or an AES-192 bit key, this field contains a 74 byte value. When the key is an AES-128 bit key, this field contains a 58 byte value. When the key is an AES-256 bit key, this field contains a 90 byte value.

`Number of Components`

Field 3, the number of key components to divide the key into. The minimum value is 2 and the maximum value is 4.

`AES Check Digit Method`

Field 4, the method used to calculate check digits for an AES key. For 3-DES keys, the Network Security Processor ignores this field. For AES keys, one of the following values must be present.

Value	Description
C	The key used to calculate a MAC of an all zero block using the CMAC algorithm. The leftmost 10 hexadecimal digits are the check digits.
H	SHA-256 hash of the key (64 hexadecimal digits)

Table 11-12 Command 16E: Divide a Key into Components

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	16E
1	Header	8	printable ASCII
2	Header,$E_{MFK.E}$(Key),MAC	58, 74, 90	printable ASCII
3	Number of components	1	2 - 4
4	AES Check Digit Method	0, 1	C, H

Responding Parameters

`26E`

Field 0, the response identifier.

`Key Check Digits`

Field 1, the check digits of the key. For 3DES keys, the check digits are the first four digits that result from encrypting zeros using the key. If option 88 is enabled, this field will contain six check digits. For AES keys, the check digits will be generated by the value specified in field 4 of the command.

Header,$E_{MFK.E}$(Comp-1),MAC

Field 2, the first key component encrypted under the MFK. When the key component is either a 3DES or an AES-192 bit key, this field contains a 74 byte value. When the key component is an AES-128 bit key, this field contain a 58 byte value. When the key component is an AES-256 bit key, this field contain a 90 byte value.

`Component 1 Check Digits`

Field 3, the check digits of the first key component. For 3DES key components, the check digits are the first four digits that result from encrypting zeros using the key component. If option 88 is enabled, this field will contain six check digits. For AES key components, the check digits will be generated by the value specified in field 3 of the command.

Header,$E_{MFK.E}$(Comp-2),MAC

Field 4, the second key component encrypted under the MFK. When the key component is either a 3DES or an AES-192 bit key, this field contains a 74 byte value. When the key component is an AES-128 bit key, this field contain a 58 byte value. When the key component is an AES-256 bit key, this field contain a 90 byte value.

`Component 2 Check Digits`

Field 5, the check digits of the second key component. For 3DES key components, the check digits are the first four digits that result from encrypting zeros using the key component. If option 88 is enabled, this field will contain six check digits. For AES key components, the check digits will be generated by the value specified in field 3 of the command.

[Header,$E_{MFK.E}$(Comp-3),MAC]

Field 6, the third key component encrypted under the MFK. This field will be empty when the number of key components specified in field 3 of the command contains a value less than 3. When the key component is either a 3DES or an AES-192 bit key, this field contains a 74 byte value. When the key component is an AES-128 bit key, this field contain a 58 byte value. When the key component is an AES-256 bit key, this field contain a 90 byte value.

`[Component 3 Check Digits]`

Field 7, the check digits of the third key component. This field will be empty when the number of key components specified in field 3 of the command contains a value less than 3. For 3DES key components, the check digits are the first four digits that result from encrypting zeros using the key component. If option 88 is enabled, this field will contain six check digits. For AES key components, the check digits will be generated by the value specified in field 3 of the command.

[Header,$E_{MFK.E}$(Comp-4),MAC]

Field 8, the fourth key component encrypted under the MFK. This field will be empty when the number of key components specified in field 3 of the command contains a value less than 4. When the key component is either a 3DES or an AES-192 bit key, this field contains a
74 byte value. When the key component is an AES-128 bit key, this field contain a 58 byte value. When the key component is an AES-256 bit key, this field contain a 90 byte value.

`[Component 4 Check Digits]`

Field 9, the check digits of the fourth key component. This field will be empty when the number of key components specified in field 3 of the command contains a value less than 4. For 3DES key components, the check digits are the first four digits that result from encrypting zeros using the key component. If option 88 is enabled, this field will contain six check digits. For AES key components, the check digits will be generated by the value specified in field 3 of the command.

Table 11-13 Response 26E: Divide a Key into Components

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	26E
1	Key Check Digits	4, 6, 10, or 64	0 - 9, A - F
2	Header,$E_{MFK.E}$(Comp-1),MAC	58, 74, 90	printable ASCII
3	Comp-1 Check Digits	4, 6, 10, or 64	0 - 9, A - F
4	Header,$E_{MFK.E}$(Comp-2),MAC	58, 74, 90	printable ASCII
5	Comp-2 Check Digits	4, 6, 10, or 64	0 - 9, A - F
6	[Header,$E_{MFK.E}$(Comp-3),MAC]	0, 58, 74, 90	printable ASCII
7	[Comp-3 Check Digits]	0, 4, 6, 10, or 64	0 - 9, A - F
8	[Header,$E_{MFK.E}$(Comp-4),MAC]	0, 58, 74, 90	printable ASCII
9	[Comp-4 Check Digits]	0, 4, 6, 10, or 64	0 - 9, A - F

Usage Notes

- Randomly generated key components are not adjusted to odd parity.
- Do not use this command to create key components for HMAC-SHA1 or HMAC-SHA256 keys that are longer than 176-bits.
- The Working Key AKB header byte 6 value determines the key component header byte 6 value, per the following table:

Description	Key Header Byte 6 Value	Component Header Byte 6 Value
Generic key component	0	C
Number of key components required	0	2, 3, 4
KEK for use in command 11B	I (capital I)	i (lowercase i)
KEK for use in command 11B	K	k (lowercase k)
Component of a KEK with E in byte 6	E	e (lowercase e)

Description	**Key Header Byte 6 Value**	**Component Header Byte 6 Value**
Component of a key with N in byte 6	N	n (lowercase n)
Component of a derivation key	d (lowercase d)	1

Examples

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Divide a 2key-3DES key into 2 components

- The key value: 0123456789ABCDEF FEDCBA9876543210, check digits = 08D7
  The key in AKB format:
  1V3NE000,1C33AD320C016E5DE2F175AD10450BE007E0FDF5A97A51D6,86218BFF2ECD9D25

The command looks like this:

```
<16E#1V3NE020#1V3NE000,1C33AD320C016E5DE2F175AD10450BE007E0FDF5A97A
51D6,86218BFF2ECD9D25#2##>
```

The Network Security Processor generates random component values, the response will be similar to this:

```
<26E#08D7#1V3NE020,78D3AFA8EB3C65A9D565BCE7332623E0E70A8B3DF39BA6D4,
FBAAF95D9C00D0AC#BD27#1V3NE020,2070D8EED085BC71CBA8942BC3EBB59DA237FE
DA00048B93,8424EFF3C6E3652A#2F73#####>
```

Divide an AES-128 key into 4 components

- The key value: 1A23C4D5E6F789100123456789ABCDEF check digits = 75C6F11844
  The Key in AKB format:
  1DJNE0N0,8EB3BC6D992B383180D1564FED32C533,5A9D0434E5DC5E5E
- CMAC check digits: 75C6F11844

The command looks like this:

```
<16E#1DJNE0n0#1DJNE0N0,8EB3BC6D992B383180D1564FED32C533,5A9D0434E5D
C5E5E#4#C#>
```

The Network Security Processor generates random component values, the response will be similar to this:

```
<26E#75C6F11844#1DJNE0n0,4202B00D386722BF044373E3BEBBD486,4AEFA5847
A9B499C#3C6DFC7B7B#1DJNE0n0,FDAC89588ACF24997C8E2413AE79AD02,A313AF
F7ED74A338#F4BB67F44A#1DJNE0n0,AC2A85447E53336B2087CEFE4F0B598C,7DB
038095945A97A#D001C51AB9#1DJNE0n0,B5F79FF03357B2201FB8141BB7B24618,
048AE4B3CFE01A7D#9FBDEA5BC1#>
```

Print Component Letter (Command 16F)

Command 16F is used to print a component letter for an existing component or a randomly generated component. This command supports both 3DES and AES key components.

warning The print job will contain the cleartext component. Appropriate security measures are required to ensure that only authorized personnel have access to the printer, and that communications between the Network Security Processor and the printer are not monitored.

This command is not enabled in the Network Security Processor's default security policy. It is only allowed on the NIC1 Print Command Port.

note It is highly recommended that this command be enabled for a specific number of executions. For information on how to configure the Network Security Processor to limit how many times this command can be executed, see the Command Count feature which is documented in section 4 of the *Atalla Secure Configuration Assistant-3 User Guide*.

The host application creates a component letter template (as a standard ASCII text file or Microsoft Word document). The component letter template must contain both a component marker string and a check digits marker string. These marker strings indicate where the cleartext component and check digits will be inserted into the template. An optional reference marker string is supported.

The host application uses this command to send the component letter template to the Network Security Processor along with the component (in AKB format). Or it can instruct the Network Security Processor to generate a random component. The Network Security Processor decrypts or generates the component, searches the component letter template for the component and check digit marker strings, and then replaces them with the cleartext component and check digit values, and optionally the reference value. The Network Security Processor sends the component letter print job to the printer.

To reduce the component letter template size, company logos and other graphics should be preprinted on the paper that is loaded into the printer.

Command

```
<16F#Letter Type#Continuation Flag#[Continuation Index]#[Header]#
[Header,EMFK.E(Component),MAC]#[Component Length]#
[Component Marker String]#[Check Digit Marker String]#
[Reference Marker String]#Data Encoding#Data Type#
Letter Template Size#Data Block Length#Data Block#>
```

Response

```
<26F#[Continuation Index]#[Header,EMFK.E(Component),MAC]#
Component Check Digits#>[CRLF]
```

Calling Parameters

`16F`

Field 0, the command identifier.

`Letter Type`

Field 1, this field is specifies the type of letter to be printed.

Specify a letter type of 0 (zero) to print a test page. The following restrictions apply to printing a test page: field 2 must be the number 0 (zero), fields 3 through 9 must be empty, and field 10 must be the letter A.

To print a component letter, specify a letter type value of 1.

`Continuation Flag`

Field 2, the continuation flag. The following table defines the allowed values.

Value	Description
0	Entire component letter template is included in this command
1	The command contains the first block of a multi-block component letter template
2	The command contains an intermediate block of a multi-block component letter template
3	The command contains the final block of a multi-block component letter template
4	Cancel current print job; removes a partial component letter template from Network Security Processor’s memory

`[Continuation Index]`

Field 3, this index specifies which of the four internal memory storage locations the Network Security Processor used to store the first component letter template data block. This field must be empty when the continuation flag (field 2) is set to a value of 0 or 1. This field must be empty if the command is used to send the first data block of the component letter template. For subsequent commands used to send intermediate and final data blocks, the value of this field must match the value returned in field 1 of the response to the command that was used to send the first data block of the component letter template. When the continuation flag (field 2) is set to a value of 2, 3, or 4, this field can contain the values 0, 1, 2, or 3.

`[Header]`

Field 4, the eight character header of the component to be generated by the Network Security Processor. Byte 6 of the component header must contain one of the following values: C, 1, 2, 3, 4, i, e, k, or n. This field must be empty when field 5 contains a key component. This field is ignored when the continuation flag (field 2) contains a 1, 2, or 4.

[Header,$E_{MFK.E}$(Component),MAC]

Field 5, the key component encrypted under the MFK. Byte 6 of the key component header must contain one of the following values: C, 1, 2, 3, 4, i, e, k, or n. The length of component AKB can be 58 or 74 bytes. If byte 2 of the header contains the capital letter J, the length of the AKB can be 90 bytes. The Network Security Processor will generate a random component when this field is empty, and the continuation flag (field 2) contains a value of 0 or 3. This field is ignored when the continuation flag (field 2) contains a 1, 2, or 4.

`[Component Length]`

Field 6, the length of the component to be generated by the Network Security Processor. The random values will be adjusted to odd parity for 3DES key components. This field can contain one of these values:

Value	Description
S	1key-3DES key (single-length)
D	2key-3DES key (double-length)
T	3key-3DES key (triple-length)
128	AES-128 key
192	AES-192 key
256	AES-256 key

This field must be empty when field 5 contains a component. This field is ignored when the continuation flag (field 2) contains a 1, 2, or 4.

`[Component Marker String]`

Field 7, the component marker string in the letter template file that identifies the location where the cleartext component will be printed. The component marker string is 19 characters, it represents the 16 characters of the component with spaces between each set of 4 characters. This field can contain upper and lower case letters (A-Z, a-z) and numeric digits (0-9). This field will be ignored if the continuation flag (field 2) is 1, 2 or 4.

`[Check Digit Marker String]`

Field 8, the marker character string in the letter template file that identifies the location where the check digits will be printed. This field can contain upper and lower case letters (A-Z, a-z) and numeric digits (0-9). The contents of the check digit marker string are arbitrary, but the length must be the same as the length of the check digits, except for AES check digits method H. For a 3DES key component, the length of the check digits will be 4, if option 88 is enabled it will be 6. For an AES key component, if the check digit marker string is 10 characters, method C will be used to calculate the check digits. When the check digit marker string is 19 characters, method H will be used for calculating the check digits for an AES key component. The component letter will only contain the leftmost 16 digits of the method H check digits, not the entire 64 hexadecimal characters. This field will be ignored if the continuation flag (field 2) is 1, 2 or 4.

`[Reference Marker String]`

Field 9, the marker character string in the letter template file that identifies the location where the reference value will be printed. The contents of the reference marker string are arbitrary. If present, this field must be 19 characters and can contain upper and lower case letters (A-Z, a-z) and numeric digits (0-9). The reference value that will be printed in the component letter is the 16 character MAC from the component AKB. The reference value can be included to help match the component letter with the encrypted component on the host application’s database. Printing the reference value is optional, it will be printed only if the reference marker string is present in the letter template file. This field will be ignored if the continuation flag (field 2) is 1, 2 or 4.

`Data Encoding`

Field 10, the encoding used for the component, check digits and reference marker strings in the letter template file. This field can contain one of these values:

Value	Description
A	ASCII encoding, where 1234 = 0x31323334
W	Windows encoding (16-char, little endian), where 1234 = 0x3100320033003400

`Data Type`

Field 11, only binary data is supported. This field must contain the letter B.

`Letter Template Size`

Field 12, the size of the complete component letter template file. The maximum size of the component letter template file is 1,048,576 bytes (1 megabyte).

`Block Data Length`

Field 13, the length of the data sent in this data block. The maximum value is 30000.

`Data Block`

Field 14, the binary data of the component letter template file. The maximum amount of binary data is 30000 bytes.

Table 11-14 Command 16F: Print Component Letter

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	16F
1	Letter Type	1	0, 1
2	Continuation Flag	1	0 - 4
3	[Continuation Index]	0, 1	0 - 3
4	[Header]	0, 8	printable ASCII
5	[Header,$E_{MFK.E}$(Component),MAC]	0, 58, 74, 90	printable ASCII
6	[Component Length]	0, 1, 3	S, D, T, 128, 192, 256
7	[Component Marker String]	0, 19	0 - 9, A - Z, a - z,
8	[Check Digits Marker String]	0, 4, 6, 10, 19	0 - 9, A - Z, a - z
9	[Reference Marker String]	0, 19	0 - 9, A - Z, a - z
10	Data Encoding	1	A, W
11	Data Type	1	B
12	Letter Template Size	1 - 7	0 - 1048576
13	Block Data Length	1 - 5	0 - 30000
14	Data Block	1 - 30000	binary

Responding Parameters

`26F`

Field 0, the response identifier.

`[Continuation Index]`

Field 1, this field will match field 3 of the command if the continuation flag (command-field 2) is 2, 3, or 4. It will be empty if the continuation flag is 0 or 1.

[Header,$E_{MFK.E}$(Component),MAC]

Field 2, the component generated by the Network Security Processor encrypted under the MFK. When the component is either a 3DES or an AES-192 bit key component, this field contains a 74 byte value. When the component is an AES-128 bit key component, this field contains a 58 byte value. When the component is an AES-256 bit key component, this field contains a 90 byte value. This field will be empty when a component is provided in field 5 of the command.

`Component Check Digits`

Field 3, the check digits of the key component. For a 3DES key component, the check digits are the first four digits that result from encrypting zeros using the key component. If option 88 is enabled, this field will contain six check digits. For an AES key component, the check digits will be generated by the value specified in field 7 of the command.

Table 11-15 Response 26F: Print Component Letter

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	26F
1	[Continuation Index]	0, 1	0 - 3
2	[Header,$E_{MFK.E}$(Component),MAC]	0, 58, 74, 90	printable ASCII
3	Component Check Digits	4, 6, 10, or 64	0 - 9, A - F

Usage Notes

When a letter template file requires multiple commands to process the entire letter template file, the Network Security Processor will clear the entire letter template file from its memory on any of the following error conditions. In this case, correct the error and send all of the commands required to process the entire letter template file again.

- An invalid letter template length. The total number of bytes received is not equal to the total number of bytes specified in the field 12 of the command.
- An invalid component specified in field 5. The Network Security Processor will return an error code 07.
- A TCP/IP connection or send/receive error is detected. The Network Security Processor will return an error code 11.
- A Network Security Processor execution error. The Network Security Processor will return an error code 08.
- The marker string is not present in the template file. The Network Security Processor will return an error code 12.

Command syntax errors, such as invalid number of fields in a command or invalid character in a command, will not cause the Network Security Processor to erase the letter template file from its internal memory slot. In this case, correct the syntax error and send the command again to print the component letter.

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF, check digits = B196.
See 3key-3DES Key (Triple-Length) for component values.

Generate a random Card Verification Value Key 3key-3DES component

- Component Header: 1CDNE0C0
- Component Marker String: 1234567890123456789
- Check Digit Marker String: zzzz
- Reference Value Marker String: xxxxxxxxxxxxxxxxxxx

ASCII Component Letter Template Text

```
Cleartext 3Key-3DES Key Component

Block 1: 1234567890123456789

Block 2: 1234567890123456789

Block 3: 1234567890123456789

Check Digits: zzzz

Reference Number: xxxxxxxxxxxxxxxxxxx
```

The command looks like this. For visibility purposes in this example the binary data in the data block field (field 14) is presented in hexadecimal format.

```
<16F#1#0##1CDNE0C0##T#1234567890123456789#zzzz#xxxxxxxxxxxxxxxxxxx#
A#B#193#193#436C656172746578742033 4B65792D33444553204B657920436F6D7
06F6E656E740D0A0D0A20426C6F636B20313A2031323334353637383930313233 34
35363738390D0A20426C6F636B20323A20313233343536373839303132333435363
738390D0A20426C6F636B20333A20313233343536373839303132333435363738 39
0D0A0D0A436865636B20446967697473 3A207A7A7A7A0D0A0D0A5265666572656E6
365204E756D6265723A2078787878787878787878787878787878787878780D0A#>
```

The Network Security Processor generates a random component value, the response will be similar to this:

```
<26F##1CDNE0C0,88D91664E9327D4C2F18F6BB6F95AE6AFE1D85FCE269408A,2E4
0E819F2B4DC79#4989#>
```

The cleartext generated key component is:

```
0BD6 45B9 37CE E6FB FE31 2023 1626 2AFE 2C57 FD02 EA04 5E29
check digits = 4989
```

12

Utility Commands

This section describes the commands to, test the communications link between the host and the Network Security Processor, configure the Network Security Processor, and obtain a variety of operating information about the Network Security Processor.

Quick Reference

Table 12-1 identifies the utility commands.

Table 12-1 Utility Commands

Command	**Name**	**Purpose**
Utility commands		
00	Echo	Tests the communications link between the host and the Network Security Processor.
9A#CLEAR_LOG	Clear Log	Clears the system log.
9A#CONFIG-Request	Network Security Processor Configuration Status	Returns a list of commands which are enabled and disabled.
9A#COUNT	Network Security Processor Command Count Status	Returns a list of commands being counted along with the current count value.
9A#DIAGTEST	Security Processor Crypto Test	Returns the result of the cryptographic test.
9A#HEADERS	Security Processor Allowed AKB Headers	Returns a list of AKB headers that are allowed in the SCA Calculate AKB function.
9A#ID	Network Security Processor Status ID	Returns the commands and options that are enabled in the Network Security Processor.
9A#KEY	Network Security Processor Status Key	Returns the Network Security Processor's key length and check digit information.
101	Configure Network Security Processor Options	Enables or disables specific operating parameters.

Table 12-1 Utility Commands (continued)

Command	**Name**	**Purpose**
102	Command Monitoring	Counts the number of PIN, sanity, CVV/CVC/CSC, and MAC verification failures that have been processed. It can also count the number of times an enabled command has been processed.
103	Get Average CPU Utilization	Returns a percentage value which is the average CPU utilization for the Network Security Processor.
105	License Premium Value Commands and Options	Licenses premium value commands and options.
106	Define Temporary Serial Number	Allows you to define a temporary serial number.
107	Confirm Temporary Serial Number	Activates the temporary serial number.
108	Define Security Policy	Allows you to define which commands and options will be enabled or disabled.
109	Confirm Security Policy	Activates the defined security policy.
1101	Get Image ID	Returns the software image version information.
1104	Get Temporary Serial Number Information	Returns the temporary serial number and the number of hours remaining before it expires.
1110	Get System Configuration Information	Returns the Network Security Processor system software information, and cryptographic subsystem software version information
1111	Get Date and Time	Returns the Network Security Processor system date and time in Universal Coordinated Time
1113	Get Average CPU Utilization	Returns a percentage value which is the average CPU utilization for the Network Security Processor.
1120	Get System Information	Returns the Network Security Processor serial number, product ID, system software information, and a personality version field.
1216	Get Battery Life Remaining	Returns the number of days remaining before the battery expiration messages start appearing in the log.
1221	Return IP Address of Network Security Processor	Returns the IP Address of the Network Security Processor.
1223	TCP/IP Socket Information	Returns information on the number of available sockets.
1226	Get Application Key Check Digits	Returns the check digits of the MFK and Pending MFK.

Table 12-1 Utility Commands (continued)

Command	Name	Purpose
1227	Reset to Factory State	Resets the Network Security Processor to factory state.
1228	Confirm Reset to Factory State	Completes the Reset to Factory State procedure.

Echo Test Message (Command 00)

Command 00 can be used to test the communications link between the host and the Network Security Processor. A value of 00 is returned in response to the command, or in response to an error condition in another command. See Application Error Messages for information on error codes.

Command

```
<00#Message#>
```

Response

```
<00#0000Version Level#Message#>[CRLF]
```

Calling Parameters

`00`

Field 0, the command identifier.

`Message`

Field 1, the test message to be echoed in the response. This message can be from one to 1999 bytes long and can contain any character or number except "#", ">", and "<".

Table 12-2 Command 00: Echo Test Message

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	00
1	Message	1 - 2000	Any except #, <, >

Responding Parameters

`00`

Field 0, the response identifier.

`0000Version Level`

Field 1, the software version level.

`Message`

Field 2, the message sent in the command is returned. This field is from one to 1999 bytes long and can contain any character or number, except "#", ">", and "<".

Table 12-3 Response 00: Echo Test Message

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	00
1	0000Version level	6	0 - 9
2	Message	1 - 2000	Any except #, <, >

Example

The following examples illustrate Command 00 used to echo a message. The response includes the first two digits of the Network Security Processor software version and the message.

The command looks like this:

```
<00#This is a test.#>
```

The Network Security Processor returns the following response:

```
<00#000035#This is a test.#>
```

Network Security Processor Clear Log (Command 9A)

Command 9A – This command closes the current system log on the USB flash memory device, clears the system log that is stored in memory, and then uses the current data and time to create a new system log on the USB flash memory device.

Command

```
<9A#CLEAR_LOG#>
```

Response

```
<AA#Status#>[CRLF]
```

Calling Parameters

`9A`

Field 0, the command identifier.

`CLEAR_LOG`

Field 1, the request to the security processor to clear the system log.

Table 12-4 Command 9A: Security Processor CLEAR_LOG

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier.	2	9A
1	CLEAR_LOG	9	CLEAR_LOG

Responding Parameters

`AA`

Field 0, the response identifier.

`Status`

Field 1, there are two possible status values:

DONE - confirmation that the system log has been cleared.

LOG DOES NOT EXIST - indicates an error clearing the system log.

Table 12-5 Response AA: Security Processor CLEAR_LOG

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	AA
1	Status	4 or 18	DONE, LOG DOES NOT EXIST

Usage Notes

In certain situations, such as when option 44 is enabled, the amount of command/response data can exceed the capacity of the system log. When this situation occurs, no new system log information can be recorded. Use the <9A#CLEAR_ LOG#> command to the clear the system log and create a new system log.

Example

This example illustrates sending Command 9A to clear the Network Security Processor's system log.

The command looks like this:

```
<9A#CLEAR_LOG#>
```

The response looks similar to this:

```
<AA#DONE#>
```

Network Security Processor Configuration Status (Command 9A)

Command 9A – Network Security Processor Configuration Status returns a list of enabled or disabled commands and options with a high security exposure, followed by a list of enabled commands with a low security exposure. It also returns the sequence number and serial number of the Network Security Processor. Use this command to confirm that your Network Security Processor security policy has been implemented correctly.

This command does not list four digit utility commands such as 1101, 9109, 1223, etc., they are not under the control of the security policy and are always enabled.

Command

```
<9A#CONFIG-Request#>
```

Response

```
<AA#Serial Number#Commands/Options with High Security Exposure#
Commands/Options with Low Security Exposure#Sequence Number#>[CRLF]
```

Calling Parameters

`9A`

Field 0, the command identifier.

`CONFIG-Request`

Field 1, the request to the Network Security Processor for a list of commands and options. There are three possible values:

CONFIG-ON instructs the Network Security Processor to return in field 3, a list of *enabled* commands and options that have a high security exposure. Field 4 will contain a list of *enabled* commands and options that have a low security exposure.

CONFIG-OFF instructs the Network Security Processor to return in field 3, a list of *disabled* commands and options that have a high security exposure. Field 4 will contain a list of *disabled* commands and options that have a low security exposure.

CONFIG-ALL instructs the Network Security Processor to return a list of *all* commands and options included in the Network Security Processor regardless of their enabled/disabled status. Commands and options that have a high security exposure are listed in field 3. Commands and options that have a low security exposure are listed in field 4.

note There are some customer specific commands that may appear in the list, they are not documented in this manual.

Table 12-6 Command 9A: Network Security Processor Configuration Status

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier.	2	9A
1	Request	9 or 10	CONFIG-ON, CONFIG-OFF, CONFIG-ALL

Responding Parameters

`AA`

Field 0, the response identifier.

`Serial Number`

Field 1, the serial number of the Network Security Processor.

`CONFIG Request`

Field 2, the configuration request.

If the value is CONFIG-ON, fields three and four of the response will contain the list of enabled commands and options. The enabled key types (see Byte 1, Key Usage) for import (option E0) and export (option E1) are also returned in this field.

If the value is CONFIG-OFF, fields three and four of the response will contain the list of disabled commands and options.

If the value is CONFIG-ALL, fields three and four of the response will contain the list of all commands and options contained in the Network Security Processor.

`Commands and Options with High Security Exposure`

Field 3, the list of commands and options that have a high security exposure.

`Commands and Options with Low Security Exposure`

Field 4, the list of commands and options that have a low security exposure.

`Sequence Number`

Field 5, a hexadecimal value that indicates the number of times the security policy has been updated.

Table 12-7 Response AA: Network Security Processor Configuration Status

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	AA
1	Serial Number	7	ASCII

Table 12-7 Response AA: Network Security Processor Configuration Status (continued)

Field	Contents	Length (bytes)	Legal Characters
2	CONFIG Request	9, or 10	CONFIG-ON, CONFIG-OFF, CONFIG-ALL
3	Commands and Options with a High Security Exposure	variable	ASCII
4	Commands and Options with a Low Security Exposure	variable	ASCII
5	Sequence Number	16	0-9, A - F

Usage Notes

You can send Command 9A to the Network Security Processor after the security policy has been implemented to confirm that correct commands and options are enabled or disabled.

Examples

This example illustrates sending Command 9A and receiving list of enabled commands and options. The command looks like this:

```
<9A#CONFIG-ON#>
```

The response looks similar to this.

```
<AA#JL06RP#CONFIG-ON#(62),(63),(A0)="4",(A1)="S",(A2)="S",
(E0)="",(E1)=""#00,10,31,32,3A,5C,5E,70,71,72,73,74,7E,7F,93,99,
9A,9B,9C,9E,9F,101,103,105,106,107,108,109,113,304,335,346,348,
350,352,354,356,357,359,35A,35F,365,36A#0000000000000001#>
```

This example illustrates sending Command 9A and receiving list of disabled commands and options. The command looks like this:

```
<9A#CONFIG-OFF#>
```

The response looks similar to this.

```
<AA#JL06RP#CONFIG-OFF#11,13,14,15,1A,1B,1C,1E,1F,30,33,34,35,
36,37,38,39,3D,3F,55,58,59,5D,5F,75,76,77,78,79,7A,7B,90,94,
95,96,97,98,B1,B2,B3,B4,B5,B6,B7,BA,BB,BC,BD,BE,BF,D0,D1,D2,
D3,D4,D5,D6,D7,D8,D9,DA,DB,102,110,111,112,114,115,117,118,
119,11A,11B,11C,11E,11F,120,121,122,123,124,125,126,127,12A,
12B,12C,12D,12F,131,132,133,134,135,136,138,139,15E,161,162,
163,16A,16E,16F,301,302,305,306,307,308,309,30A,30B,30C,30D,
30E,30F,319,31A,31B,31C,31D,31E,31F,321,322,323,324,325,326,
328,329,32A,32B,32C,32D,32E,331,332,333,334,336,337,338,339,
33A,33B,33C,33D,33E,33F,340,341,342,343,344,345,347,349,34A,
34C,34D,34E,34F,351,353,355,358,35B,35C,35D,360,361,362,363,
364,36B,36C,370,371,372,373,374,375,376,377,37A,37B,381,382,
383,384,385,386,387,388,389,38A,38B,38C,38D,390,391,392,39A,
39B,39C,3A1,3A2,3A3,3A4,3B2,3B3,3B4,3B5,3B6,3B7,3B8,3EA,3F0,
3F1,3FA,3FB,3FC,3FD,(42),(43),(46),(47),(48),(49),(4A),(4B),
(4C),(4D),(4E),(4F),(60),(61),(64),(66),(68),(69),(6A),(6B),
(6C),(6E),(6F),(80),(81),(82),(83),(84),(87),(88),(89),(8A),
(8B),(8C),(8E),(8F),(C1),(E2),(E3),(E4),(E5)#(20),(21),(23),
(27),(28),(44)#0000000000000001#>
```

This example illustrates sending Command 9A and receiving list of all commands and options.

```
<9A#CONFIG-ALL#>
```

The response looks similar to this.

```
<AA#JL06RP#CONFIG-ALL#11,13,14,15,1A,1B,1C,1E,1F,30,33,34,35,36,
37,38,39,3D,3F,55,58,59,5D,5F,75,76,77,78,79,7A,7B,90,94,95,96,
97,98,B1,B2,B3,B4,B5,B6,B7,BA,BB,BC,BD,BE,BF,D0,D1,D2,D3,D4,D5,
D6,D7,D8,D9,DA,DB,102,110,111,112,114,115,117,118,119,11A,11B,
11C,11E,11F,120,121,122,123,124,125,126,127,12A,12B,12C,12D,12F,
131,132,133,134,135,136,138,139,15E,161,162,163,16A,16E,16F,301,
302,305,306,307,308,309,30A,30B,30C,30D,30E,30F,319,31A,31B,31C,
31D,31E,31F,321,322,323,324,325,326,328,329,32A,32B,32C,32D,32E,
331,332,333,334,336,337,338,339,33A,33B,33C,33D,33E,33F,340,341,
342,343,344,345,347,349,34A,34C,34D,34E,34F,351,353,355,358,35B,
35C,35D,360,361,362,363,364,36B,36C,370,371,372,373,374,375,376,
377,37A,37B,381,382,383,384,385,386,387,388,389,38A,38B,38C,38D,
390,391,392,39A,39B,39C,3A1,3A2,3A3,3A4,3B2,3B3,3B4,3B5,3B6,3B7,
3B8,3EA,3F0,3F1,3FA,3FB,3FC,3FD,(42),(43),(46),(47),(48),(49),
(4A),(4B),(4C),(4D),(4E),(4F),(60),(61),(62),(63),(64),(66),(68),
(69),(6A),(6B),(6C),(6E),(6F),(80),(81),(82),(83),(84),(87),(88),
(89),(8A),(8B),(8C),(8E),(8F),(A0)="4",(A1)="S",(A2)="S",(C1),
(E0)="",(E1)="",(E2),(E3),(E4),(E5)#00,10,31,32,3A,5C,5E,70,71,
72,73,74,7E,7F,93,99,9A,9B,9C,9E,9F,101,103,105,106,107,108,109,
113,304,335,346,348,350,352,354,356,357,359,35A,35F,365,36A,(20),
(21),(23),(27),(28),(44)#0000000000000001#>
```

Network Security Processor Count Status (Command 9A)

Command 9A – Network Security Processor Count Status returns a list of commands that are being counted along with the current count value (in decimal). Each time the Network Security Processor successfully processes a command that is being counted the counter value is decremented by 1. Commands that are not successfully processed by the Network Security Processor, such as commands that contain syntax errors that result in an error response returned from the Network Security Processor, are not counted.

caution Once the counter value reaches zero, the Network Security Processor will return an error <00#0300xx#> instead of processing the command.

A maximum of nine cryptographic commands can be counted. Utility commands and options cannot be counted. See section 4 of the *Secure Configuration Assistant-3 User Guide* for instructions on how to configure command counting, or see command 108.

This command can be used to determine the number of commands processed by the Network Security Processor.

Command

```
<9A#COUNT#>
```

Response

```
<AA#Serial Number#Reserved#[Command-Counter#][Command-Counter#]
[Command-Counter#][Command-Counter#][Command-Counter#]
[Command-Counter#][Command-Counter#][Command-Counter#]
[Command-Counter#]>[CRLF]
```

Calling Parameters

`9A`

Field 0, the command identifier.

`COUNT`

Field 1, the request to the security processor for a list of commands that have been enabled for counting.

Table 12-8 Command 9A: Security Processor Count Status

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier.	2	9A
1	Function	5	COUNT

Responding Parameters

`AA`

Field 0, the response identifier.

`Serial Number`

Field 1, the serial number of the Network Security Processor.

`Reserved`

Field 2, this field is reserved for future use.

`[Command-Count#]`

Field 3, the command being counted followed by the current counter value. This field is present only if there is at least one command being counted.

`[Command-Count#]`

Field 4, the command being counted followed by the current counter value. This field is present only if there is at least two command being counted.

`[Command-Count#]`

Field 5, the command being counted followed by the current counter value. This field is present only if there is at least three command being counted.

`[Command-Count#]`

Field 6, the command being counted followed by the current counter value. This field is present only if there is at least four command being counted.

`[Command-Count#]`

Field 7, the command being counted followed by the current counter value. This field is present only if there is at least five command being counted.

`[Command-Count#]`

Field 8, the command being counted followed by the current counter value. This field is present only if there is at least six command being counted.

`[Command-Count#]`

Field 9, the command being counted followed by the current counter value. This field is present only if there is at least seven command being counted.

`[Command-Count#]`

Field 10, the command being counted followed by the current counter value. This field is present only if there is at least eight command being counted.

`[Command-Count#]`

Field 11, the command being counted followed by the current counter value. This field is present only if there are nine command being counted.

Table 12-9 Response AA: Security Processor Count Status

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	AA
1	Serial Number	7	ASCII
2	Reserved	1	0-9
3	[Command-Count#]	15	0-9, A-F
4	[Command-Count#]	15	0-9, A-F
5	[Command-Count#]	15	0-9, A-F
6	[Command-Count#]	15	0-9, A-F
7	[Command-Count#]	15	0-9, A-F
8	[Command-Count#]	15	0-9, A-F
9	[Command-Count#]	15	0-9, A-F
10	[Command-Count#]	15	0-9, A-F
11	[Command-Count#]	15	0-9, A-F

Usage Notes

You can send Command 9A to the security processor to determine the number of commands processed.

Example

The command looks like this:

```
<9A#COUNT#>
```

The response looks similar to this. Field 3 of the response shows that command 10 is counted and the current counter value is 50. Field 4 of the response shows that command 11 is being counted and the current counter value is 40.

```
<AA#D126XL#1#0010-0000000050#0011-0000000040#>
```

Network Security Processor Crypto Test (Command 9A)

Command 9A – This command performs a cryptographic test on the Network Security Processor.

Command

```
<9A#DIAGTEST#Algorithm#Reserved#>
```

Response

```
<AA#Result#>[CRLF]
```

Calling Parameters

`9A`

Field 0, the command identifier.

`DIAGTEST`

Field 1, the request to the Network Security Processor to perform the cryptographic test.

`Algorithm`

Field 2, the algorithm test to be performed. Specify 0 (zero) or 1 to perform the 3DES test.

`Reserved`

Field 3, this field must be empty.

Table 12-10 Command 9A: Security Processor Crypto Test

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	9A
1	DIAGTEST	8	DIAGTEST
2	Algorithm	1	0,1
3	Reserved	0	

Responding Parameters

`AA`

Field 0, the response identifier.

`Results`

Field 1, the result of the tests. The result of a successful test is “OK”. The possible results for a failed test are: 3DES_KEY_ERROR", "3DES_ENCRYPT_FAIL", "3DES_ENCRYPT_MISMATCH", "3DES_DECRYPT_FAIL", and "3DES_DECRYPT_MISMATCH".

Table 12-11 Response AA: Security Processor Crypto Test

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	AA
1	Result	2, varies	OK, A-Z

Usage Notes

Only 3DES tests are performed.

Example

The command looks like this:

```
<9A#DIAGTEST#0##>
```

When the test completes successfully, the Network Security Processor returns this response.

```
<AA#OK#>
```

Network Security Processor AKB header list (Command 9A)

Command 9A – This command returns a list of the valid AKB headers which can be used in the SCA Calculate AKB function.

Command

```
<9A#HEADERS#>
```

Response

```
<AA#List of Allowed Headers#>[CRLF]
```

Calling Parameters

`9A`

Field 0, the command identifier.

`HEADERS`

Field 1, the action to be performed.

Table 12-12 Command 9A: Network Security Processor AKB header list

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	9A
1	HEADERS	7	HEADERS

Responding Parameters

`AA`

Field 0, the response identifier.

`List of Allowed headers`

Field 1, the comma separated list of AKB headers that are supported by the Network Security Processor.

Table 12-13 Response AA: Network Security Processor AKB header list

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	AA
1	Result	varies	0-9 and A-Z

Example

The command looks like this:

```
<9A#HEADERS#>
```

The response will be similar to this.

```
<AA#HEADERS#1KDNE000,1KDEE000,1KDDE000,1kDNE000,1kDNN000,1KDNE00n,
1KDNE00T,1PUNE000,1PUEE000,1PUDE000,1PDNE000,1PDEE000,1PDDE000,1cD
NE000,1cDEE000,1cDDE000,1K3NE000,1A3NE000,1A7NE000,1A3NE0C0,1A3NE0
20,1A7NE0C0,1A7NE020,1PUNE00n,1V4NE000,1V4VE000,1PDNE00I,1PDEE00I,
1PDDE00I,1iDNE000,1DDNE000,1DDEE000,1DDDE000,1mFNE00m,1DsNE000,1DD
NE00n,1DJNE000,1DJEE000,1DJDE000,1JJNE000,1DDNE00I,1DDEE00I,1DDDE0
0I,1MHNE000,1MHNN000,1MHGE000,1MHGN000,1MHVE000,1MHVN000,1M2NE000,
1M2NN000,1M2GE000,1M2GN000,1M2VE000,1M2VN000,1MDNE000,1MDGE000,1MD
VE000,1c7NE000,1c7EE000,1c7DE000,1cDNE00B,1mXNE000,1mXGE000,1mXVE0
00,1CDNE000,1CDGE000,1CDVE000,1MDNE00n,1dDNE00n,1cDNN00b,1CsNE000,
1CsGE000,1CsVE000,1MDNE00T,1MDNE001,1MDNE002,1MDNE003,1MDNE004,1MD
NE005,1VUNE000,1V0NE000,1VINE000,1V3NE000,1VVNE000,1VBNE000,1VNNE0
00,1VaNE000,1mFNE00p,1VUGE000,1VUVE000,1V0GE000,1V0VE000,1VIGE000,
1VIVE000,1V3GE000,1V3VE000,1VVGE000,1VVVE000,1VBGE000,1VBVE000,1VN
GE000,1VNVE000,1VaGE000,1VaVE000,1VrNN000,1mDEN00r,1mDEN00b,1VPNE0
00,1VPGE000,1VPVE000,1VZNE000,1ADNE000,1ndNE000,1nuNE000,1ADNE0C0,
1ADNE020,1IDNE000,1nCNE000,1nUNE000,1IsNE000,1IJNE000,1MDNE00C,1MD
GE00C,1MDVE00C,1dDNE000,1mZNE000,1mFNE00M,1dsNE000,1mVNE000,1mENE0
00,1miNE000,1mENE00a,1mEDE00a,1mEEE00a,1mENE00d,1mEDE00d,1mEEE00d,
1mENE00e,1mEDE00e,1mEEE00e,1mENE00f,1mEDE00f,1mEEE00f,1mENE00g,1mE
DE00g,1mEEE00g,1mONE000,1IDNE00M,1mFVE00M,1mFGE00M,1mENE00M,1mEDE0
0M,1mEEE00M,1mENE00E,1mEDE00E,1mEEE00E,1mFGE000,1mFNE00P,1mFDE00P,
1IDNE00P,1TDNE001,1TDNE002,1TDNE003,1TDNE004#>
```

Network Security Processor Status ID (Command 9A)

Command 9A – Network Security Processor Status ID returns the Network Security Processor's current configuration and serial number. Use this command to ensure that only authorized commands and options are enabled. This command does not return status of utility commands, nor does it highlight high security exposure commands or options, see Network Security Processor Configuration Status (Command 9A), for more information on obtaining commands listed by security exposure.

Command

```
<9A#ID#>
```

Response

```
<AA#Serial No.#Type[,Currently Enabled Options],
import="[allowed import AKB header types]",
export="[allowed export AKB header types]"#
Minimum PIN Length,PIN Length Character,
DUKPT Session Key Length#[Enabled 0X Commands]#
[Enabled 1X Commands]#[Enabled 3X Commands]#
[Enabled 5X Commands]#[Enabled 7X Commands]#
[Enabled 9X Commands]#[Enabled BX Commands]#
[Enabled DX Commands]#[Enabled 10X Commands]#
[Enabled 11X Commands]#[Enabled 12X Commands]#
[Enabled 13X Commands]#[Enabled 15X Commands]#
[Enabled 16X Commands]#[Enabled 30X Commands]#
[Enabled 31X Commands]#[Enabled 32X Commands]#
[Enabled 33X Commands]#[Enabled 34X Commands]#
[Enabled 35X Commands]#[Enabled 36X Commands]#
[Enabled 37X Commands]#[Enabled 38X Commands]#
[Enabled 39X Commands]#[Enabled 3AX Commands]#
[Enabled 3BX Commands]#[Enabled 3EX Commands]#
[Enabled 3FX Commands]#>[CRLF]
```

Calling Parameters

`9A`

Field 0, the command identifier.

`ID`

Field 1, the request to the Network Security Processor for current configuration information.

Table 12-14 Command 9A: Network Security Processor Status ID

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	9A
1	Configuration-information request identifier	2	ID

Responding Parameters

`AA`

Field 0, the response identifier.

`Serial No.`

Field 1, the factory-assigned serial number.

This field is six bytes long and contains ASCII characters.

`Type[,Currently Enabled Options],import="[allowed import AKB header types]",export="[allowed export AKB header types]"`

Field 2, Product identification, followed by all currently enabled options. This field's length depends on the Network Security Processor's configuration. This field can contain the numbers 0 to 9, the characters A to Z, a to z, and ",". The enabled key types (see Byte 1, Key Usage) for import (option E0) and export (option E1) are also returned in this field.

`Minimum PIN Length,PIN Length Character,DUKPT Session Key Length`

Field 3, this field contains 3 values. The minimum valid PIN length which is defined in the Network Security Processor's security policy using option A0. The PIN Sanity Indicator which is defined in the Network Security Processor's security policy using option A1. The 3DES DUKPT Session Key length which is defined in the Network Security Processor's security policy using option A2.

`[Enabled 0X Commands]`

Field 4, a listing of the 0X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 0X commands have been configured for use, this field is empty.

`[Enabled 1X Commands]`

Field 5, a listing of the 1X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 1X commands have been configured for use, this field is empty.

`[Enabled 3X Commands]`

Field 6, a listing of the 3X commands that have been configured for use. This field’s length depends on the Network Security Processor’s configuration. It can contain the characters 0 to 9, A to Z, and “,”. If none of the 3X commands have been configured for use, this field is empty.

`[Enabled 5X Commands]`

Field 7, a listing of the 5X commands that have been configured for use. This field’s length depends on the Network Security Processor’s configuration. It can contain the characters 0 to 9, A to Z, and “,”. If none of the 5X commands have been configured for use, this field is empty.

`[Enabled 7X Commands]`

Field 8, a listing of the 7X commands that have been configured for use. This field’s length depends on the Network Security Processor’s configuration. It can contain the characters 0 to 9, A to Z, and “,”. If none of the 7X commands have been configured for use, this field is empty.

`[Enabled 9X Commands]`

Field 9, a listing of the 9X commands that have been configured for use. This field's length depends on the Network Security Processor’s configuration. It can contain the characters 0 to 9, A to Z, and “,”. If none of the 9X commands have been configured for use, this field is empty.

`[Enabled BX Commands]`

Field 10, a listing of the BX commands that have been configured for use. This field’s length depends on the Network Security Processor’s configuration. It can contain the characters 0 to 9, A to Z, and “,”. If none of the BX commands have been configured for use, this field is empty.

`[Enabled DX Commands]`

Field 11, a listing of the DX commands that have been configured for use. This field’s length depends on the Network Security Processor’s configuration. It can contain the characters 0 to 9, A to Z, and “,”. If none of the DX commands have been configured for use, this field is empty.

`[Enabled 10X Commands]`

Field 12, a listing of the 10X commands that have been configured for use. This field’s length depends on the Network Security Processor’s configuration. It can contain the characters 0 to 9, A to Z, and “,”. If none of the 10X commands have been configured for use, this field is empty.

`[Enabled 11X Commands]`

Field 13, a listing of the 11X commands that have been configured for use. This field’s length depends on the Network Security Processor’s configuration. It can contain the characters 0 to

9, A to Z, and ",". If none of the 11X commands have been configured for use, this field is empty.

```
[Enabled 12X Commands]
```

Field 14, a listing of the 12X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 12X commands have been configured for use, this field is empty.

```
[Enabled 13X Commands]
```

Field 15, a listing of the 13X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 13X commands have been configured for use, this field is empty.

```
[Enabled 15X Commands]
```

Field 16, a listing of the 15X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 15X commands have been configured for use, this field is empty.

```
[Enabled 16X Commands]
```

Field 17, a listing of the 16X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 16X commands have been configured for use, this field is empty.

```
[Enabled 30X Commands]
```

Field 18, a listing of the 30X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 30X commands have been configured for use, this field is empty.

```
[Enabled 31X Commands]
```

Field 19, a listing of the 31X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 31X commands have been configured for use, this field is empty.

```
[Enabled 32X Commands]
```

Field 20, a listing of the 32X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 32X commands have been configured for use, this field is empty.

```
[Enabled 33X Commands]
```

Field 21, a listing of the 33X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 33X commands have been configured for use, this field is empty.

`[Enabled 34X Commands]`

Field 22, a listing of the 34X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 34X commands have been configured for use, this field is empty.

`[Enabled 35X Commands]`

Field 23, a listing of the 35X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 35X commands have been configured for use, this field is empty.

`[Enabled 36X Commands]`

Field 24, a listing of the 36X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 36X commands have been configured for use, this field is empty.

`[Enabled 37X Commands]`

Field 25, a listing of the 37X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 37X commands have been configured for use, this field is empty.

`[Enabled 38X Commands]`

Field 26, a listing of the 38X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 38X commands have been configured for use, this field is empty.

`[Enabled 39X Commands]`

Field 27, a listing of the 39X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 39X commands have been configured for use, this field is empty.

`[Enabled 3AX Commands]`

Field 28, a listing of the 3AX commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 3AX commands have been configured for use, this field is empty.

`[Enabled 3BX Commands]`

Field 29, a listing of the 3BX commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 3BX commands have been configured for use, this field is empty.

`[Enabled 3EX Commands]`

Field 30, a listing of the 3EX commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 3EX commands have been configured for use, this field is empty.

`[Enabled 3FX Commands]`

Field 31, a listing of the 35X commands that have been configured for use. This field's length depends on the Network Security Processor's configuration. It can contain the characters 0 to 9, A to Z, and ",". If none of the 3FX commands have been configured for use, this field is empty.

Table 12-15 Response AA: Network Security Processor Status ID

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Response identifier	2	AA
1	Serial number	6	ASCII characters
2	Currently enabled options	*n**	0 - 9, A - Z, a - z, ","
3	Minimum PIN length, PIN Length character, DUKPT session key length	5	0 - 9, ","
4	[Enabled 0X commands]	0 - *n**	0 - 9, A - F, ","
5	[Enabled 1X commands]	0 - *n**	0 - 9, A - F, ","
6	[Enabled 3X commands]	0 - *n**	0 - 9, A - F, ","
7	[Enabled 5X command]	0 - *n**	0 - 9, A - F, ","
8	[Enabled 7X commands]	0 - *n**	0 - 9, A - F, ","
9	[Enabled 9X commands]	0 - *n**	0 - 9, A - F, ","
10	[Enabled BX commands]	0 - *n**	0 - 9, A - F, ","
11	[Enabled DX commands]	0 - *n**	0 - 9, A - F, ","
12	[Enabled 10X commands]	0 - *n**	0 - 9, A - F, ","
13	[Enabled 11X commands]	0 - *n**	0 - 9, A - F, ","
14	[Enabled 12X commands]	0 - *n**	0 - 9, A - F, ","

Table 12-15 Response AA: Network Security Processor Status ID (continued)

Field	Contents	Length (bytes)	Legal Characters
15	[Enabled 13X commands]	0 - *n**	0 - 9, A - F, “,”
16	[Enabled 15X commands]	0 - *n**	0 - 9, A - F, “,”
17	[Enabled 16X commands]	0 - *n**	0 - 9, A - F, “,”
18	[Enabled 30X commands]	0 - *n**	0 - 9, A - F, “,”
19	[Enabled 31X commands]	0 - *n**	0 - 9, A - F, “,”
20	[Enabled 32X commands]	0 - *n**	0 - 9, A - F, “,”
21	[Enabled 33X commands]	0 - *n**	0 - 9, A - F, “,”
22	[Enabled 34X commands]	0 - *n**	0 - 9, A - F, “,”
23	[Enabled 35X commands]	0 - *n**	0 - 9, A - F, “,”
24	[Enabled 36X commands]	0 - *n**	0 - 9, A - F, “,”
25	[Enabled 37X commands]	0 - *n**	0 - 9, A - F, “,”
26	[Enabled 38X commands]	0 - *n**	0 - 9, A - F, “,”
27	[Enabled 39X commands]	0 - *n**	0 - 9, A - F, “,”
28	[Enabled 3AX commands]	0 - *n**	0 - 9, A - F, “,”
29	[Enabled 3BX commands]	0 - *n**	0 - 9, A - F, “,”
30	[Enabled 3EX commands]	0 - *n**	0 - 9, A - F, “,”
31	[Enabled 3FX commands]	0 - *n**	0 - 9, A - F, “,”

* Length varies.

Usage Notes

There are some undocumented commands that may appear in the list. These are custom commands which are not generally available.

Example

Obtain a list of enabled commands and options.

The command looks like this:

```
<9A#ID#>
```

The Network Security Processor issues a response that contains the following information:

- Serial number: JL08R2

- Device Type: A10160
- Enabled Options: 62, 63
- Allowed Import Key Types: none
- Allowed Export Key Types: none
- Minimum PIN length: 4
- Sanity indicator: S
- 3DES DUKPT session key length: Single
- Enabled 0x commands: 00
- Enabled 1X commands: 10
- Enabled 3X commands: 31, 32, 3A
- Enabled 5X commands: 5C, 5E
- Enabled 7X commands: 70, 71, 72, 73, 74, 7E, 7F
- Enabled 9X commands: 93, 99, 9A, 9B, 9C, 9E, 9F
- Enabled 10x commands: 101, 103, 105, 106, 107, 108, 109
- Enabled 11X commands: 113
- Enabled 33X commands: 304
- Enabled 33X commands: 335
- Enabled 34x commands: 346, 348
- Enabled 35X commands: 350, 352, 354, 356, 357, 359, 35A, 35F
- Enabled 36X commands: 365, 36A

The Network Security Processor returns a response similar to this:

```
<AA#JL08R2#A10160,62,63,Import="",Export=""#4,S,S#00#10#31,32,3A#
5C,5E#70,71,72,73,74,7E,7F#93,99,9A,9B,9C,9E,9F###101,103,105,106
,107,108,109#113#####304###335#346,348#350,352,354,356,357,359,35
A,35F#365,36A########>
```

Network Security Processor Status Key (Command 9A)

Command 9A – Network Security Processor Status Key command returns the number of available key locations in the volatile table, as well as the check digits of the Master File Keys stored in the Network Security Processor's non-volatile key table.

Command

```
<9A#KEY#>
```

Response

```
<AA#Remaining Locations#[MFK Name]#[MFK Check Digits]#
[MFK Length]#[Pending MFK Name]#[Pending MFK Check Digits]#
[Pending MFK Length]#[Retired MFK Name]#[Retired MFK Check Digits]#
[Retired MFK Length]####>[CRLF]
```

Calling Parameters

`9A`

Field 0, the command identifier.

`KEY`

Field 1, the request to the Network Security Processor for current key information.

Table 12-16 Command 9A: Network Security Processor Status Key

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	2	9A
1	Key-information request identifier	3	KEY

Responding Parameters

`AA`

Field 0, the response identifier.

`Remaining Locations`

Field 1, the number of available locations in the volatile table. This field contains a 4 byte decimal value. The value returned does **not** include the key locations available for the Diebold Number Tables.

`[MFK Name]`

Field 2, the Master File Key's name, MFK1. This field is empty if a Master File Key does not exist.

`[MFK Check Digits]`

Field 3, the Master File Key's check digits. This field contains a 4 byte hexadecimal value. This field is empty if a Master File Key does not exist.

`[MFK Length]`

Field 4, the Master File Key's length. This field returns either a D or T to indicate that the Master File Key is either a 2key-3DES (double length) or a 3key-3DES (triple length) key. This field is empty if a Master File Key does not exist.

`[Pending MFK Name]`

Field 5, the pending Master File Key's name, PMFK1. This field is empty if a pending Master File Key does not exist.

`[Pending MFK Check Digits]`

Field 6, the pending Master File Key's check digits. This field is a 4 byte hexadecimal value. This field is empty if a pending Master File Key does not exist.

`[Pending MFK Length]`

Field 7, the Pending Master File Key's length. This field returns either a D or T to indicate that the Pending Master File Key is either a 2key-3DES (double length) or a 3key-3DES (triple length) key. This field is empty if a pending Master File Key does not exist.

`[Retired MFK Name]`

Field 8, the retired Master File Key's name. This field will contain the name of the retired Master File Key. This field is empty if a retired Master File Key does not exist.

`[Retired MFK Check Digits]`

Field 9, the retired Master File Key's check digits. This field will contain the check digits of the retired Master File Key. This field is empty if a retired Master File Key does not exist.

`[Retired MFK Length]`

Field 10, the retired Master File Key's length. This field returns either a D or T to indicate that the retired Master File Key is either a 2key-3DES (double length) or a 3key-3DES (triple length) key. This field is empty if a retired Master File Key does not exist.

Fields 11 through 13 will be empty.

Table 12-17 Response AA: Network Security Processor Status Key

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	2	AA
1	Remaining locations	4	0 - 9
2	[MFK name]	0, 4	0 - 9, A - Z

Table 12-17 Response AA: Network Security Processor Status Key (continued)

Field	Contents	Length (bytes)	Legal Characters
3	[MFK Check Digits]	0, 4	0 - 9, A - F
4	[MFK length]	0, 1	D or T
5	[Pending MFK name]	0, 5	PMFK1
6	[Pending MFK Check Digits]	0, 4	0 - 9, A - F
7	[Pending MFK length]	0, 1	D or T
8	[Retired MFK name]	0, 4,5	0 - 9, A - Z
9	[Retired MFK Check Digits]	0, 4	0 - 9, A - F
10	[Retired MFK length]	0, 1	D or T
11-13	Reserved	0	

Example

The 3key-3DES Master File Key is:
2ABC 3DEF 4567 0189 9810 7645 FED3 CBA2 0123 4567 89AB CDEF,
check digits = B196. See 3key-3DES Key (Triple-Length) for component values.

Obtaining Key status.

The command looks like this:

```
<9A#KEY#>
```

The Network Security Processor returns a response that contains the following information:

- Remaining locations in volatile table: 1000
- Master file key’s name: MFK1
- Master file key’s check digits: B196
- Master file key’s length: Triple length (T)
- Pending master file key’s name: PMFK1
- Pending master file key’s check digits: 2590
- Pending master file key’s length: Triple length (T)

The Network Security Processor returns the following response:

```
<AA#1000#MFK1#B196#T#PMFK1#2590#T#######>
```

Configure Network Security Processor Option (Command 101)

Command 101 enables and disables various operating parameters. The values defined for these options are stored in non-volatile memory. Power cycling the Network Security Processor does not change the value of an option.

In version 1.50, the ability to control the response returned to the 1101 and 1110 commands has been added.

Command

```
<101#[Option Text]#>
```

Response

```
<201#Y#>[CRLF]
```

Calling Parameters

`101`

Field 0, the command identifier.

`Option Text`

Field 1, the option text. Option text is made up of option words. Each option word consists of a three-digit option ID and a one-digit action flag. The length of this field must be zero or a multiple of four. When this field is empty, all options will be set to their default values.

Option ID	Description
020	Append the Master File Key name to all responses except the response of the status command, 9A. The default action is do not append the Master File Key name.
021	Append the detailed error information to error responses. The default action is do not append a detailed error.
023	Remove the carriage return and line feed from all responses. The default action is carriage return and line feed are appended to all responses.
027	Use the right-most 4 PIN digits for Diebold PIN verification. The default action is to use the leftmost 4 PIN digits.
028	Return “HP Atalla” in the responses to the 1101 and 1110 commands. The default action, when this option is disabled, is to return “HPE Atalla” in the responses to the 1101 and 1110 commands.

Option ID	Description
044	Log the command in error and response to the system log. The default action is do not log. **When enabled, this option can have a significant negative impact on the performance of the Network Security Processor. This option should only be enabled to capture an invalid command that generates an Network Security Processor error response. Once the invalid command has been captured it is highly recommended that this option be disabled.** When this option is enabled, warning messages may appear in the system log for commands sent by the SCA to the Network Security Processor, such as <Remote#Info#> and <9A#HEADERS#>. This is not an error condition it is normal behavior when this option is enabled.

Action Flag	Meaning
D	Disable option
E	Enable option

Table 12-18 Command 101: Configure Network Security Processor Option

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	101
1	Option text	multiple of 4	0 - 9, E, D

Responding Parameters

```
201
```

Field 0, the command identifier.

```
Y
```

Field 1, an indicator that the Network Security Processor has been configured with the options specified in the command.

Table 12-19 Response 201: Configure Network Security Processor Option

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	201
1	Configuration confirmation	1	Y

Examples

Set multiple configuration options

- Append the Master File Key name to the response; indicated by the option text 020E.
- Append detailed error information to response 00; indicated by the option text 021E.
- Use the right-most 4 PIN digits for Diebold PIN verification; indicated by option text 027E

The command looks like this:

```
<101#020E021E027E#>
```

The Network Security Processor returns the following response:

```
<201#Y#>
```

Enable error logging

The command looks like this:

```
<101#044E#>
```

The Network Security Processor returns the following response:

```
<201#Y#>
```

Disable error logging

The command looks like this:

```
<101#044D#>
```

The Network Security Processor returns the following response:

```
<201#Y#>
```

Disable the carriage return and line feed (CRLF)

The command looks like this:

```
<101#023E#>
```

The Network Security Processor returns the following response:

```
<201#Y#>
```

Reset options to their default values

The command looks like this:

```
<101##>
```

The Network Security Processor returns the following response:

```
<201#Y#>
```

Command Monitoring (Command 102)

Command 102 allows you to obtain the number of PIN, sanity, CVV/CVC/CSC, and MAC verification failures that have been processed by the Network Security Processor. It can also be used to count the number of times an enabled command has been processed by the Network Security Processor.

This command is not enabled in the Network Security Processor's default security policy. To use this command, you must enable it in the Network Security Processor's security policy.

note This command is only allowed on the USB and Serial interfaces, and also on the Management ports.

Command

```
<102#Action#Mode#[Command]#>
```

Response

```
<202#Action#Mode#Start Time#End Time#[Count]#>
```

Calling Parameters

`102`

Field 0, the command identifier.

`Action`

Field 1, the action to be performed. The allowed values are:

Value	Description
START	Start monitoring.
RETRIEVE	Return the count value, reset the count value to zero and continue monitoring.
STOP	Stop monitoring.

`Mode`

Field 2, the mode of operation. The allowed values are:

Value	Description
0	Count the number of PIN verification failures when the Network Security Processor processes any of these commands: D0, 32, 36, 37, 38, 3A, 3F, 322, 323, 328, 329, 32A and 387.
1	Count the number of PIN sanity failures when the Network Security Processor processes any of these commands: 31, 32, 33, 35, 36, 37, 38, 39, 3A, 3D, 3F, 90, BA, BB, BD, 161, 163, 163, 322, 323, 328, 329, 32A, 331, 335, 346, 347, 362, 363, 364, 370, 371, 372, 387, 3A2 and 3A3.
2	Count the number of CVV/CVC/CSC verification failures when the Network Security Processor processes any of these commands: 3A, 5E, 357, 359, 35A, 35F and 36A.
3	Count the number of MAC verification failures when the Network Security Processor processes any of these commands: 58, 5C, 5F, 99, 9C, BA, BB, DA, 301, 30B, 30D, 30E, 346, 348, 355 and 381.
4	Count the number of times the commands, specified in field 3, have been successfully processed.

`[Command]`

Field 3, the list of enabled commands to be counted. This field must contain a command ID, or a comma separated list of command IDs when field 1 (Action) contains the word START and field 2 (Mode) contains the value 4, otherwise it must be empty. A maximum of 16 commands can be counted.

Table 12-20 Command 102: Command Monitoring

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	102
1	Action	4, 5, 8	START, RETRIEVE, STOP
2	Mode	1	0 - 4
3	[Command]	0 - 80	0 - 9, A - F, “,”

Responding Parameters

`202`

Field 0, the response identifier.

`Action`

Field 1, the action value supplied in field 1 of the command.

`Mode`

Field 2, the mode value supplied in field 2 of the command.

`Start Time`

Field 3, the date/time when the monitoring task was started.

The format is: `YYYYMMDD HH:MM:SS`
A start time value of 20121221 19:03:12, is December 21, 2012 7:03:12 PM.

`End Time`

Field 4, the date/time when the monitoring task was stopped or data was retrieved. This field will be empty when the action in field 1 contains the value START.

The format is: `YYYYMMDD HH:MM:SS`
An end time value of 20121221 20:03:12, is December 21, 2012 8:03:12 PM.

`[Count]`

Field 5, the count value. This field will be empty when the action specified in field 1 contains the value START.

The count value and format of this field depends on the mode specified in field 2. When field 2 contains a mode value in the range of 0 - 3, this field will contain a count value indicating the number of times the mode being counted has occurred. When field 2 contains a mode value of 4, this field will contain a count value of the command being counted, the format is `CMDID=COUNT`. Multiple commands are separated by a comma.

Table 12-21 Response 202: Command Monitoring

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	202
1	Action	4, 5, 8	START, RETRIEVE, STOP
2	Mode	1	0 - 4
3	Start Time	17	0 - 9, “:”
4	End Time	17	0 - 9, “:”
5	[Count]	0, varies	0 - 9, A - F, “=”, “,”

Usage Notes

- Multiple instances of this command can run concurrently however only one instance of each mode is allowed.
- The RETRIEVE action resets the count value to zero.
- The maximum count value is 4,294,967,295. The count value will be reset to zero if the maximum count value is exceeded.

Examples

Start counting the number of PIN Verification failures

The command looks like this:

```
<102#START#0##>
```

The Network Security Processor issues a response similar to this:

```
<202#START#0#20121221 19:57:58###>
```

Get the number of times a PIN has failed to verify

```
<102#RETRIEVE#0##>
```

The Network Security Processor issues a response similar to this:

```
<202#RETRIEVE#0#20121221 19:57:58#20121221 20:20:41#3#>
```

Stop counting the number of PIN verification failures

The command looks like this:

```
<102#STOP#0##>
```

The Network Security Processor issues a response similar to this:

```
<202#STOP#0#20121221 20:20:41#20121221 20:20:51#0#>
```

Start a command count for commands 31 and 335

The command looks like this:

```
<102#START#4#31,335#>
```

The Network Security Processor issues a response similar to this:

```
<202#START#4#20121221 19:03:12###>
```

Get the number of times commands 31 and 335 have been successfully processed

The command looks like this:

```
<102#RETRIEVE#4##>
```

The Network Security Processor issues a response similar to this:

```
<202#RETRIEVE#4#20121221 20:07:25#20121221 20:09:12#
31=971,335=244#>
```

Stop the command counting test

The command looks like this:

```
<102#STOP#4##>
```

The Network Security Processor issues a response similar to this:

```
<202#STOP#4#20121221 20:09:12#20121221 20:09:24#31=88,335=6#>
```

Get CPU Utilization (Command 103)

This command returns the average CPU Utilization. It is very similar to command 1113. Unlike command 1113, this command requires the user to send a start message, and then a separate retrieve or stop message. The retrieve and stop messages will reply with the measured CPU utilization over the period. A sytem power on will delete any utilization history.

This command is enabled in the Network Security Processor's default security policy.

note This command is only allowed on the USB and Serial interfaces, and also on the Management ports.

Command

```
<103#Action##>
```

Response

```
<203#Action#Start Time#[End Time]#[CPU Busy %]#>
```

Calling Parameters

`103`

Field 0, the command identifier.

`Action`

Field 1, the action to be performed. The allowed values are:

Value	Description
START	Start monitoring.
RETRIEVE	Return the count value, reset the count value to zero and continue monitoring.
STOP	Stop monitoring.

Table 12-22 Command 103: Get CPU Utilization

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	103
1	Action	4, 5, 8	START, RETRIEVE, STOP

Responding Parameters

`203`

Field 0, the response identifier.

`Action`

Field 1, the action value supplied in field 1 of the command.

`Start Time`

Field 2, the date/time when the monitoring task was started.

The format is: `YYYYMMDD HH:MM:SS`
A start time value of 20160521 19:03:12, is May 21, 2016 7:03:12 PM.

`[End Time]`

Field 3, the date/time when the monitoring task was stopped or data was retrieved. This field will be empty when the action in field 1 contains the value START.

The format is: `YYYYMMDD HH:MM:SS`
An end time value of 20160521 20:03:12, is May 21, 2016 8:03:12 PM.

`[CPU Busy %]`

Field 4, the CPU busy percentage. This field will be empty when the action specified in field 1 contains the value START.

Table 12-23 Response 203: Get CPU Utilization

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	203
1	Action	4, 5, 8	START, RETRIEVE, STOP
2	Start Time	17	0 - 9, “:”
3	[End Time]	0, 17	0 - 9, “:”
4	[CPU Busy %]	0, 2	0 - 9

Usage Notes

- The RETRIEVE action resets the Start Time, and the CPU Busy % value to zero.
- Only one instance of this command can be running at a time. Attempts to start additional measurements will result in an error response.

- The RETRIEVE and STOP actions are only valid after a measurement has been started. If no measurement is running, sending a RETRIEVE or STOP action will result in an error response.

Examples

Start the measurement

```
<103#START#>

<203#START#20160511 21:50:29###>
```

Retrieve the average CPU utilization since Start

```
<103#RETRIEVE#>

<203#RETRIEVE#20160511 21:50:29#20160511 21:50:44#10#>
```

Retrieve the average CPU utilization since previous Retrieve

note Start Time is equal to the end time of the previous Retrieve

```
<103#RETRIEVE#>

<203#RETRIEVE#20160511 21:50:44#20160511 21:50:57#30#>
```

Stop the measurement

```
<103#STOP#>

<203#STOP#20160511 21:50:57#20160511 21:51:25#20#>
```

License Premium Value Commands and Options (Command 105)

This command provides the facility to license premium value commands and options in a specific Network Security Processor.

The serial number of the Network Security Processor is required when placing an order for a premium value commands and options. When the order is processed, a Command 105 for that specific serial number will be provided. Be sure you send the Command 105 you receive to the correct Network Security Processor. Field one of the response to a Command <9A#ID#> will contain the serial number.

This command updates non-volatile memory with the configuration. Premium value commands and options are not lost if the Network Security Processor is powered off. It is not necessary to send this command each time the Network Security Processor is powered on.

note After sending the command 105 to the Network Security Processor, you must also add the premium value command(s) or option(s) to Network Security Processor's security policy using either the SCA, or the SCA and commands 108 and 109. Resetting the Network Security Processor to factory state erases the license.

Command

```
<105#Serial Number#Encrypted Configuration#MAC#>
```

Response

```
<205#Status#Version#>[CRLF]
```

Calling Parameters

`105`

Field 0, the command identifier.

`Serial Number`

Field 1, the serial number of the Network Security Processor. In versions 1.13 and above, lowercase characters are allowed.

`Encrypted Configuration`

Field 2, the encrypted configuration. When decrypted by the Network Security Processor, this field defines the premium value commands and options to be enabled.

`MAC`

Field 3, the Message Authentication Code. The Network Security Processor validates the MAC before it processes the configuration.

Table 12-24 Command 105: Enable Premium Value Commands and Options

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	105
1	Serial Number	6	0 - 9, A - Z, a -z
2	Encrypted Configuration	various	0 - 9, A - F
3	MAC	9	0 - 9, A - F, space

Responding Parameters

`205`

Field 0, the command identifier.

`Status`

Field 1, the status of processing the command.

- COMPLETED indicates the command was successfully processed.
- MAC MISMATCH indicates that the MAC did not validate.
- CONF INVALID indicates that the decrypted configuration contained an error.
- SN MISMATCH indicates that the serial number of the Network Security Processor does not match the serial number in the command 105.

`Version`

Field 2, the version of the command.

Table 12-25 Response 205: Enable Premium Value Commands and Options

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	205
1	Status	12	COMPLETED, MAC MISMATCH, CONFIG INVALID, SN MISMATCH
2	Version	4	0 - 9, A - Z

Example

Using Command 105 to enable a premium value command or option. The serial number is 123456. This is not a valid example.

```
<105#123456#E7F35DA354A09F32#B65F 3CA0#>
```

The Network Security Processor returns a response similar to this:

```
<205#COMPLETED#VER1#>
```

Define Temporary Serial Number (Command 106)

Each Network Security Processor has a unique permanent serial number. This serial number is used to create a unique command 105, that when sent to the Network Security Processor, licenses premium value commands or options. The licensed premium value commands or options must then be enabled in the Network Security Processor's security policy using either the SCA or commands 108 and 109.

If a Network Security Processor that is configured with premium value commands or options fails, it will be replaced with an Network Security Processor that has a different permanent serial number. To quickly configure the replacement Network Security Processor with the same premium value commands or options as those licensed in the failed Network Security Processor, the replacement Network Security Processor must be loaded with a temporary serial number which is the serial number of the failed Network Security Processor. This allows the replacement Network Security Processor to accept the command 105 created for the failed Network Security Processor.

Commands 106 and 107 operate as a pair, they are used to temporarily load the serial number of another Network Security Processor into a replacement Network Security Processor.

note If the SCA is used to initialize the Network Security Processor, you can use the SCA Set Temporary Serial Number feature instead of commands 106 and 107.

The temporary serial number is stored when the Network Security Processor successfully processes a command 107. If power is lost before the command 107 is processed, the temporary serial number is erased. If this should happen, you must send the command 106 again, and then send the corresponding command 107.

warning This temporary serial number is valid for 120 hours (5 days) from the time that the temporary serial number was set in the Network Security Processor. For example, if the Network Security Processor receives the temporary serial number on Wednesday at 6:30 AM, the temporary serial number will expire at 6:00AM on Monday. **If the Network Security Processor does not receive a command 105 based on its permanent serial number within 120 hours, all premium value commands and options are reset to the Network Security Processor's default security policy.** To prevent this from happening, you must perform these two steps within this 120 hour time period:

1) Contact HPE Security - Data Security technical support and provide the serial numbers of the failed and replacement Network Security Processor. They will generate a new command 105 based on the replacement Network Security Processor's serial number.

2) Send this new command 105 to the Network Security Processor.

The temporary serial number is erased when the Network Security Processor receives a command 105 based on its permanent serial number.

Command

```
<106#Permanent Serial Number#Temporary Serial Number#>
```

Response

```
<206#Status#Permanent Serial Number#Temporary Serial Number#
Challenge#Check Digits#>[CRLF]
```

Calling Parameters

`106`

Field 0, the command identifier.

`Permanent Serial Number`

Field 1, the permanent serial number.

`Temporary Serial Number`

Field 2, the temporary serial number you wish to load into the replacement Network Security Processor. This should be the permanent serial number of the failed Network Security Processor. You can obtain this value from the Command 105 issued for the failed Network Security Processor or from the label on the back of the failed Network Security Processor.

Table 12-26 Command 106: Define Temporary Serial Number

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	106
1	Permanent Serial Number	6	0 - 9, A - Z
2	Temporary Serial Number	6	0 - 9, A - Z

Responding Parameters

`206`

Field 0, the command identifier.

`Status`

Field 1, the status of processing the command.

- COMPLETED indicates that the command completed successfully.

- SN MISMATCH indicates that the permanent serial in the Network Security Processor does not match the permanent serial number in the command 106. Use command <9A#ID#> to obtain the permanent serial number.
- TMP EXISTS indicates that the Network Security Processor already has a temporary serial number. If the wrong temporary serial number has been loaded you must reset the Network Security Processor to factory state to erase it.

`Permanent Serial Number`

Field 2, the Network Security Processor permanent serial number.

`Temporary Serial Number`

Field 3, the temporary serial number input as field 2 of the command. This field will be empty unless the status field in the response contains COMPLETED.

`Challenge Number`

Field 4, the challenge number. This random value must be encrypted under the MFK. You can use the SCA Encrypt Challenge feature to perform this task.

The encrypted value is used in field 1 of command 107. This field will be empty unless the status field in the response contains COMPLETED.

`Check Digits`

Field 5, the check digits of the challenge number. Use this value to confirm that you have correctly entered the challenge into the SCA. This field will be empty unless the status field in the response contains COMPLETED.

Table 12-27 Response 206: Define Temporary Serial Number

Field	**Contents**	**Length (bytes)**	**Legal Characters**
0	Response identifier	3	206
1	Status	11	COMPLETED, SN MISMATCH, TMP EXISTS
2	Permanent Serial Number	6	0 - 9, A - Z
3	Temporary Serial Number	0, 6	0 - 9, A - Z. This field will be empty unless field 1 contains COMPLETED.
4	Challenge Number	0, 16	0 - 9, A - F. This field will be empty unless field 1 contains COMPLETED.
5	Check Digits	0, 4	0 - 9, A - F. This field will be empty unless field 1 contains COMPLETED.

Usage Notes

- This command will return an error response if the Network Security Processor is not in a security association and does not contain a Master File Key.

Example

Using Command 106 to define a temporary serial number.

The command looks like this:

```
<106#123456#654321#>
```

The Network Security Processor returns a response similar to this:

```
<206#COMPLETED#123456#654321#7C54B39AAE85A011#A371#>
```

Confirm Temporary Serial Number (Command 107)

Command 107 is used to implement the temporary serial number you defined using command 106.

note If the SCA is used to initialize the Network Security Processor, you can use the SCA Set Temporary Serial Number feature instead of commands 106 and 107.

This command should only be used if you have premium value commands and options enabled with command 105, and the Network Security Processor for which the command 105 was generated has failed.

Command

```
<107#Cryptogram of the Challenge#>
```

Response

```
<207#Status#Permanent Serial Number#Temporary Serial Number#>[CRLF]
```

Calling Parameters

`107`

Field 0, the command identifier.

`Cryptogram of the Challenge`

Field 1, the challenge, from the response to command 106, encrypted under the Master File Key (MFK). This cryptogram of the challenge can be generated using the SCA.

Table 12-28 Command 107: Implement Temporary Serial Number

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	107
1	Cryptogram of the Challenge	16	0 - 9, A - F

Responding Parameters

`207`

Field 0, the command identifier.

`Status`

Field 1, the status can be one of these values:

- COMPLETED indicates that the command was successfully processed.
- NO TMP SN indicates that the Network Security Processor does not have a challenge number or serial number in memory. Repeat command 106.
- BAD CHALLENGE indicates that the challenge was not correct. Make sure the check digits of the challenge entered into the SCA match those returned in the 206 response.

`Permanent Serial Number`

Field 2, the permanent serial number of the Network Security Processor.

`Temporary Serial Number`

Field 3, the temporary serial number defined with command 106. This field will be empty unless the status field in the response contains COMPLETED.

Table 12-29 Response 207: Implement Temporary Serial Number

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	207
1	Status	9, 13	COMPLETED, NO TMP SN, BAD CHALLENGE
2	Permanent Serial Number	6	0 - 9, A - Z
3	Temporary Serial Number	0, 6	0 - 9, A - Z. This field will be empty unless field 1 contains COMPLETED.

Example

Using Command 107 to implement a temporary serial number.

This is not a valid example.

The command looks like this:

```
<107#B203A98A64C8F906#>
```

The Network Security Processor returns a response similar to this:

```
<207#COMPLETED#123456#654321#>
```

Define Security Policy (Command 108)

This command provides the facility to enable and disable commands and options in the Network Security Processor. Most of the commands and options supported in the Network Security Processor are disabled in the Network Security Processor's default security policy. To use these commands and options, you must enable them with commands 108 and 109.

note If the SCA is used to initialize the Network Security Processor, you can use the SCA NSP Configuration Management feature instead of commands 108 and 109.

Commands 108 and 109 work as a pair to define and then implement a security policy. Use command 108 to define the security policy. The response to the 108 command is a challenge that must be encrypted under the Master File Key (MFK). See the Encrypt Challenge section of the *Secure Configuration Assistant-3 Users Guide*. This encrypted challenge is then used as an input to command 109 to implement the security policy.

If the Network Security Processor is power cycled after the command 108 has been processed, but before the command 109 has been processed, the security policy defined by command 108 will not be implemented. Before a security policy can take effect, commands 108 and 109 must be successfully processed as a pair, without an intervening power cycle.

Using this command it is possible to disable the serial number validation and sequence counter checking, if both of these security parameters are disabled, a warning message will be returned in the response message. You must acknowledge this message in the subsequent command 109.

See Appendix C, “Summary of Commands and Options” for a complete list of commands and options that can be enabled or disabled using this command.

Premium value commands and options, enabled with command 105, must be added to Network Security Processor's security policy with the SCA and commands 108 and 109 before they can be used by the Network Security Processor.

Command Counting

The command count table resides in non-volatile RAM, it is maintained even if the Network Security Processor loses power. The table is constructed such that a maximum of nine cryptographic commands can be counted.

Command 108 supports the ability to specify a command count. The count value must be in the range of 1 to 4 billion (4,000,000,000). Utility commands and options cannot be counted. Premium value commands must be licensed with a command 105 before they can be counted.

Each time the Network Security Processor successfully processes a command that is being counted, the count value is decremented by 1. Commands that are not successfully processed by the Network Security Processor, such as commands that contain syntax error(s) that result in an error response returned from the Network Security Processor, are not counted.

caution Once the count value reaches zero, the Network Security Processor will return an error <00#0300xx#> instead of processing the command. The command <9A#COUNT#> can be used to obtain the current count value for all commands in the command count table.

The count value is specified using the letter “N” or “n”, followed by the count value (in decimal). When a command count has been specified, the command is automatically enabled in the Network Security Processor’s security policy for that number of executions, any previously defined count value is replaced by the count value currently being specified.

If a command that is currently being counted is disabled in the Network Security Processor’s security policy, the count value for that command remains in the command count table, such that, if the command is ever enabled the count value will be applied.

A command that is currently being counted can be removed from the command count table using the letter “R” or “r”. When a command is removed from the command count table it is also disabled in the Network Security Processor’s security policy.

When a Network Security Processor is reset to factory state, or the Network Security Processor’s security policy is reset to the default, all data stored in the command count table is erased.

See Examples, for some security polices that demonstrate command counting.

Command

```
<108#Security Policy#>
```

Response

```
<208#[Warning Message]#Left Challenge#Left Challenge Check Digits#
Right Challenge#Right Challenge Check Digits#Counter#
Sequence Number#Serial Number#>[CRLF]
```

Calling Parameters

`108`

Field 0, the command identifier.

```
Security Policy
```

Field 1, the security policy string. The security policy is a string that defines what commands and options are enabled or disabled. The format is:

Command ID followed by an equal sign “=”, followed by a one-digit action flag “e” or “E” for enable, and “d” or “D” for disable. The Command ID must be upper case. For example, to enable command 1A, the security policy string would be 1A=e, or 1A=E. The security policy strings 1a=e or 1a=E, are **not** correct because the command ID is not upper case.

Options are surrounded by parenthesis, they must be uppercase, for example (6C) is correct, (6c) is not correct. The option is followed by an equal sign “=”, followed by a value which is surrounded by double quotes. The option value can be either upper or lower case. For example, to enable option 6C the security policy string would be (6C)=”e”, or (6C)=”E”.

If multiple commands or options are to be enabled or disabled in the same security policy string, they must be separated by a semicolon”;”, embedded spaces are not allowed. For example, 1A=e;(6C)=”D”. See Examples for some typical security policies.

A command or option can only have one value for a given security policy. For example, if a security policy string enables a command then subsequently disables it in the same string, an error 20 will be returned.

When setting options E0 and E1, be aware that any previous value will be replaced with the current value. If for example your existing value for option E0 is “VD”, and you wish to add P for a PIN Encryption Key, you must specify option E0 as (E0)=“VDP”. If you specify it as (E0)=”P”, only PIN Encryption Keys can be imported.

If this field contains the word “FACTORY”, all commands and options will be set to the factory default security policy. If necessary, you can use this value to quickly undue a security policy and return the Network Security Processor to a known state. The word factory is not case-sensitive.

Table 12-30 Command 108: Define Security Policy

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	108
1	Security Policy	0 - 4000	0 - 9, A - F, FACTORY

Responding Parameters

```
208
```

Field 0, the command identifier.

```
[Warning Message]
```

Field 1, a warning message that indicates that both the serial number checking option (6E) and sequence counter checking option (6F) have been disabled, either prior to, or as a result of, this command. This warning field must be acknowledged in command 109 as part of the response to the challenge. The warning message is:

```
SECURITY PRECAUTION: Are you sure?
```

This message will only appear when options (6E) and (6F) are enabled in the security policy.

`Left Half Challenge`

Field 2, the left half of the challenge. This value must be encrypted under the MFK. Use the Encrypt Challenge feature in the SCA to encrypt the challenge.

`Left Half Challenge Check Digits`

Field 3, the check digits are the leftmost 4 digits of the result from encrypting zeros using the Left Half Challenge. If option 88 is enabled, this field will contain six check digits. Use this value to confirm that you have correctly entered the left half of the challenge into the SCA.

`Right Half Challenge`

Field 4, the right half of the challenge. This value must be encrypted under the MFK. Use the Encrypt Challenge feature in the SCA to encrypt the challenge.

`Right Half Challenge Check Digits`

Field 5, the check digits are the leftmost 4 digits of the result from encrypting zeros using the Right Half Challenge. If option 88 is enabled, this field will contain six check digits. Use this value to confirm that you have correctly entered the right half of the challenge into the SCA.

`Counter`

Field 6, the number of times an attempt has been made to update the security policy. It is displayed so you can monitor the number of times command 108 has been attempted. The counter is incremented each time a command 108 is received without a subsequent command 109. The counter is reset to zero when a command 108 and 109 pair have been successfully processed. This value is maintained in volatile memory, therefore each time the Network Security Processor is powered on this value will be reset to zero.

`Sequence Number`

Field 7, the number of times the security policy has been successfully updated. This value is used in processing of the security policy. It is included in the response so you can keep track of the number of times the Network Security Processor security policy has been updated. This value is stored in non-volatile memory and is incremented as the result of successfully processing a command 109.

`Serial Number`

Field 8, the serial number of the unit. This value is unique to each Network Security Processor, and is used in processing the security policy. It is included in the response so you can keep track of which unit to send the subsequent command 109. This value is stored in non-volatile memory and cannot be changed.

Table 12-31 Response 208: Define Security Policy

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	208
1	[Warning Message]	0 or 34	
2	Left Half Challenge	16	0 - 9, A - F
3	Left Half Check Digits	4 or 6	0 - 9, A - F
4	Right Half Challenge	16	0 - 9, A - F
5	Right Half Check Digits	4 or 6	0 - 9, A - F
6	Counter	1 - 4	0 - 9
7	Sequence Number	16	0 - 9
8	Serial Number	6 - 7	variable

Examples

Using Command 108 to define several security policies

If you use Hyperterminal to communicate with the Network Security Processor, be advised that the Hyperterminal feature Paste to Host (Control V) will either truncate or change the value of the double quote character. Therefore, you cannot copy (Control C) and paste (Control V) examples that have an option ID and value. Instead, you must manually enter the command 108 into Hyperterminal. An indication you are experiencing this problem is that you will get an error 23 as a response instead of the response listed.

Enable specific commands and options

This example illustrates using Command 108 to enable the following commands and options:

Commands: 30 and 90
Options: 66, set the minimum PIN length to 6, and enable 1key-3DES keys (single- length)

The command looks like this:

```
<108#30=e;90=E;(66)=”e”;(A0)=”6”;(6C)=”E”#>
```

The Network Security Processor returns a response similar to this:

```
<208##23A4DF7983208992#4AF3#12C42BDAD34798FF#7BB2#1#
0000000000000001#A7PV87#>
```

Disable specific commands and options

This example illustrates using Command 108 to disable the following commands and options:

Commands: 30 and 90
Option: 66

The command looks like this:

```
<108#30=D;90=d;(66)="d"#>
```

The Network Security Processor returns a response similar to this:

```
<208##23A4DF7983208992#4AF3#12C42BDAD34798FF#7BB2#1#
0000000000000001#A7PV87#>
```

Enable and disable commands and options with a single command

This example illustrates using Command 108 to enable the following commands and options:

Commands: 30 and 90
Option: 66
Import Key Types: P and M
Export Key Types: V and d

And disable the following commands and options:

Commands: 10 and 98.
Options: 60 and 65.

And to set the minimum PIN length to 6, and to set the sanity indicator to "L".

The command looks like this:

```
<108#30=e;90=e;(66)="e";(E0)="PM";(E1)="Vd";10=d;98=d;
(60)="d";(65)="d";(A0)="6";(A1)="L"#>
```

The Network Security Processor returns a response similar to this:

```
<208##23A4DF7983208992#4AF3#12C42BDAD34798FF#7BB2#1#
0000000000000001#A7PV87#>
```

Disable the sequence number and serial number validation.

This example produces a warning message because the security policy disables both the sequence number and serial number validation, by enabling options (6E) and (6F), respectively.

The command looks like this:

```
<108#(6E)="e";(6F)="e"#>
```

The Network Security Processor returns a response similar to this:

```
<208#SECURITY PRECAUTION: Are you sure?# 23A4DF7983208992#
4AF3#12C42BDAD34798FF#7BB2#1#0000000000000001#A7PV87#>
```

Enable the factory security policy

This example shows how to reinstate the factory security policy.

The command looks like this:

```
<108#FACTORY#>
```

The Network Security Processor returns a response similar to this:

```
<208##23A4DF7983208992#4AF3#12C42BDAD34798FF#7BB2#1#
0000000000000001#A7PV87#>
```

Enable command counting

This example shows how to enable commands 10 for 100 executions and command 31 for 50,000 executions.

The command looks like this:

```
<108#10=n100;31=N50000#>
```

The Network Security Processor returns a response similar to this:

```
<208##0DC2D50EF492E33D#30D6#DAF29816C8B96843#9EBC#1#
0000000000000004#A7PV87#>
```

Disable a command and removing it from the counter table

This example shows how to remove command 10 from the command count table and disable it in the Network Security Processor's security policy.

```
<108#10=r#>
```

The Network Security Processor returns a response similar to this:

```
<208##AD4029E607385DDA#99D5#CEDF326710E08F49#0D37#1#
0000000000000005#A7PV87#>
```

Confirm Security Policy (Command 109)

Command 109 is used to implement the security policy you defined using command 108.

note If the SCA is used to initialize the Network Security Processor, you can use the SCA NSP Configuration Management feature instead of commands 108 and 109.

Command

```
<109#[Warning Acknowledgement]#Cryptogram of the Challenge#>
```

Response

```
<209#Security Policy#Sequence Number#Serial Number#>[CRLF]
```

Calling Parameters

`109`

Field 0, the command identifier.

`[Warning Acknowledgement]`

Field 1, the warning acknowledgment. If the security policy, defined in the 108 command, disabled both the sequence number and serial number validation, options (6E) and (6F), a warning message “SECURITY PRECAUTION: Are you sure?” was included in the 208 response. You must supply the following warning acknowledgment message:

```
I accept
```

before the Network Security Processor’s security policy will be implemented. This field is not case sensitive. Leave this field empty if the Network Security Processor’s security policy does not disable both of these options.

`Cryptogram of the Challenge`

Field 2, the challenge encrypted under the MFK. This cryptogram of the challenge can be generated using the SCA feature Encrypt Challenge.

Table 12-32 Command 109: Confirm Security Policy

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	3	109
1	[Warning Acknowledgment]	0 or 8	I accept
2	Cryptogram of the Challenge	32	0 - 9, A - F

Responding Parameters

`209`

Field 0, the command identifier.

`Security Policy`

Field 1, the security policy that was just implemented. If the security policy is defined by several command 108 commands, this field will only show the security policy for the most recent 108 command. You can use the Network Security Processor Configuration Status (Command 9A) to obtain a complete list of commands and options enabled and disabled in the Network Security Processor.

`Sequence Number`

Field 2, the number of times the security policy has been successfully updated. This value is used in processing of the security policy. It is displayed so you can keep track of the number of times each of your Network Security Processor's security policy has been updated.

`Serial Number`

Field 4, the serial number of the unit. This value is unique to each Network Security Processor, and is used in processing of the security policy. This value is displayed so you can be certain which Network Security Processor has had its security policy updated.

Table 12-33 Response 209: Confirm Security Policy

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	3	209
1	Security Policy	variable, 4000 characters maximum	variable
2	Sequence Number	16	0 - 9
3	Serial Number	8	variable, ASCII

Example

Implement a security policy

The command looks like this:

```
<109##203A98A64C8F900C62E1E8368E43A751#>
```

The Network Security Processor returns a response similar to this:

```
<209#30=e;90=e;32=e;37=e;(66)=”e”;(A0)=”6”;10=d;30=d;
98=d;(65)=d#0000000000000001#A7PV87#>
```

Get ID of Current Image (Command 1101)

Command 1101 allows you to obtain the software image, image CRC checksum, and the product code of the Network Security Processor.

In version 1.50, the image name has been changed, it starts with “HPE Atalla”. You can enable option 028 if your host application expects the image name to start with “HP Atalla”.

Command

```
<1101#>
```

Response

```
<2101#Image ID#Image CRC Checksum#Product Code#>[CRLF]
```

Calling Parameters

`1101`

Field 0, the command identifier.

Table 12-34 Command 1101: Get ID of Current Image

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	1101

Responding Parameters

`2101`

Field 0, the response identifier.

`Image ID`

Field 1, the Network Security Processor’s image ID, which consists of the image name, version number, and creation date.

`Image CRC Checksum`

Field 2, the CRC checksum of the Network Security Processor image.

`Product Code`

Field 3, the Network Security Processor’s product code.

Table 12-35 Response 2101: Get ID of Current Image

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	4	2101
1	Image ID	0 - n	Any except # < >
2	Image CRC Checksum	4	0 - 9, A - F
3	Product Code	1	1

Example

Use Command 1101 to obtain the version of the image in the Network Security Processor.

The command looks like this:

```
<1101#>
```

The Network Security Processor issues a response similar to this:

```
<2101#HPE Atalla A10160-AKB Version: 1.60, Date: Oct 25 2016,
Time: 11:32:09#DA77#1#>
```

When option 028 is enabled, the Network Security Processor issues a response similar to this:

```
<2101#HP Atalla A10160-AKB Version: 1.60, Date: Oct 25 2016, Time:
11:32:09#DA77#1#>
```

Get Temporary Serial Number Information (Command 1104)

Command 1104 allows you to obtain the temporary serial number and the number of hours remaining before it expires.

Command

```
<1104#>
```

Response

```
<2104#Temporary Serial Number#Remaining Hours#>
```

Calling Parameters

`1104`

Field 0, the command identifier.

Table 12-36 Command 1104: Get Temporary Serial Number Information

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	1104

Responding Parameters

`2104`

Field 0, the response identifier.

`Temporary Serial Number`

Field 1, the temporary serial number.

`Remaining Hours`

Field 2, the number of hours before the temporary serial number expires.

Table 12-37 Response 2104: Get Temporary Serial Number Information

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	4	2104
1	Temporary Serial Number	0, 6	A - Z, 0 - 9
2	Remaining Hours	1 - 3	0 - 120

Usage Notes

If no temporary serial number has been defined or if it has expired, the response to this command will be:

```
<2104##0#>
```

Example

Use Command 1104 to obtain the temporary serial number information.

The command looks like this:

```
<1104#>
```

The Network Security Processor issues a response similar to this indicating that the temporary serial number is 123456 and it will expire in 48 hours.

```
<2104#123456#48#>
```

Get System Configuration Information (Command 1110)

Command 1110 allows you to obtain the Network Security Processor system software information, and cryptographic subsystem software version information.

In version 1.50, the image name has been changed, it starts with “HPE Atalla”. You can enable option 028 if your host application expects the image name to start with “HP Atalla”.

Command

```
<1110#>
```

Response

```
<2110#Software Version Information#Image ID#Image CRC Checksum#
Product Code#>[CRLF]
```

Calling Parameters

`1110`

Field 0, the command identifier.

Table 12-38 Command 1110: Get System Configuration Information

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	1110

Responding Parameters

`2110`

Field 0, the response identifier.

`Software Version Information`

Field 1, the Network Security Processor’s base kernel software’s version number, and date and time it was created.

`Image ID`

Field 2, the Network Security Processor’s image ID, which consists of the image name, version number, and creation date.

`Image CRC Checksum`

Field 3, the CRC checksum of the Network Security Processor image.

`Product Code`

Field 4, the Network Security Processor's product code.

Table 12-39 Response 2110: Get System Configuration Information

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	4	2110
1	Software version information	variable	Any except #, <, or >
2	Image ID	0 - n	Any except # < >
3	Image CRC Checksum	4	0 - 9, A - F
4	Product Code	1	1

Example

Use Command 1110 to obtain the system configuration information of the Network Security Processor.

The command looks like this:

```
<1110#>
```

The Network Security Processor returns a response similar to this:

```
<2110#Axx160, Version: 1.60, Date: Oct 25 2016, Time: 11:33:15#HPE
Atalla A10160-AKB Version: 1.60, Date: Oct 25 2016, Time:
11:32:09#DA77#1#>
```

When option 028 is enabled, the Network Security Processor issues a response similar to this:

```
<2110#Axx160, Version: 1.60, Date: Oct 25 2016, Time: 11:33:15#HP
Atalla A10160-AKB Version: 1.60, Date: Oct 25 2016, Time:
11:32:09#DA77#1#>
```

Get System Date and Time (Command 1111)

Command 1111 allows you to obtain the Network Security Processor’s system date and time in Universal Coordinated Time. The date and time are set during the manufacturing process and cannot be changed.

Command

```
<1111#>
```

Response

```
<2111#YYMMDDHHMMSS#>[CRLF]
```

Calling Parameters

`1111`

Field 0, the command identifier.

Table 12-40 Command 1111: Get System Date and Time

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	1111

Responding Parameters

`2111`

Field 0, the response identifier.

`YYMMDDHHMMSS`

Field 1, two digit year, two digit month, two digit day, two digit hour, two digit minute, two digit second.

Table 12-41 Response 2111: Get System Date and Time

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	4	2111
1	YYMMDDHHMMSS	12	0 - 9

Example

Obtain the Network Security Processor system date and time

The command looks like this:

```
<1111#>
```

The Network Security Processor returns a response similar to this:

```
<2111#160525115300#> (May 25, 2016 11:53:00)
```

Get Average CPU Utilization (Command 1113)

Command 1113 allows you to obtain a percentage value which is the average CPU utilization for the Network Security Processor. The time period for the measurement is specified in the command. At the end of the time period the Network Security Processor returns a response which contains a percentage value indicating the average CPU utilization.

note This command is only allowed on the USB and Serial interfaces, and also on the Management ports.

Command

```
<1113#Test Period#>
```

Response

```
<2113#Percent Utilized#>[CRLF]
```

Calling Parameters

`1113`

Field 0, the command identifier.

`Test Period`

Field 1, the number of seconds that the test will run. The minimum value is 1, the maximum value is 10.

Table 12-42 Command 1113: Get Average CPU Utilization

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	1113
1	Test Period	1 - 2	1 - 10

Responding Parameters

`2113`

Field 0, the response identifier.

`Percent Utilized`

Field 1, the average CPU utilization during the test period.

Table 12-43 Response 2113: Get Average CPU Utilization

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	4	2113
1	Percent Utilized	1-3	0 - 100

Usage Notes

The Network Security Processor does not return a response until the test completes.

Example

Obtain the average CPU utilization for a 10 second time period

The command looks like this:

```
<1113#10#>
```

The Network Security Processor returns a response similar to this:

```
<2113#37#>
```

Get System Information (Command 1120)

Command 1120 allows you to obtain the Network Security Processor serial number, product ID, system software information, and a personality version.

Command

```
<1120#>
```

Response

```
<2120#SerialNumber#ProductID#LoaderVersion#PersonalityVersion#>
[CRLF]
```

Calling Parameters

`1120`

Field 0, the command identifier.

Table 12-44 Command 1120: Get System Information

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	1120

Responding Parameters

`2120`

Field 0, the response identifier.

`SerialNumber`

Field 1, the serial number of the Network Security Processor.

`ProductID`

Field 2, the model number of the Network Security Processor.

`LoaderVersion`

Field 3, the version number of the program used to load system images into the Network Security Processor.

`PersonalityVersion`

Field 4, the Ax150 personality major version number that was used as a base to create this version. For example a value of 3.70 in this field indicates that the Ax150 version 3.70 was used as a base to create this Ax160 version.

Additional capability has been added to the Ax160 version when the value in this field ends with the letter “X”. For example a value of 3.7x in this field indicates that features and functions added after Ax150 version 3.70 but before Ax150 version 3.80 are present in this Ax160 version.

Customers that have both Ax150 and Ax160 Network Security Processors can use this command to verify that both Network Security Processor models are running functionally equivalent software.

Table 12-45 Response 2120: Get System Information

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	4	2120
1	SerialNumber	variable	0 - 9, A - Z
2	ProductID	variable	0 - 9, A - Z
3	LoaderVersion	variable	0 - 9, A - Z
4	PersonalityVersion	variable	0 - 9, A - Z

Example

Use Command 1120 to obtain the system information of the Network Security Processor.

The command looks like this:

```
<1120#>
```

The Network Security Processor returns a response similar to this:

```
<2120#SerialNumber=JL06RP#ProductID=A10160#LoaderVersion=0.67 02
APR 2015 17:04:43#PersonalityVersion=AKB 3.9X#>
```

Get Battery Life Remaining (Command 1216)

Command 1216 – Use this command to return the expected number of days remaining for the internal batteries. Once this value reaches zero, the batteries should be replaced by an HPE technician.

Command

```
<1216#1#>
```

Response

```
<2216#1#Days Remaining#>[CRLF]
```

Calling Parameters

`1216`

Field 0, the command identifier.

`1`

Field 1, this field must contain the number 1.

Table 12-46 Command 1216: Get Battery Life Remaining

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	1216
1	1	1	1

Responding Parameters

`2216`

Field 0, the response identifier.

`1`

Field 1, the value specified in command field 1.

`Days Remaining`

Field 2, the number of days remaining of battery life, in the range of 0 - 700. The value is the approximate number of days until the battery voltage level drops to an unacceptable level.

Table 12-47 Response 2216: Get Battery Life Remaining

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	2216
1	1	1	1
2	Days Remaining	4	0-9

Example

Get battery life remaining

The command looks like this:

```
<1216#1#>
```

The NSP returns a response similar to this (which indicates that there are 200 days of battery life remaining):

```
<2216#1#200#>
```

Return IP Address of Network Security Processor (Command 1221)

Command 1221 – Use this command to return the IP Address of the Network Security Processor.

Command

```
<1221#>
```

Response

```
<2221#NIC1 IP Address#[NIC2 IP Address#]>[CRLF]
```

Calling Parameters

`1221`

Field 0, the command identifier.

Table 12-48 Command 1221: Return IP Address of Network Security Processor

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	1221

Responding Parameters

`2221`

Field 0, the response identifier.

`NIC1 IP Address`

Field 2, the NIC1 IP Address of Network Security Processor.

`[NIC2 IP Address#]`

Field 1, the NIC2 IP Address of Network Security Processor. This field will be present when:

- Option 87 has been enabled in the Network Security Processor’s security policy.
- The Network Security Processor was powered on with a config.prm file that contained a valid IP address in the TCPIP parameter IPADDR_2.

Table 12-49 Response 2221: Return IP Address of Network Security Processor

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	2221
1	NIC1 IP Address	varies	0 - 9
2	[NIC2 IP Address]	varies	0 - 9

Example

Return IP address of Network Security Processor

When only one NIC is configured, the Network Security Processor returns a response similar to this:

```
<2221#127.0.0.1#>
```

When both NICs are configured, the Network Security Processor returns a response similar to this:

```
<2221#127.0.0.1#127.0.1.2#>
```

TCP/IP Socket Information (Command 1223)

Command 1223 – Use this command to return the number of sockets on the Network Security Processor that are available for new connections, the total number of new sockets the Network Security Processor can open, and the number of sockets available for reconnect from the host that sent this command.

In version 1.14 and above, option 023 applies to this command. All of the other options that can be defined by command 101 do not apply to this command.

Command

```
<1223#[Port#][NIC#]>
```

Response

```
<2223#Remaining Sockets#Total Sockets#Reconnect Sockets#>[CRLF]
```

Calling Parameters

`1223`

Field 0, the command identifier.

`[Port#]`

Field 1, this field is optional if field 2 is not included in the command. This field is required if field 2 is included in the command. If present, socket information will be returned in the response for the port number specified in this field (PORT_ASCII, PORT_STATUS or PORT_MANAGEMENT). If the command is received by the Network Security Processor on either the serial or USB port, and field 2 is not specified, socket information will be returned in the response for the NIC1 port number specified in this field.

If this field is not present, and the command was received by either NIC1 or NIC2, socket information on the port number that received the command is returned in the response. If the command is received by the Network Security Processor on either the serial or USB port, and this field is not present, socket information for the NIC1 PORT_ASCII will be returned in the response.

`[NIC#]`

Field 2, this field is optional. If present, socket information will be returned in the response for the NIC specified in this field (1 or 2).

If this field is not present, socket information for the NIC that received the command is returned in the response. If the command is received by the Network Security Processor on either the serial or USB port, socket information will be returned in the response for the NIC1 port number specified in field 1 of the command.

Table 12-50 Command 1223: TCP/IP Socket Information

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	1223
1	[Port#]	1 - 5	0 - 9
2	[NIC#]	1	1 or 2

Responding Parameters

`2223`

Field 0, the response identifier.

`Remaining Sockets`

Field 1, the number of sockets available for a new connection. This value is the difference between the total number of sockets minus the number of sockets opened on the Network Security Processor. For example, if the total number of sockets is 16, and the application running on host A has 6 open sockets on the Network Security Processor, and the application running on host B has 3 open sockets on the Network Security Processor, this field would contain the value 7.

`Total Sockets`

Field 2, the number of TPC/IP sockets available for use in the Network Security Processor. This value is the number of sockets specified in the MAX_CLIENTS_ASCII or MAX_CLIENTS_ASCII_2 parameter in the CONFIG.PRM file.

`Reconnect Sockets`

Field 3, the number of reconnect sockets is equal to the number of Network Security Processor sockets that are connected to the host that sent the <1223#> command. If reconnect sockets are not enabled, this field will contain the letter “X”.

If a specific host establishes 10 new socket connections with the Network Security Processor, it has 10 reconnect sockets available. Reconnect sockets are used by the Network Security Processor only when all available sockets are in use.

Here is an example of when reconnect sockets are used. Assume that the Network Security Processor is configured for 16 sockets, and host A has 11 sockets open on the Network Security Processor, and host B has 3 sockets open on the Network Security Processor; 14 of the 16 possible sockets are in use on the Network Security Processor.

Host B loses power which leaves 3 sockets on the Network Security Processor in a hung state. Host B is immediately restarted and attempts to reconnect to the Network Security Processor, the first 2 socket open requests from Host B will be granted as new socket connects by the Network Security Processor, whereas the third socket open request from Host B will be granted as a reconnect socket by the Network Security Processor.

Any attempts by a host other than A or B to connect with the Network Security Processor will fail, as all available sockets are in use (Host A is using 11, and Host B is using 5, three of which are hung). At this point, Host A has 11 reconnect sockets available, and Host B has 2 reconnect sockets available.

Assume that another application on Host A now tries to establish 12 socket connections to the Network Security Processor, only the first 11 will be granted as reconnect sockets, the 12th open request will fail, as there are no available sockets on the Network Security Processor and Host A has used all of its reconnect sockets.

Assume another application on Host B now tries to establish 4 socket connections to the Network Security Processor, only the first 2 will be granted as reconnect sockets, the last 2 open requests will fail, as there are no available sockets and Host B has used all of its reconnect sockets.

After the KEEP_ALIVE_TIME expires (default is 20 minutes) the Network Security Processor detects that the 3 sockets originally established with Host B are hung, they are deleted from the Network Security Processor, and the number of available sockets is set to 3.

Table 12-51 Response 2223: TCP/IP Socket Information

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	2223
1	Remaining Sockets	1 - 2	0 - 9
2	Total Sockets	1 - 2	0 - 9
3	Reconnect Sockets	1 - 2	0 - 9, X

Example

Use Command 1223 to return the number of available sockets on the Network Security Processor.

The command looks like this:

```
<1223#>
```

The Network Security Processor returns the following response:

```
<2223#10#16#6#>
```

which indicates that there are 10 sockets available (six sockets in use on the Network Security Processor) out of a total of 16, and for this host there are six sockets available for reconnect.

Get Key Check Digits (Command 1226)

Command 1226 – Use this command to obtain the check digits of the Master File Key and Pending Master File Key you have loaded into the Network Security Processors non-volatile key table.

Command

```
<1226#>
```

Response

```
<2226#[MFK1=xxxx]#[PMFK1=xxxx]#####>[CRLF]
```

Calling Parameters

`1226`

Field 0, the command identifier.

Table 12-52 Command 1226: Get Key Check Digits

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	1226

Responding Parameters

`2226`

Field 0, the response identifier.

`[MFK1=xxxx]`

Field 1, Master File Key check digits.

`[PMFK1=xxxx]`

Field 2, Pending Master File Key check digits.

Table 12-53 Response 2226: Get Key Check Digits

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	2226
1	Master File Key check digits.	Variable	0 - 9, A - F, “=”, “MFK1””
2	Pending Master File Key check digits.	Variable	0 - 9, A - F, “=”, “PMFK1”

Example

The following example illustrate using Command 1226 to obtain check digits.

The command looks like this:

```
<1226#>
```

The Network Security Processor returns a response similar to this:

```
<2226#MFK1=B196#PMFK1=7A2E#####>
```

Reset to Factory State (Command 1227)

Command 1227 – Use this command to reset the Network Security Processor to the factory state. This command erases the user-defined security policy, and all user-defined keys. The default security policy is restored. You should use this command only when you need to add the Network Security Processor to a security association. Use Command 1228 to confirm your request to reset the Network Security Processor to the factory state.

note This command will only be processed if it is received from either the USB or serial port.

Command

```
<1227#RESET_TO_FACTORY_STATE#[MODE#]>
```

Response

```
<2227#nnnnnn#>[CRLF]
```

Calling Parameters

`1227`

Field 0, the command identifier.

`RESET_TO_FACTORY_STATE`

Field 1, statement sent to the Network Security Processor.

`[MODE#]`

Field 2, an optional field. When this field is present and contains the word “CLEAR”, the Network Security Processor will erase its security audit log.

Table 12-54 Command 1227: Reset to Factory State

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	1227
1	RESET_TO_FACTORY_STATE	23	Same as “Contents”
2	[MODE#]	0, 5	CLEAR

Responding Parameters

`2227`

Field 0, the response identifier.

```
nnnnnn
```

Field 1, the six random digits returned by the Network Security Processor. These digits are used in field one of command 1228 to confirm and complete the reset process. If you provide the incorrect value in the 1228 command, you will need to repeat the 1227 command again to generate a new nnnnnn value.

Table 12-55 Response 2227: Reset to Factory State

Field	Contents	Length (bytes)	Legal Characters
0	Command identifier	4	2227
1	Random digits	6	0 - 9, A - F

Examples

The following example illustrate using Command 1227 to reset the Network Security Processor to factory state.

The command looks like this:

```
<1227#RESET_TO_FACTORY_STATE#>
```

The Network Security Processor issues a response similar to this:

```
<2227#nnnnnn#>
```

The following example illustrates using Command 1227 to reset the Network Security Processor to factory state and clear the security audit log.

The command looks like this:

```
<1227#RESET_TO_FACTORY_STATE#CLEAR#>
```

The Network Security Processor issues a response similar to this:

```
<2227#nnnnnn#>
```

Confirm Reset to Factory State (Command 1228)

Command 1228 – Use this command to confirm the Network Security Processor's return to factory state. This command erases the user-defined security policy, and all user-defined keys. The default security policy is restored. You should use this command only when you need to add the Network Security Processor to a security association. You must first use command 1227 to generate the input for field 1.

note This command will only be processed if it is received from either the USB or serial port.

Command

```
<1228#nnnnnn#>
```

Response

```
<2228#Status#>[CRLF]
```

Calling Parameters

`1228`

Field 0, the command identifier.

`nnnnnn`

Field 1, the six random digits from Field 1 of Response 2227.

Table 12-56 Command 1228: Confirm Reset to Factory State

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	4	1228
1	nnnnnn	6	0 - 9, A - F

Responding Parameters

`2228`

Field 0, the response identifier.

`status`

Field 1, the result of processing the command.

Status	Description
ok	Confirmation that the Network Security Processor has reset to factory state.
Bad Confirmation	Indicates that the wrong value was entered. You must repeat the command 1227 again to obtain a new nnnnnn value.

Table 12-57 Response 2228: Confirm Reset to Factory State

Field	Contents	Length (bytes)	Legal Characters
0	Response identifier	4	2228
1	Status	2, 16	ok, Bad Confirmation

Example

The following example illustrate using Command 1228 to confirm the reset to factory state.

The command looks like this:

```
<1228#nnnnnn#>
```

The Network Security Processor returns the following response:

```
<2228#ok#>
```

13

Error Messages

Application Error Messages

If the Network Security Processor encounters a command syntax error, an error response message is returned. The format of the error response is:

<00#XXYYZZ#>

The response ID of 00 indicates an error is being returned.

Table 13-1 lists the error number and its description that is returned in field XX.

note When xx = 04 the next two digits (yy) indicate the total number of fields that were expected in the command.

Table 13-1 Error Types

Error	Description
00	Response to test message
01	Length out of range
02	Invalid character
03	Value out of range
04	Invalid number of parameters
05	Parity error
06	Key usage error
07	Key usage error
08	Execution error
09	Execurtion error

Table 13-1 Error Types (continued)

Error	Description
10	Key Length error
11	Printing error
12	Marker string not found
20	Serial number set, cannot modify it
21	The Network Security Processor not in a Security Association, or serial number is not present.
22*	Non-existent command or option
23*	Invalid command or option
24	Incorrect challenge
25	Incorrect Acknowledgement
26*	Duplicate command or option
27	No challenge to verify, a command 109 has been received without a prior command 108
28	The configuration string in command 108 is too long.
29	Unable to allocate memory for the configuration string.
41	ASRM timed out waiting for the response from the Network Security Processor.
73	Header mismatch
92	Autokey error
93	Factory keys already generated
94	No factory keys generated

* If this error is generated when processing security policy commands, the error response will include an additional field after the XXYYZZ field. This additional field will contains the first item found in error.

`YY` – the first field found to be in error.

note The field reported in the error response may not be the first, or only, field in the command that contains an error.

If this field returns the value 00, any of the following may be true:

— Your command specified an invalid command number.

— The MFK is missing.

— The response has been sent simply as an echo of a command.

`ZZ` – the software version level of the cryptographic command processor.

Detailed Errors

The detailed error is appended as a separate field after the error field (XXYYZZ). Detailed errors are only included if option 021 is enabled, see Configure Network Security Processor Option (Command 101) for more information on enabling detailed application errors. Table 13-2 lists the detailed application error messages by number, and provides the description of each message.

Table 13-2 Detailed Application Errors

Error	Description
1	Invalid command string length
2	Invalid command length
3	Invalid parameter length
4	Passcode length not matched with user data
5	Non empty field - conflicts with other fields
101	Invalid command string format
102	Invalid character
200	Value out of range
201	Invalid command
202	Invalid parameter value
209	Invalid key length specified
211	Invalid ANSI-formatted message authentication code
212	Invalid MAC
215	Invalid checksum on string
216	Value in field is not same as other field
217	Count value not greater than zero
218	Command count table is full
220	No free key slot for RSA key
230	Command does not exist
301	Too many fields
302	Too few response fields

Table 13-2 Detailed Application Errors (continued)

Error	Description
303	Too few fields
304	Initialization vector is missing
305	Wrong combination of keys
306	Invalid number of parameters
307	PAN does not match validation data
501	Table entry in use
502	Table full
509	Key did not have odd parity
510	Specified variant cannot be used
511	KD1 or KD2 check digits do not match expected check digits
512	Single length key not allowed
513	Command 14-5, weak key
514	Command 14-5, keys have different length
516	Session key check digits do not match expected check digits
517	Key check digits do not match expected check digits
518	MAC does not match
519	Cannot load key into key table
520	MFK already loaded
521	Not in factory state
601	Non-existent module key entry
602	Non-existent MFK
603	Non-existent KEK
604	Non-existent Pending MFK
605	Incorrect entry of double-length key location
607	Security violation
612	MFK name in command does not match the MFK name
613	Pending MFK name in command does not match the pending MFK name
623	Key location empty

Table 13-2 Detailed Application Errors (continued)

Error	Description
630	TR31 key block MAC error
631	TR31 clear key length error
632	Invalid TR31 header
702	Internal error
708	Internal error
713	Internal error
714	BSAFE error
715	DUKPT error
716	Random number generator error
717	Failed self-test
718	Command not allowed in PCI-HSM mode
801	Failed hardware function
802	Failed ACS function (general)
805	Failed ACS function (Response returned smaller than minimum)
806	Failed ACS function (Response length invalid)
807	Failed ACS function (Response ID incorrect)
808	Failed ACS function (Response ID invalid)
809	Failed ACS function (Command had NULL error)
810	Failed ACS function (Command had NULL first item)
811	Failed ACS function (Response had NULL item)
812	Failed ACS function (Response had NULL first item)
813	Failed ACS function (Command ID was modified)
814	Failed ACS function (Command has invalid item)
815	Failed ACS function (Command has invalid first item)
816	Failed ACS function (Response has invalid item)
817	Failed ACS function (Response has invalid first item)
818	Failed ACS function (Command contains too many fields)
819	Failed ACS function (Response contains too many fields)

Table 13-2 Detailed Application Errors (continued)

Error	Description
820	Failed ACS function (Command buffer format is invalid)
821	Failed ACS function (Response buffer format is invalid)
822	Failed ACS function (Monitor process already running)
823	Failed ACS function (Monitor process not running)
901	Expecting a single-length key and received a double length key
902	Expecting a double-length key and received a single length key
903	The double-length key is a replicated single-length key
1100	No continuation indexes are available
1101	Specified continuation index is empty
1102	Invalid print job length
1103	Unable to obtain a socket on the printer
1104	Unable to connect to printer
1105	Unable to send print job to printer, error returned from printer
1200	Marker string not found
2000	The serial number is already set, it cannot be modified
2100	The serial number is not loaded
2101	NSP is not in a security association
2200	Non-existent command item in the configuration string
2300	Invalid command item format
2301	Command 105 has not been received
2303	The command is a counted command
2304	Option E5 is not enabled
2400	The input HASH in command 109 does not match the stored hash
2500	The acknowledgment text is incorrect or missing
2600	Conflicting duplication of a configuration parameter
2700	Command 109 was received without a prior command 108
7300	The variant of the key in table incorrect
7301	The variant for a decimalization table is wrong

Table 13-2 Detailed Application Errors (continued)

Error	**Description**
The following errors will only be present in commands sent by the SCA.	
9201	RSA keys already exists
9202	Global key data is corrupted
9203	Cannot allocate memory with mymalloc
9205	Failed signature verification
9208	Failed certificate verification
9209	Cannot sign the certificate or bad signature
9210	CBC encryption/decryption failed
9211	No communication key
9212	No session key
9213	MAC computation or verification failed
9214	Invalid buffer data length
9215	Invalid data length inside the header
9219	RSA key generation failed
9220	Certificates do not match
9240	Error writing flash memory
9241	Error writing EEROM memory

Examples

Response 00 due to an Error Condition

The command contains an invalid value for the table location.

```
<72#5678#>
```

The Network Security Processor issues the following response.

```
<00#030114#>
```

This response indicates the following:

- The field’s value is out of range (indicated by 03).
- Field 1 is in error (indicated by 01).
- The Network Security Processor’s software version number is 1.4x.

If the detailed error feature (option 21) is enabled the response is:

```
<00#030114#0202#>
```

This response indicates the following:

- The detailed error (0202) indicates an invalid parameter value.

Response 00 indicating the command is not enabled

The command syntax is valid, however the command is not enabled in the Network Security Processors security policy

```
<11B#2#A30FFFDC12C5884D#1DDNN0I0,69AFDA80E06E2F4E30D6D1EDACCBB36E7B
EFAE872CCC14F7,9D6FCC8892846DC5#>
```

The Network Security Processor issues the following response.

```
<00#0300xx#>
```

where xx is the first two digits of the Network Security Processor's software version number.

A

Introduction to Cryptography and Cryptographic Keys

In 1973, the National Institute of Standards Technology (NIST) approved the use of an algorithm, the Data Encryption Algorithm (DEA), for providing data security in communications systems. DEA and a set of specifications, which require the algorithm to be implemented in high-speed hardware, is the technique accepted by federal standards. The algorithm, coupled with the
high-speed hardware, is known as the Data Encryption Standard (DES).

Data Encryption Standard (DES)

DES provides the data processing industry with a standard encryption technique that is acceptable in financial applications. The details of DES are well known, the strength of this process depends on the complexity of DES and the length of the secret key. Because of the successful exhaustive search attacks on single length DES keys, the financial industry is replacing single DES with triple DES.

The DES algorithm can operate in the eight basic modes that are described in FIPS PUB number 81. The commonly used modes of operation are briefly described:

Electronic Code Book (ECB)

In this mode, the DES unit uses a 56-bit key to encrypt or decrypt 64 bits of data and output ciphertext or plain text. This mode can be used to encrypt small data quantities 64 bits in length. For Triple DES, three 56-bit keys are used to encrypt and decrypt (see Figure A-1 on page A-3).

Cipher Block Chaining (CBC)

In this mode, the DES unit either encrypts or decrypts long strings of data blocks in multiples of eight bytes. The MACing commands 56, 98 and 99 and command 97 data encryption/ decryption use the CBC mode of DES. For Triple DES, three 56-bit keys are used to encrypt (see Figure A-2 on page A-4), and decrypt (see Figure A-3 on page A-4).

Message Authentication

DES can also be used to provide message integrity; it insures that the original message was received by the recipient, without being altered. See Federal Information Processing Standard 113, International Standards Organization 8731, American National Standards Institute X9.19, and ISO 9797 for more information.

Triple DES (3DES)

Triple DES utilizes three DES keys for encrypting/decrypting data. The combination of all three keys is referred to as a key component. This is referred to as a triple-length key or a 3key-3DES key.

Three DES cycles are used to either encrypt or decrypt information. When encrypting, the data is first encrypted using Key 1, the result is decrypted using Key 2, and this second result is encrypted using Key 3. The name 3DES indicates that there are three DES cycles used in the process (encrypt, decrypt, encrypt).

To decrypt data, the process is reversed (decrypt, encrypt, decrypt). When decrypting, the encrypted data is decrypted using Key 1, the result is encrypted using Key 2, and this second result is decrypted using Key 3.

Many systems use only 2 keys in a 3DES process. In this instance Key 1 is equal in value to Key 3, that is this value is used in the first and third step of the process. This is referred to as a double-length key or a 2key-3DES key.

It is also possible to use just one key in a 3DES process. In this instance Key 1 is equal in value to Key 2 and Key 3, that is this value is used in all three steps of the process. The result will be the same as using DES. This is referred to as a single-length key or a 1key-3DES key.

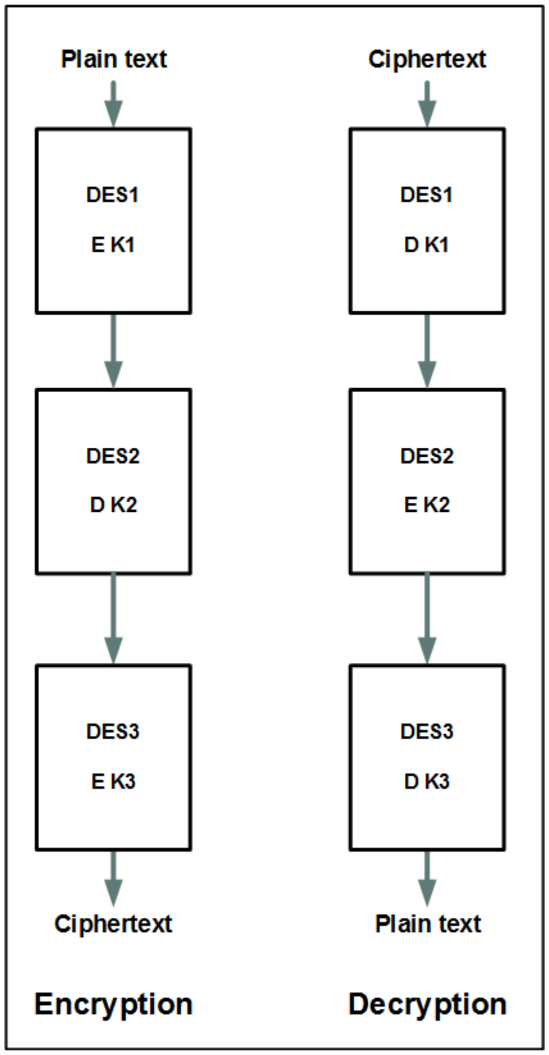


Figure A-1 TDEA Electronic Codebook

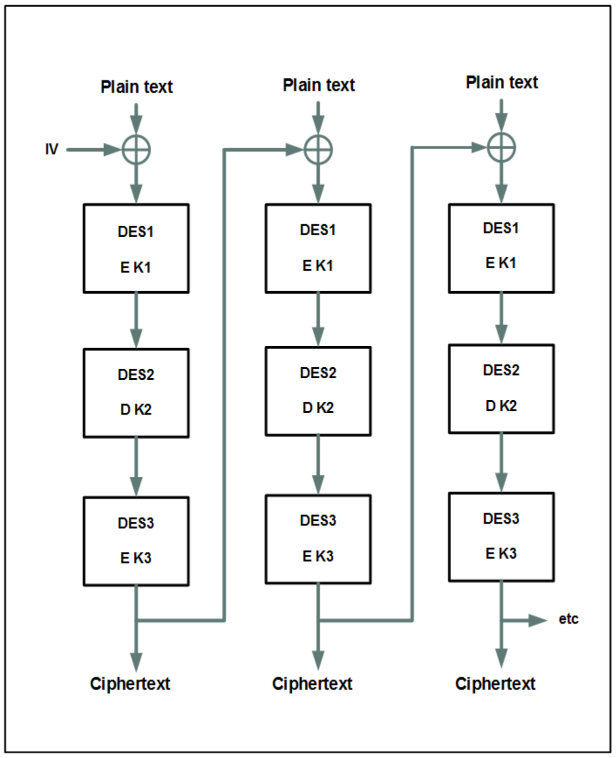


Figure A-2 TDEA Cipher Block Chaining - Encryption

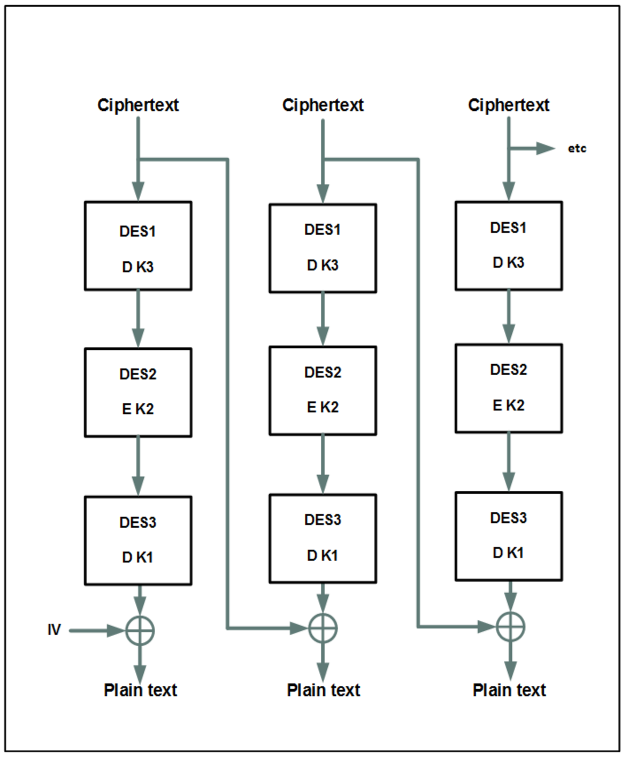


Figure A-3 TDEA Cipher Block Chaining - Decryption

Key Attributes

Key Length

The Network Security Processor supports DES keys that contain either 64 bits (single-length, 16 hexadecimal digits), 128 bits (double-length, 32 hexadecimal digits) or 192 bits (triple-length, 48 hexadecimal digits) of unique key material. DES keys use the eighth bit of each byte as a parity bit, this parity bit is not used in the encryption or decryption operation, therefore a single-length contains 56 bits, a double-length key contains 112 bits, and a triple-length key contains 168 bits, of unique keying material. See Key Parity for more information.

A triple-length key consists of three key blocks, each contain 16 hexadecimal characters, each block is a unique value, this is called a 3key-3DES key. A double-length key consists of two key blocks, each contain 16 hexadecimal characters, each block is a unique value, this is called a 2key-3DES key. A single-length key has only one key block which contains 16 hexadecimal characters, this is called a 1key-3DES key.

The Network Security Processor requires the Master File Key, and Pending Master File Key to be either a 2key-3DES key (double-length) or a 3key-3DES key (triple-length).

The Master File Key must be a 3key-3DES key (triple-length) if working keys will be 3key-3DES key (triple-length).

Key Components

In a financial network, secret keys are used to encrypt sensitive data, such as a customer's PIN, as it flows through the network. To prevent any one individual from possessing the secret key, the secret key value is divided into key components. Key components are maintained by trusted individuals for entry into the Network Security Processor. A multi-component key increases security because a different person is assigned to create each component. This way, nobody knows the value of the entire key, reducing the possibility of a security breach. Once all the key components have been entered, the Network Security Processor combines them into a final secret key value. This secret key value can then be used to either encrypt or decrypt information as it passes through the network.

Each 3DES key can have up to four key components. When you define a 3DES key, you are prompted for the number of key components. When all key components have been entered they are automatically combined into one secret key value. A 3key-3DES key has three key blocks for each key component. A 2key-3DES key has two key blocks for each key component. A 1key-3DES key has one key block for each component.

3key-3DES Key (Triple-Length)			
Component (Check Digits)	**Block 1**	**Block 2**	**Block 3**
Component 1 (35C1)	9205 48E6 FEB1 4A62	0BD1 45B6 6B72 A3BB	5865 2863 425A 38A9
Component 2 (53B3)	B8B9 7509 BBD6 4BEB	93C1 33F3 95A1 6819	5946 6D04 CBF1 F546
Final Key (B196)	2ABC 3DEF 4567 0189	9810 7645 FED3 CBA2	0123 4567 89AB CDEF

2key-3DES Key (Double-Length)		
Component (Check Digits)	**Block 1**	**Block 2**
Component 1 (2E0D)	C8F4 BD02 FD31 FFEE	674D 1508 5489 4275
Component 2 (3178)	E248 80ED B856 FE67	FF5D 634D AA5A 89D7
Final Key (057A)	2ABC 3DEF 4567 0189	9810 7645 FED3 CBA2

1key-3DES Key (Single-Length)	
Component (Check Digits)	**Block 1**
Component 1 (D5D4)	0123 4567 89AB CDEF
Component 2 (8422)	DDF2 C8B7 AA6C 4DBF
Final Key (DB8E)	DCD1 8DD0 23C7 8050

Check Digits

Given that the exact same secret key value must reside at both the sending entity and receiving entity, and that no one individual actually knows the secret key value, a method was required to allow both entities to be confident that they possess the correct secret key value. The Atalla method uses the 3DES algorithm to encrypt 16 zeros using the secret key value. The first (leftmost) four digits of the encrypted result are called check digits. Check digits are created for each key component, and also for the complete secret key value. If the check digits at both entities are the same, there is a high probability that they share the same secret key.

Key check digits are generated by encrypting zeros with the Key Block1, decrypting the result using the Key Block2, and encrypting this second result with the Key Block3. Other check digits techniques exist, Generate Check Digits (Command 7E) can be used to generate various types of check digits.

Key Parity

The leftmost 7 bits of a key byte are used in the DES algorithm, the rightmost bit, called a parity bit, is not. A single-length DES key, which contains 64 bits, only uses 56 bits. A double-length DES key, which contains 128 bits, only uses 112 bits. A triple-length key, which contains 192 bits, only uses 168 bits.

Most cryptographic systems require keys to be odd parity. This means that when a pair of hexadecimal characters (1 byte) is converted to binary format, the result contains an odd number of one bits. Here is an example of a key that is odd parity:

Clear-text key value: 01 23 45 67 89 AB CD EF, each byte contains an odd number of one bits.

Byte Value	**Binary Value**	**one bits**
01	0000 0001	1
23	0010 0011	3
45	0100 0101	3
67	0110 0111	5
89	1000 1001	3
AB	1010 1011	5
CD	1100 1101	5
EF	1110 1111	7

To convert an even hexadecimal byte to odd parity, adjust the least significant bit of the rightmost character of the byte. This adjusts the parity without changing the value of the key. Here is an example of adjusting a key with several even bytes to odd parity.

Key Value: 12 34 56 78 90 AB CD FF

Byte Value	Binary Value	Adjusted Binary Value	Adjusted Byte Value
12	0001 0010	0001 0011	13
34	0011 0100	odd parity	34
56	0101 0110	0101 0111	57
78	0111 1000	0111 1001	79
90	1001 0000	1001 0001	91
AB	1010 1011	odd parity	AB
CD	1100 1101	odd parity	CD
FF	1111 1111	1111 1110	FE

The parity adjusted key value is: 13 34 57 79 91 AB CD FE

Weak and Semi-weak DES Keys

Table A-1 contains a list of DES key values that are not secure. For example, a key value of all zeros cannot be used to securely encrypt. If a weak key value is entered at the SCA, a warning message is displayed. For production systems, avoid using weak key values. Note, an even parity key is not identified as a weak key.

Table A-1 Weak and Semi-weak Keys

Weak Keys	Semi-weak Keys	
0101 0101 0101 0101	E001 E001 F101 F101	01FE 01FE 01FE 01FE
FEFE FEFE FEFE FEFE	01E0 01E0 01F1 01F1	FE01 FE01 FE01 FE01
E0E0 E0E0 F1F1 F1F1	FE1F FE1F FE0E FE0E	011F 011F 010E 010E
1F1F 1F1F 0E0E 0E0E	1FFE 1FFE 0EFE 0EFE	1F01 1F01 0E01 0E01
	E01F E01F F10E F10E	E0FE E0FE F1FE F1FE
	1F0E 1F0E 0EF1 0EF1	FEE0 FEE0 FEF1 FEF1

HMAC Keys

HMAC keys have variable lengths, therefore the format of the key field portion of an AKB that contains an HMAC key will be different than that of a 3DES key.

Header	,	Key length	Key Value	Padding	,	MAC
(8 characters)		(2 bytes)	(variable length)	(variable length)		(16 characters)

The key field will consist of these values:

- A two-byte hexadecimal key length value which specifies the key length in bytes. For example, a 256-bit HMAC key will have a two-byte length value of 0x0020 and a 128-bit HMAC key will have a two-byte length value of 0x0010.
- Key value, in hexadecimal characters, where each hexadecimal character represents 4 bits of key material. The minimum length is 80-bits (20 hexadecimal characters) and the maximum length is 256-bits (64 hexadecimal characters). The length of the key value must be a multiple of 8.
- Padding characters, if necessary. The number of hexadecimal padding characters depends upon the length of the HMAC key. HMAC keys that are 176- or 240-bits are not padded, all other HMAC key lengths are padded such that the length of the key field is a multiple of 16.

The key field length varies based on the length of the HMAC key, therefore the length of an AKB that contains an HMAC key will be either 74, 90 or 106 characters as shown in the following table.

HMAC Key size in bits	**Padding characters**	**Total number of hexadecimal characters in the key field**	**AKB Length in characters**
80 through 176	24 through 0	48	74
184 - 240	14 through 0	64	90
248 - 256	14 through 12	80	106

If Network Security Processors that have the same Master File Key will be used to generate MAC values and also to verify them, you can use premium value command 39A to generate random HMAC keys in AKB format, and then use the AKBs in commands 39B and 39C.

If you will exchange HMAC keys with another external entity that uses the Atalla Key Block format, you can import or export HMAC keys that are 176-bits using commands 11 and 13. If you will export HMAC keys to an external entity that does not use the Atalla Key Block format, you can use command 1A to export HMAC keys that have a maximum key size of 176-bits, however the receiving entity must be able to support extracting the key value from the encrypted key data. To exchange HMAC keys that have more than 176-bits, you must use the SCA to manually generate key components, and then send each key component to a separate person associated with the entity that will be receiving the HMAC key. Use the following procedure to generate key component values.

1. Both the generating and receiving entity must agree on a key size. For this example, the key size will be 192 bits.
2. Divide the key size by 8, for example 192 / 8 = 24.

3. Convert result in step 2 to hexadecimal, for example 24 decimal is 18 in hexadecimal notation (0x18).

4. Prepend two zeros (00) to the step 3 result, for example 0018. This value is the hexadecimal length of the HMAC key in bytes.

5. Determine the number of hexadecimal characters that are required for the HMAC key, each hexadecimal character represents 4 bits of key material, therefore divide the step 1 result by 4, for example 192 / 4 = 48.

6. Add 4 (the number of characters in the step 4 result) to the step 5 result, for example 4 + 48 = 52.

7. Determine the number of padding characters. Using the table above, you can see that a 192-bit HMAC will be formatted in a 90 character AKB, where 64 characters are the key field. Therefore, you must subtract the step 6 result from 64 to obtain the number of padding characters to append to the key value, for example
64 - 52 = 12. A 192-bit HMAC key will be right-padded with 12 hexadecimal characters.

8. Construct key component 1, this example uses this key component value:
8F9DD02FA457CE6B0BC4C71A2692494380A7F43B0204948A

   a. Key length is 0018.

   b. Key value is
   8F9DD02FA457CE6B0BC4C71A2692494380A7F43B0204948A
   Provide this value to a person at the external entity.

   c. 12 random pad characters are C4B92302DAB5

   d. Concatenate a, b, and c values, this is key component 1:
   00188F9DD02FA457CE6B0BC4C71A2692494380A7F43B0204948AC4B92302DA
   B5

9. Construct key component 2, this example uses this key component value:
6BDC54C81C20CDA7523D7FC2834331160198BF3BA1B02CC4

   a. Key component 1 will be exclusive or'd with key component 2 therefore to preserve the key length value specified in key component 1, the key length value entered for key component 2 must be 0000.

   b. Key value is
   6BDC54C81C20CDA7523D7FC2834331160198BF3BA1B02CC4
   Provide this value to a different person at the external entity.

   c. 12 random pad characters are A9361EA433DD

   d. Concatenate a, b, and c values, this is key component 2:
   00006BDC54C81C20CDA7523D7FC2834331160198BF3BA1B02CC4A9361EA433D
   D

10. Use the SCA Calculate AKB function, which is documented in section 5 of the *Secure Configuration Assistant-3 User Guide*, to input the key components values in step 8d and 9d.

note You must purchase and enable option C1 in the Network Security Processor, this will configure the SCA-3 to support variable length key component entry.

The first security administrator starts the transaction, and then enters the first key component. The second security administrator approves the transaction data, and then enters key component 2. The SCA sends these key components to the Network Security Processor where they are combined to form the HMAC key, and then formatted into an AKB. The AKB returned at the end of this function, can be used in commands 39B and 39C.

a. Select **2** for the number of key components.

b. Select a type of **AKB Key** - **Variable-length key**.

c. Select a key component length. This value is computed by multiplying the result from step 6 by 4, for example 52 * 4 = 208. Tap **Next** to continue.

d. Define and input the AKB header value. The recommended header for a HMAC-SHA1 key is 1MHNE000. The recommended header for the HMAC-SHA256 key is 1M2HNE000. Refer to the syntax of commands 39B and 39C, for the list of allowed headers.

e. At the component entry screen, input the key component 1 value from step 8d, and then tap **Next**. Confirm that the key component is displayed correctly, and then record the check digits. Tap **OK** to continue.

f. The second security administrator inserts their smart card, and then enters their PIN. They tap **Select**, enter the key component 2 from step 9d, and then tap **Next**. They confirm that the key component is displayed correctly, and then record the check digits. They tap **OK** to continue.

g. The SCA will display the AKB and its check digits, record these value or save them in the SCA Data Table.

Sample Clear-text Key Component Form

****

Keep a copy of this completed form for your records, store securely, in a tamper evident envelope. Write the key name, key component number, date, and author on the exterior of the envelope.

Note: This individual must not have access to any other key component for this key.

Key Name ______________________________ Key Component Number ______

1. Enter the Key Block1:

   -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

2. Enter the Key Block2:

   (Enter a value in this field if the key component is for a 2key-3DES key or a 3key-3DES key)

   -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

3. Enter the Key Block3:

   (Enter a value in this field if the key component is for a 3key-3DES key)

   -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

4. Enter the Final key check digits:

   -- -- -- --

Name of Institution: ____________________________

Generated by: _______________________ Date: _____________

****

Sample Atalla Key Block Form

****

Keep a copy of this completed form for your records, store securely, in a tamper evident envelope. Write the key name, date, and author on the exterior of the envelope.

Key Name ______________________________ Key check digits: __ __ __ __

Enter the HEADER.

__ __ __ __ __ __ __ __

Enter the Encrypted Key Field.

__ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __

__ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __

__ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __

Enter the Message Authentication Code (MAC).

__ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __

This key is encrypted under the ______________________.

The check digits of the encrypting key are: __ __ __ __

Name of Institution: ____________________________

Generated by: _______________________ Date: _____________

****

B

Understanding Financial Interchange Networks

This section introduces financial interchange networks, the networks on which secure transactions travel, and outlines the tasks involved in initializing these networks.

Overview

Financial interchange networks are computer networks that facilitate on-line funds transfers. This type of network consists of three primary components: acquirer nodes, issuer nodes, and switches.

- Acquirer node – the computer is attached to automated teller machines (ATMs) or PIN pads that introduce transactions into the network.
- Issuer node – the computer that belongs to the financial institution that has an account relationship with the consumer. An issuer can have ATMs or PIN pads attached to it, enabling it to act as both an issuer and an acquirer.
- Switch node – the computer that directs transactions from multiple acquirers to the appropriate issuer. A switch can have ATMs or PIN pads attached to it.

Network security processors can reside with each of these nodes to ensure the security of data as it travels from point to point within the network.

Figure B-1 on page B-2 illustrates a simple financial interchange network.

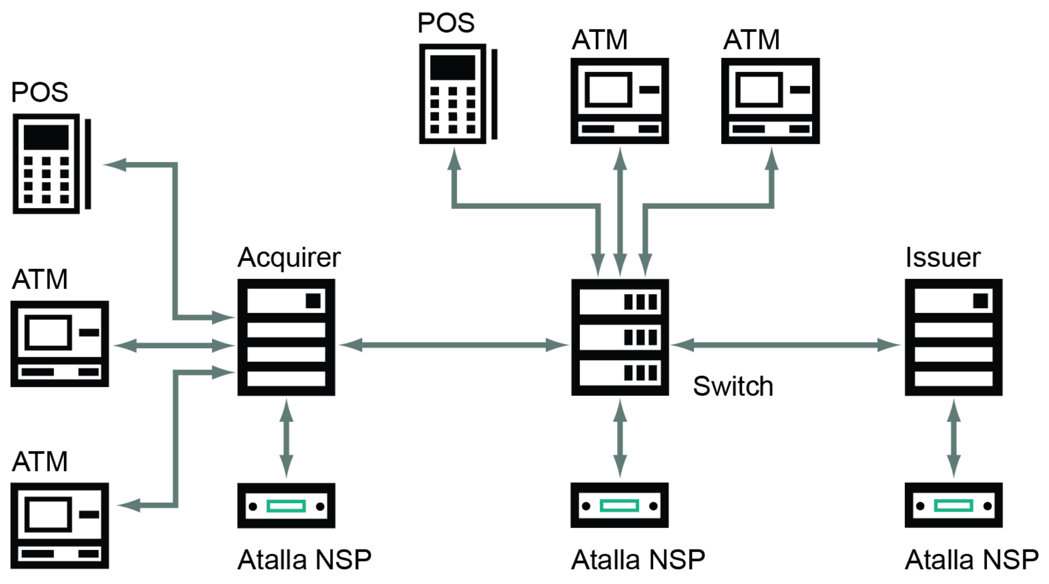


Figure B-1 Simple Financial Interchange Network

Initializing the Financial Interchange Network

This section explains the purpose of network initialization and describes typical network initialization tasks.

Purpose

Initialization refers to the process of establishing keys to be shared by a pair of participants on the financial network who agree to do business together. Establishing common keys facilitates transmitting data and decrypting encrypted messages.

For example, suppose that bank A decides to join switch XYZ, a switch that allows the customers of all participating financial institutions to transact business from any participant's ATM. In order for bank A customers to initiate transactions at, say, bank B, the switch must be able to translate the keys that bank B uses for encryption to keys that bank A recognizes and can decipher. To that extent, the switch must have access to both bank A's and bank B's Key Exchange and encryption keys. The switch may need to have access to other keys that each bank uses, too. The switch stores its members' Key Exchange Keys and encryption keys in its host database encrypted under its MFK.

Figure B-2 illustrates the key sharing necessary for bank A and bank B to do business on switch XYZ.

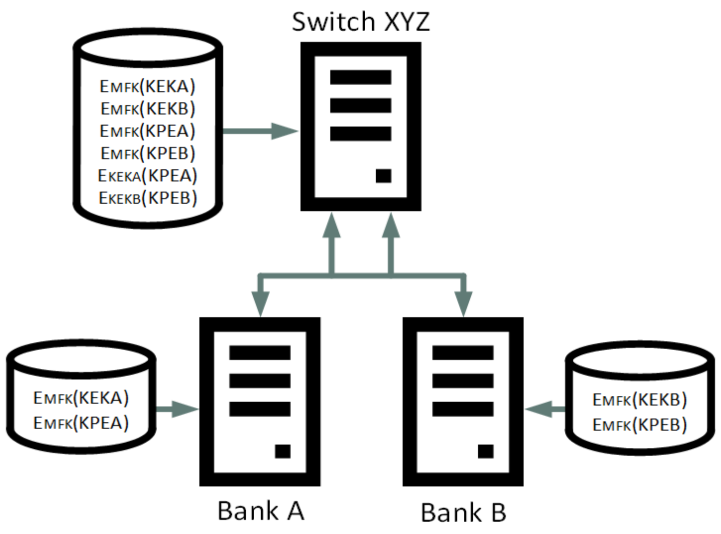


Figure B-2 Key Sharing

While this example may not resemble your network exactly, the idea of key sharing, as central to meaningful communication, is fundamental to any network that transmits secure data.

Initialization Checklist

The following list contains the tasks that are typically considered part of network initialization.

- Loading the Master File Key (MFK) into the security processor.
- Establishing the Key Exchange Key (KEK) that will be shared between nodes on the network.
- Establishing a PIN encryption key to be shared between ATMs or PIN pads and the host.
- Establishing a PIN encryption key to be shared between nodes.
- Establishing other working keys to be shared between nodes.

Establishing a PIN Encryption Key to be Shared by ATMs or PIN Pads and the Host

This subsection explains how to establish a PIN encryption key to be shared by ATMs or PIN pads and the host computer.

ATM-to-Host. The following list outlines the procedure for establishing a PIN encryption key to be shared by an ATM and a host computer.

1. Generate the ATM A key, the ATM B key, and the ATM master key.

2. Encrypt the ATM B key using the ATM A key.
3. Encrypt the ATM master key using the ATM B key.
4. Manually enter the ATM A key into the ATM.
5. Manually enter the cryptogram of the ATM B key into the ATM.
6. Program the ATM to decrypt the ATM B key using the ATM A key. When this process has finished, the clear text of the ATM B key will be in the ATM's memory.
7. Download the ATM master key from the host to the ATM, and then decrypt it using the ATM B key. Now the host and the ATM share the ATM master key. When a consumer enters his or her PIN at this ATM, the PIN block will be encrypted using the ATM master key, and then decrypted at the host using the same key.

PIN pad-to-host (using VISA DUKPT key management). VISA™ DUKPT key management specifies that the keys used to encrypt and transmit data be unique for every transaction. As such, the key used to initialize PIN-pad-to-host communication is used by the VISA DUKPT algorithm to derive unique transaction keys, but does not facilitate transactions on its own. The following list outlines the procedure for establishing a common PIN encryption key to be shared by a PIN pad and host computer.

1. Generate a derivation key.
2. Store the cryptogram of the derivation key in the host database.
3. Depending on the type of PIN pads being used on your network, you may want to maintain a database of each type's attributes. Alternatively, you can opt to receive this information as part of each transaction.
4. Load each PIN pad with an initial key serial number (IKSN).
5. Generate the initial PIN encryption key by encrypting the initial key serial number using the appropriate derivation key.
6. Load each PIN pad with an initial PIN encryption key.

Establishing a PIN Encryption Key to be Shared by Two Nodes

This subsection explains how to establish a PIN encryption key to be shared by two nodes. This procedure is the same for networks using VISA DUKPT key management and for those that do not.

1. The originating node generates a PIN encryption key using Command 10. The command returns two versions of the key: one to store locally, encrypted under the MFK and one to send to the other node, encrypted using the KEK that it shares with the receiving node. It then stores one cryptogram and sends the other cryptogram to the receiving node.

2. The receiving node stores the cryptogram on its host database.

Establishing Other Working Keys to be Shared by Nodes

The procedure for establishing any working key to be shared by two nodes is the same as the procedure for establishing a common PIN encryption key. Refer to Establishing a PIN Encryption Key to be Shared by Two Nodes above for instructions on establishing common working keys.

C

Summary of Commands and Options

Use this appendix to quickly locate information on a specific command or option. Also provided is a recommendation for security officers on how and when to enable security options. Table C-1 on page C-2 lists the Network Security Processor commands in numerical order.Table C-2 on page C-20 lists the Network Security Processor Options in numerical order. The Security Policy column in these tables contains one of the following three values:

- Standard - Commands and Options are ON in the Network Security Processor's default security policy. They can be disabled using either the SCA, or commands 108 and 109.
- Security Exposure - Commands are OFF in the Network Security Processor's default security policy. They can be enabled using either the SCA, or commands 108 and 109.
- Security Relevant - Options are OFF in the Network Security Processor's default security policy. The default option value provides backwards compatibility. It may not be the most secure value and should not be considered as a recommendation.
- Premium Value - Commands and Options are OFF in the Network Security Processor's default security policy. They must be purchased in the form of a command 105. After the command 105 has been sent to the Network Security Processor, the security policy must be updated to include these commands and options. This is accomplished using either the SCA, or commands 108 and 109.

note Some premium value commands and options, listed in the following tables, were developed for specific customers. These customer specific commands and options are not available for use by any other customer. They are listed in these tables only because they are included in the response to command <9A#CONFIG-ALL#> and <9A#CONFIG-OFF#>.

Network Security Processor Commands

Table C-1 Command Locator

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
00*	Echo Test Message	Tests the communications link between the host and the security processor.	On	No	12-4
10	Generate 3DES Working Key, Any Type	Generates a variety of working keys. The command returns the generated key in two forms: one for storing locally and one for transmitting to another node.	On	No	3-4
11	Export a Working Key in AKB format	Translates a working key from encryption using the Master File Key to encryption using the Key Exchange Key for transmitting to another network node.	Off	No	3-8
13	Import a Working Key in AKB format	Translates a working key from encryption using the Network Security Processor’s Key Exchange Key to encryption using the Master File Key.	Off	No	3-12
14	Load ATM Master Key – IBM 3624	Encrypts the ATM master key for downloading to IBM 3624 ATMs.	Off	No	3-15
14	Load ATM Master Key – IBM 4731	Encrypts the ATM master key for downloading to IBM 4731 ATMs.	Off	No	3-19
15	Change ATM Communications Key – Docutel	Encrypts a communications key for downloading to Docutel ATMs.	Off	No	3-23
15	Change ATM Communications Key – IBM 3624	Encrypts a communications key for downloading to an IBM 3624 ATM.	Off	No	3-26
15	Change ATM Communications Key – IBM 4731	Encrypts a communications key for downloading to an IBM 4731 ATM.	Off	No	3-30
1A	Export a Working Key in non-AKB format	Translates a working key from encryption using the Master File Key to a Key Exchange Key.	Off	No	3-33
1B	Derive Key	Customer Specific Command	Off	Yes	n/a
1C	Generate Session Key	Customer Specific Command	Off	Yes	n/a

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
1E	Generate New Initial Key for PIN Pad Using VISA UKPT	Supports an obsolete, proprietary technique for key loading and is not compatible with any current POS terminals. It is not required or recommended in any DUKPT environment.	Off	Yes	3-36
1F	Generate Token Key	Customer Specific Command	Off	Yes	n/a
30	Encrypt PIN	Encrypts a clear-text PIN.	Off	Yes	4-21
31	Translate PIN	Translates a PIN from encryption under one key to encryption under ANSI format.	On	No	4-23
31	Translate PIN – VISA DUKPT	Translates an encrypted VISA DUKPT PIN to ANSI format.	On	No	4-27
32	Verify PIN – Identikey	Decrypts an incoming PIN and verifies it using the Atalla Identikey method of PIN verification.	On	No	4-32
32	Verify PIN – Identikey/ Encrypted Bank ID	Decrypts an incoming PIN and verifies it using the Atalla Identikey method of PIN verification. Accepts an encrypted bank ID as input.	On	No	4-38
32	Verify PIN – IBM 3624	Decrypts an incoming PIN and verifies it using the IBM 3624 method of PIN verification.	On	No	4-38
32	Verify PIN – VISA	Verifies PINs using the VISA verification method of PIN verification.	On	No	4-43
32	Verify PIN – Atalla DES Bilevel	Decrypts an incoming PIN and verifies it using the Atalla DES Bilevel method of PIN verification.	On	No	4-48
32	Verify PIN – Diebold	Decrypts an incoming PIN and verifies it using the Diebold method of PIN verification.	On	No	4-53
32	Verify PIN – NCR	Decrypts an incoming PIN and verifies it using the NCR method of PIN verification.	On	No	4-58
32	Verify PIN – Clear-PIN Comparison	Decrypts an incoming PIN and verifies it using the clear-PIN comparison method of verification.	Off	Yes	4-64

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
32	Verify PIN – PIN-Block Comparison	Decrypts two incoming PIN blocks and compares their clear-text values.	Off (Option 61)	Yes	4-68
32	Verify PIN - Burroughs	Decrypts an incoming PIN and verifies it using the Burroughs method of PIN verification.	On	No	4-71
32	Verify PIN - Atalla 2x2	Decrypts an incoming ANSI PIN Block and verifies it using the Atalla 2x2 PIN Verification Method.	On	No	4-76
33	Translate PIN – ANSI to PLUS and PLUS to ANSI	Translates an incoming, encrypted ANSI PIN to PLUS format. Also translates an incoming, encrypted PLUS PIN to ANSI format.	Off	No	4-80
33	Translate PIN – ANSI to PIN/Pad	Translates an incoming, encrypted ANSI PIN to PIN/Pad format.	Off	No	4-84
33	Translate PIN – ANSI to IBM 4731	Translates an incoming encrypted ANSI PIN to IBM 4731 format.	Off	No	4-88
33	Translate PIN – IBM 3624 to IBM 3624	Translates an incoming, encrypted IBM 3624 PIN to IBM 3624 format.	Off	No	4-92
33	Translate PIN – IBM 3624 to PIN/Pad	Translates an incoming, encrypted IBM 3624 PIN to PIN/Pad format.	Off	No	4-96
33	Translate PIN – PIN/Pad or Docutel to PIN/Pad	Translates an incoming, encrypted PIN/Pad or Docutel PIN to PIN/ Pad format.	Off	No	4-100
33	Translate PIN – PIN/Pad or Docutel to IBM 4731	Translates an incoming, encrypted PIN/Pad or Docutel PIN to IBM 4731 format.	Off	No	4-104
33	Translate PIN – IBM 4731 to PIN/Pad	Translates an incoming, encrypted IBM 4731 to PIN/Pad format.	Off	No	4-108
33	Translate PIN – IBM 4731 to IBM 4731	Translates an incoming, encrypted IBM 4731 to IBM 4731 format.	Off	No	4-112
34	PIN Translate	Customer Specific Command	Off	Yes	n/a
35	Translate PIN, Double-Encrypted Input or Output	Decrypts and re-encrypts an encrypted PIN, where the input or output is double encrypted.	Off	No	4-117
36	Verify Double-Encrypted PIN	Decrypts an incoming double-encrypted PIN and verifies it according to the specified method.	Off	No	4-121

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
37	PIN Change – Identikey	Verifies the old PIN using the Atalla Identikey method.	Off	Yes	4-125
37	PIN Change – Identikey/ Encrypted Bank ID	Verifies the old PIN using the Atalla Identikey method.	Off	Yes	4-131
37	PIN Change – IBM 3624	Verifies the old PIN using the IBM 3624 method.	Off	Yes	4-131
37	PIN Change – VISA	Verifies the old PIN using the VISA method.	Off	Yes	4-137
37	PIN Change – Atalla DES BiLevel	Verifies the old PIN using the Atalla DES BiLevel method.	Off	Yes	4-142
37	PIN Change – Diebold	Verifies the old PIN using the Diebold method.	Off	Yes	4-148
37	PIN Change – NCR	Verifies the old PIN using the NCR method.	Off	Yes	4-153
38	PIN Change	Customer Specific Command	Off	Yes	n/a
39	PIN Translate and Generate MAC	PIN Translate and Generate MAC.	Off	No	4-159
3A	Verify Card and PIN - IBM3624	Verifies the card, and then verifies the PIN using the IBM3624 algorithm.	On	No	4-164
3A	Verify Card and PIN - VISA	Verifies the card, and then verifies the PIN using the VISA algorithm.	On	No	4-173
3A	Verify Card and PIN - Atalla Bilevel	Verifies the card, and then verifies the PIN using the Atalla Bilevel algorithm.	On	No	4-181
3A	Verify Card and PIN -NCR	Verifies the card, and then verifies the PIN using the NCR algorithm.	On	No	4-189
3D	Generate PVN and Offset	Generates and Identikey PVN and IBM 3624 Offset from an encrypted PIN.	Off	Yes	4-199
3F	PIN Verify	Customer Specific Command	Off	Yes	n/a
55	Encrypt, Decrypt, or Translate data	Encrypts, Decrypts, or Translate Data using ECB mode of DES.	Off	No	5-5
58	Translate MAC	Verifies a MAC and Generates a new MAC.	Off	No	6-5

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
59	Generate MAC and Encrypt or Translate Data	Generates a MAC and Encrypts or Translates Data.	Off	No	6-13
5C	Verify and Generate MAC for VISA UKPT	Verifies a Message Authentication Code and generates an approval or denial Message Authentication Code.	On	No	6-26
5D	Generate CVV/CVC	Generates a Card Verification Value/Card Validation Code.	Off	No	7-3
5E	Verify CVV/CVC	Verifies Card Verification Value/ Card Validation Code.	On	No	7-6
5F	Verify MAC and Decrypt PIN	Verifies a MAC and if successful decrypts a PIN.	Off	Yes	6-30
70	Load Volatile Table Value	Loads a value into the volatile table.	On	No	9-4
71	Delete Volatile Table Value	Deletes a value stored in a specific location.	On	No	9-7
72	Verify Volatile Table Value	Retrieves the check digits of a value stored in a specific location.	On	No	9-9
73	Clear Volatile Table	Clears a section of the volatile table.	On	No	9-12
74	Load Diebold Number Table Row	Load a row of the Diebold Number Table.	On	No	9-14
75	Enter Key Part	Encrypts a key under the MFK.	Off	Yes	n/a
76	Import KEK	Customer Specific Command	Off	Yes	n/a
77	Export KEK	Customer Specific Command	Off	Yes	n/a
78	Import Operation Key	Customer Specific Command	Off	Yes	n/a
79	Export Operation Key	Customer Specific Command	Off	Yes	n/a
7A	Generate Check Digits	Customer Specific Command	Off	Yes	n/a
7B	Verify Check Digits	Customer Specific Command	Off	Yes	n/a
7E	Generate Check Digit	Generates check digits in order to confirm that two parties hold the same key value.	On	No	3-40
7F	Load Value to a Specific Location	Loads a value into a specified location of the volatile table.	On	No	9-16

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
90	Decrypt PIN	Decrypts an incoming PIN block and returns the clear-text PIN.	Off	Yes	4-204
93	Generate Random Number	Generates a decimal or hexadecimal random number.	On	No	5-10
94	Generate IV	Generates an initialization vector.	Off	No	5-12
95	Reformat IV	Reformats an initialization vector.	Off	No	5-14
96	Verify IV	Verifies the format and content of an initialization vector.	Off	No	5-17
97	Data Encrypt, Decrypt	Encrypts or Decrypts data.	Off	No	5-20
98	Generate Message Authentication Code	Generates a Message Authentication Code.	Off	No	6-35
99	Verify Message Authentication Code	Verifies a Message Authentication Code.	On	No	6-40
9A*	Security Processor Configuration Status	Returns the security processor's command and option status.	On	No	12-8
9A*	Security Processor Count Status	Returns a list of commands that are being counted along with the counter value.	On	No	12-12
9A*	Security Processor DIAGTEST	Returns the result of the cryptographic test.	On	No	12-15
9A*	Security Processor Status Key	Returns the security processor's current key information.	On	No	12-27
9A*	Security Processor Status ID	Returns the security processor's current configuration and serial number.	On	No	12-19
9B	Verify Response MAC	Verifies a MAC from an ACR.	On	No	6-46
9C	Verify Response ACR	Verifies a MAC from an ACR	On	No	n/a
9E	Translate Working Key from Current MFK to Pending MFK	Translates working keys for use with the pending Master File Key.	On	No	3-44
9F	Replace the Current MFK with the Pending MFK	Replaces the current Master File Key with the pending one.	On	No	3-47
B1	Generate PIN	Customer Specific Command	Off	Yes	n/a
B2	Generate Token Response	Customer Specific Command	Off	Yes	n/a

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
B3	Verify Token Response	Customer Specific Command	Off	Yes	n/a
B4	Generate MD4 Hash	Customer Specific Command	Off	Yes	n/a
B5	Verify Signature	Customer Specific Command	Off	Yes	n/a
B6	One Way Encryption	Customer Specific Command	Off	Yes	n/a
B7	Generate and Encrypt PIN	Customer Specific Command	Off	Yes	n/a
BA	PIN Translate ANSI to PIN Pad and MAC Verify	Translates an ANSI PIN Block to PIN/Pad and Verifies a MAC.	Off	No	4-207
BB	PIN Translate ANSI to Plus an MAC Verify	Translates an ANSI PIN Block to Plus and Verifies a MAC.	Off	No	4-212
BD	PIN Translate and Generate MAC	Translates a PIN and Generates a MAC.	Off	No	4-217
BE	Verify VSVC S1 Signature	Used to validate the S1 signature and generate the S2 signature for VSVC cards.	Off	Yes	8-4
BF	Verify VSVC S3 Signature	Used to validate the S3 signature for VSVC cards.	Off	Yes	8-9
D0	Verify Clear PIN	Verifies a clear-text PIN according to the specified verification method.	Off	Yes	4-226
D1	Verify Password	Customer Specific Command	Off	Yes	n/a
D2	Modify Password	Customer Specific Command	Off	Yes	n/a
D3	Generate Initial Password Offset	Customer Specific Command	Off	Yes	n/a
D4	Verify Password	Customer Specific Command	Off	Yes	n/a
D5	Verify Signature	Customer Specific Command	Off	Yes	n/a
D6	Modify Signature	Customer Specific Command	Off	Yes	n/a
D7	Initial Signature	Customer Specific Command	Off	Yes	n/a
D8	Generate Hash	Customer Specific Command	Off	Yes	n/a
D9	Verify Hash Signature	Customer Specific Command	Off	Yes	n/a
DA	Verify MAC	Customer Specific Command	Off	Yes	n/a
DB	Convert Token Key	Customer Specific Command	Off	Yes	n/a

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
101*	Configure Security Processor Options	Enables and disables the security processor's options.	On	No	12-30
102	Command Monitoring	Counts the number of PIN, sanity, CVV/CVC/CSC, and MAC verification failures that have been processed. It can also count the number of times an enabled command has been processed.	Off	No	12-34
103*	Get CPU Utilization	Returns a percentage value which is the average CPU utilization for the Network Security Processor.	On	No	12-39
105	License Premium Value Commands and Options	Licenses Premium Value Commands and Options.	On	No	12-42
106	Define Temporary Serial Number	Allows the entry of a temporary serial number into a Network Security Processor.	On	No	12-44
107	Confirm Temporary Serial Number	Activates a temporary serial number in a Network Security Processor.	On	No	12-48
108*	Define Security Policy	Lets you enable and disable the Network Security Processor's commands and security related options.	On	No	12-50
109*	Confirm Security Policy	Implements the security policy defined in command 108.	On	No	12-57
110	Generate KSM Key	CSM command	Off	No	n/a
111	Process KSM Key	CSM command	Off	No	n/a
112	Generate CSM MAC Key	CSM command	Off	No	n/a
113	Translate Key	Decrypts a working key in AKB format that was encrypted under a KEK, and encrypts it using the same KEK and either the Electronic Code Book (ECB) or the Cipher Block Chaining (CBC) modes of DES.	On	No	3-50
114	Import Key	Customer Specific Command	Off	Yes	n/a
115	Generate Key	Customer Specific Command	Off	Yes	n/a

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
117	Import a TR-31 formatted key	Translates a TR-31 formatted Working Key that is encrypted under a Key Exchange Key to encryption under the Master File Key.	Off	No	3-54
118	Export a working key in TR-31 format	Translates a working key that is encrypted under the MFK to TR-31 format encrypted under a Key Exchange Key.	Off	No	3-60
119	Import a working key in TR-31 format	Translates a working key that is encrypted under the MFK to TR-31 format encrypted under the KBPK.	Off	No	3-68
11A	Export a working key in TR-31 format	Translates a working key that is encrypted under the MFK to TR-31 format encrypted under the KBPK.	Off	No	3-74
11B	Import Non-AKB formatted Working Keys	Translates a working key from encryption under a KEK, to encryption under the MFK. The working key is formatted in the AKB.	Off	No	3-81
11C	Convert Atalla DES Key to 3DES AKB Format	Converts working keys encrypted under a variant specific MFK, to encryption under the MFK. The working key is formatted in the AKB.	Off	No	3-85
11E	Generate Atalla 2x2 PVN	Generates a PIN Verification Number using the Atalla 2x2 method.	Off	Yes	4-229
11F	Derive Key	Customer Specific Command	Off	Yes	n/a
120	Generate RSA Key Pair	Generates a RSA public and private key pair that can be used for a variety of applications, such as Digital Envelope and Digital Signature.	Off	Yes	10-9
121	Load RSA Keys to the volatile table	Stores a RSA key in the volatile RSA key table.	Off	Yes	10-13
122	Generate AKB for Root Public Key	Generates the AKB for the root Public Key.	Off	Yes	10-16

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
123	Verity Public Key and Generate AKB PUB	Verifies the digital signature of a Public Key (or certificate) using the signing Public Key, and generates the AKB for the Public Key.	Off	Yes	10-20
124	Generate Digital Signature	Generates a digital signature of a given input using SHA1, or MD5 hash algorithms.	Off	Yes	10-28
125	Verify Digital Signature	Verifies the digital signature of a given input using MD5 or SHA1 hash algorithms.	Off	Yes	10-33
126	Generate Message Digest	Generates a digital hash of a given input using SHA1, or MD5 hash algorithms.	Off	Yes	10-38
127	Import Private Key	Customer Specific Command	Off	Yes	n/a
12A	Generate AKB for Root Public Key	Generates the AKB for the root Public Key.	Off	Yes	10-44
12B	Translate Public Key AKB from MFK to PMFK	Translates a RSA Key AKB from encrypted under MFK to PMFK.	Off	Yes	10-48
12C	Verify or Delete RSA Key Table Entry	Verifies or deletes a RSA key table entry.	Off	Yes	10-51
12D	Import Base Keys	Customer Specific Command	Off	Yes	n/a
12F	Generate ATM Master Key and encrypt with public key	Generates an ATM Master Key and encrypts it under a RSA public key.	Off	Yes	10-53
131	RSA Private Key Operation	Generates a signature on input data.	Off	Yes	n/a
132	RSA Public Key Operation	Verifies a signature.	Off	Yes	n/a
133	Export RSA Private Key	Exports an RSA Private key in CRT components.	Off	Yes	n/a
134	Import RSA Private Key	Imports an RSA Private key in PKCS #1 format.	Off	Yes	n/a
135	Export RSA Private key	Exports an RSA in ECB encrypted format.	Off	Yes	10-58
136	Generate TR-34 Key Block	Generates a random 3DES key and returns a signing token.	Off	Yes	10-62
138	RSA Data Decrypt	Decrypts data using an RSA private key	Off	Yes	5-27

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
139	Sign TR-34 Message	Signs a TR-34 message	Off	Yes	10-69
15E	Combine key components	Form a key from key components. Only allowed on the Print Command Port.	Off	No	11-11
161	Print PIN Letter	Prints a PIN mailer. Only allowed on the Print Command Port.	Off	No	11-17
162	PIN Issuance: IBM 3624	Generates an offset from a supplied PIN; generates a random PIN and corresponding offset; recover PIN from a supplied offset; Only allowed on the Print Command Port.	Off	No	11-24
163	PIN Issuance: Visa PVV	Generates a PVV from a supplied PIN; generates a random PIN and corresponding PVV. Only allowed on the Print Command Port.	Off	No	11-31
16A	Reset Printing Command Counter	Allows the host application to disable the printing commands.	Off	No	11-37
16E	Divide Key into components	Splits a key into key components. Only allowed on the Print Command Port.	Off	No	11-39
16F	Print Component Letter	Print a key component. Only allowed on the Print Command Port.	Off	No	11-44
301	Verify MAC	Customer Specific Command	Off	Yes	n/a
302	Generate MAC	Customer Specific Command	Off	Yes	n/a
304	Verify CMAC	Verifies a CMAC using the TDES algorithm.	On	No	6-52
305	Generate CMAC	Generates a CMAC using the TDES algorithm.	Off	No	6-55
306	Generate Cryptogram	Customer Specific Command	Off	Yes	n/a
307	Generate APRC	Customer Specific Command	Off	Yes	n/a
308	Generate MAC	Customer Specific Command	Off	Yes	n/a
309	Verify MAC	Customer Specific Command	Off	Yes	n/a
30A	Calculate PIN Offset	Generates a new Offset based on the old Offset.	Off	Yes	4-232

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
30B	Verify MAC	Customer Specific Command	Off	Yes	n/a
30C	Generate MAC	Customer Specific Command	Off	Yes	n/a
30D	Translate MAC	Customer Specific Command	Off	Yes	n/a
30E	Verify MAC	Customer Specific Command	Off	Yes	n/a
30F	Generate MAC	Customer Specific Command	Off	Yes	n/a
319	Generate Cryptogram	Customer Specific Command	Off	Yes	n/a
31A	Verify Check Value	Customer Specific Command	Off	Yes	n/a
31B	Decrypt Data	Customer Specific Command	Off	Yes	n/a
31C	Encrypt Password	Customer Specific Command	Off	Yes	n/a
31D	Verify Password	Customer Specific Command	Off	Yes	n/a
31E	Generate Key	Customer Specific Command	Off	Yes	n/a
31F	Verify Key	Customer Specific Command	Off	Yes	n/a
321	Verify PIN	Customer Specific Command	Off	Yes	n/a
322	Verify and Translate PIN	Customer Specific Command	Off	Yes	n/a
323	Verify PIN	Customer Specific Command	Off	Yes	n/a
324	Receive APACS 40 Request Message	Receive APACS 40 Request Message.	Off	Yes	6-58
325	Generate APACS 40 Response Message	Generate APACS 40 Response Message.	Off	Yes	6-63
326	Verify APACS 40 Confirmation Message	Verify APACS 40 Confirmation Message.	Off	Yes	6-68
328	Verify PIN	Customer Specific Command	Off	Yes	n/a
329	Verify PIN	Customer Specific Command	Off	Yes	n/a
32A	Verify PIN	Customer Specific Command	Off	Yes	n/a
32B	Import Key	Customer Specific Command	Off	Yes	n/a
32C	Verify ePIN Offset	Verifies an ePIN using an Offset.	Off	Yes	4-237
32D	Export Key	Customer Specific Command	Off	Yes	n/a
32E	PIN Masking	Customer Specific Command	Off	Yes	n/a

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
331	PIN Translate	Customer Specific Command	Off	Yes	n/a
332	PIN Translate	Customer Specific Command	Off	Yes	n/a
333	PIN Translate	Customer Specific Command	Off	Yes	n/a
334	PIN Translate	Customer Specific Command	Off	Yes	n/a
335	PIN Translate	Translates a PIN into a variety of PIN block types.	On	No	4-240
336	PIN Translate	Customer Specific Command	Off	Yes	n/a
337	PIN Translate	Customer Specific Command	Off	Yes	n/a
338	Export PIN	Customer Specific Command	Off	Yes	n/a
339	Generate PIN Offset	Customer Specific Command	Off	Yes	n/a
33A	PIN Translate	Customer Specific Command	Off	Yes	n/a
33B	Translate Response	Customer Specific Command	Off	Yes	n/a
33C	Generate Key	Customer Specific Command	Off	Yes	n/a
33D	Derive Key	Customer Specific Command	Off	Yes	n/a
33E	Data Decrypt and PIN Translate	Customer Specific Command	Off	Yes	n/a
33F	Data Encrypt and PIN Translate	Customer Specific Command	Off	Yes	n/a
340	Generate Session Key	Customer Specific Command	Off	Yes	n/a
341	Generate/Retrieve RSA Key	Customer Specific Command	Off	Yes	n/a
342	Import TMK in RSA envelope	Customer Specific Command	Off	Yes	n/a
343	Generate Terminal KPE	Customer Specific Command	Off	Yes	n/a
344	Generate KEK in RSA envelope	Customer Specific Command	Off	Yes	n/a
345	Export PVK	Customer Specific Command	Off	Yes	n/a
346	Translate DUKPT PIN to 3-DES and Verify MAC	Translates a DUKPT encrypted PIN to 3DES and verifies a MAC.	On	No	4-247
347	Translate DUKPT PIN to 3-DES and Generate MAC	Translates a DUKPT encrypted PIN to 3DES and generates a MAC.	Off	No	4-252
348	Verify MAC DUKPT	Verifies a DUKPT generated MAC.	On	No	6-71

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
349	Generate Terminal Master Key	Customer Specific Command	Off	Yes	n/a
34A	Calculate PIN	Customer Specific Command	Off	Yes	n/a
34C	Generate PIN	Customer Specific Command	Off	Yes	n/a
34D	Derive Key	Customer Specific Command	Off	Yes	n/a
34E	Derive Terminal Master Key	Customer Specific Command	Off	Yes	n/a
34F	Generate Terminal Personalization Key	Customer Specific Command	Off	Yes	n/a
350	Verify ARQC and return ARPC	Verifies an Authorization Request Cryptogram, and returns an Authorization Response Cryptogram, using either the VISA or Europay/Mastercard algorithms.	On	No	8-13
351	EMV PIN Change	Facilitates the functions required when performing EMV PIN Change without using the current PIN.	Off	Yes	8-23
352	Generate EMV MAC	Generates a MAC using either the VISA or Europay/Mastercard algorithms.	On	No	8-33
353	Generate EMV MACs	Customer Specific Command	Off	Yes	n/a
354	Generate EMV ICC Master Key	Generates an EMV ICC Master Key.	On	No	8-40
355	Verify EMV MAC	Customer Specific Command	Off	Yes	n/a
356	Validate CAP Token	Verifies an application cryptogram (AC) or signs transaction data.	On	No	8-44
357	Verify VISA dCVV	Verifies a VISA derived Card Verification Value.	On	No	7-9
358	Generate Digital Signature	Generates a signature for use in EMV smart cards.	Off	Yes	10-78
359	Verify dynamic MasterCard CVC3	Verifies a dynamic MasterCard CVC3 value.	On	No	7-13
35A	Verify AMEX CSC	Verifies an American Express Card Security Code.	On	No	7-18
35B	Generate AMEX CSC	Generates an American Express Card Security Code.	Off	No	7-22

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
35C	Cardholder Authentication Value	Customer Specific Command	Off	Yes	n/a
35D	Token-authenticated RSA Key Exchange	Customer Specific Command	Off	Yes	n/a
35F	Verify DCVV	Verify a Discover Dynamic Card Verification Value.	On	No	7-25
360	Generate Card Key	Customer Specific Command	Off	Yes	n/a
361	Generate Dynamic PAN	Customer Specific Command	Off	Yes	n/a
362	Translate Dynamic PAN	Customer Specific Command	Off	Yes	n/a
363	Generate DTC	Customer Specific Command	Off	Yes	n/a
364	Verify DTC	Customer Specific Command	Off	Yes	n/a
365	Verify Visa Cloud-Based Payments	Verifies either an MSD or qVSDC cryptogram.	On	No	8-50
36A	Verify AMEX Expresspay - Magstripe	Verifies an AMEX Expresspay value using the Magstripe mode.	On	No	7-29
36B	IVR Encryption	Customer Specific Command	Off	Yes	n/a
36C	IVR PIN Translation	Customer Specific Command	Off	Yes	n/a
370	Validate PIN	Customer Specific Command	Off	Yes	n/a
371	Change PIN	Customer Specific Command	Off	Yes	n/a
372	Translate Reference PIN Block	Customer Specific Command	Off	Yes	n/a
373	DUKPT to DUKPT PIN Translate	Customer Specific Command	Off	Yes	n/a
374	Generate PIN	Customer Specific Command	Off	Yes	n/a
375	Derive and Export Key	Customer Specific Command	Off	Yes	n/a
376	Export key under derived key	Customer Specific Command	Off	Yes	n/a
377	Generate ATM Master Key and encrypt using proprietary format	Customer Specific Command	Off	Yes	n/a
37A	Change PIN	Customer Specific Command	Off	Yes	n/a
37B	Generate ePIN Offset	Generates an ePIN Offset.	Off	Yes	4-258

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
381	Verify MAC	Customer Specific Command	Off	Yes	n/a
382	Generate MAC	Customer Specific Command	Off	Yes	n/a
383	PIN Translate	Customer Specific Command	Off	Yes	n/a
384	3DES-SHA1	Customer Specific Command	Off	Yes	n/a
385	3DES UKPT - Data Encrypt/ Decrypt	Customer Specific Command	Off	Yes	n/a
386	Generate MAC DUKPT	Generates a DUKPT MAC.	Off	No	6-75
387	PIN Verify	Customer Specific Command	Off	Yes	n/a
388	3DES DUKPT Encrypt/ Decrypt Data	Encrypts or Decrypts data	Off	No	5-31
389	TIK Generation	Customer Specific Command	Off	Yes	n/a
38A	3DES DUKPT - IFSF Encrypt/ Decrypt Data	Customer Specific Command	Off	Yes	n/a
38B	ZKA Master/Session - IFSF Encrypt/Decrypt Data	Customer Specific Command	Off	Yes	n/a
38C	Derive DUKPT Initial Key	Uses the BDK and KSN to generate the IPEK and returns it in AKB format.	Off	Yes	3-90
38D	Key Derivation	Customer Specific Command	Off	Yes	n/a
390	Encrypt/Decrypt Data	AES data encryption or decryption.	Off	Yes	5-36
391	Generate Key	Customer Specific Command	Off	Yes	n/a
392	Generate AES Check Digits	Generates AES key check digits.	Off	Yes	3-97
39A	Generate AES or HMAC Key	Generates an AES or HMAC key.	Off	Yes	3-93
39B	Generate MAC using HMAC	Generates a MAC.	Off	Yes	6-79
39C	Verify MAC using HMAC	Verifies a MAC.	Off	Yes	6-82
3A1	PIN Verify	Customer Specific Command	Off	Yes	n/a
3A2	Generate PRV	Customer Specific Command	Off	Yes	n/a
3A3	Verify PRV	Customer Specific Command	Off	Yes	n/a
3A4	PIN Translate	Customer Specific Command	Off	Yes	n/a
3B2	PIN Translate	Customer Specific Command	Off	Yes	n/a

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
3B3	PIN Translate	Customer Specific Command	Off	Yes	n/a
3B4	PIN Translate	Customer Specific Command	Off	Yes	n/a
3B5	PIN Translate	Customer Specific Command	Off	Yes	n/a
3B6	Open Registration Session	Customer Specific Command	Off	Yes	n/a
3B7	Open Banking Session	Customer Specific Command	Off	Yes	n/a
3B8	Verify Transaction	Customer Specific Command	Off	Yes	n/a
3EA	Derive Encrypted PIN	Customer Specific Command	Off	Yes	n/a
3F0	Derive Initial Key	Customer Specific Command	Off	Yes	n/a
3F1	Derive Key	Customer Specific Command	Off	Yes	n/a
3FA	Generate PIN and PVV	Customer Specific Command	Off	Yes	n/a
3FB	Generate Terminal Transaction Key	Customer Specific Command	Off	Yes	n/a
3FC	EMV PIN Change from PIN offset	Uses the supplied IBM3624 offset to determine the PIN. The PIN is then formatted and inserted into an EMV PIN change message.	Off	Yes	8-55
3FD	Authenticate multiple EMV scripts	Uses an EMV session key to generate message authenticate codes for up to 15 separate scripts.	Off	Yes	8-62
1101*	Get Image ID	Returns the image version information of the cryptographic command processor.	On	No	12-60
1104*	Get Temporary Serial Number Information	Returns the temporary serial number, if one is present.	On	No	12-62
1110*	Get System Configuration Information	Returns the NSP system software information, and cryptographic subsystem software version information.	On	No	12-64
1111*	Get Date and Time	Returns the NSP system date and time in Universal Coordinated Time.	On	No	12-66
1113*	Get Average CPU Utilization	Returns a percentage value which is the average CPU utilization for the Network Security Processor.	On	No	12-68

Table C-1 Command Locator (continued)

Command	Name	Purpose	Default Setting	Additional Cost (Premium Value)	Page
1120*	Get System Information	Returns the NSP serial number, product ID, system software information, and a personality version field.	On	No	12-70
1216*	Get Battery Life Remaining	Returns the number days of remaining battery life.	On	No	12-72
1221*	Return IP Address	Returns the IP Address of the Network Security Processor.	On	No	12-74
1223*	TCP/IP Socket Information	Returns information on the number of available sockets.	On	No	12-76
1226*	Get Application Key Check Digits	Returns the check digits of the MFK, and Pending MFK.	On	No	12-79
1227*	Reset to Factory State	Resets the Network Security Processor to factory state.	On	No	12-81
1228*	Confirm Reset to Factory State	Completes the Reset to Factory State procedure.	On	No	12-83
9109*	Return SuperKey Check Digits	Returns the SuperKey Check Digits.	On	No	n/a

* This command is not controlled by the security policy, it is always enabled.

Network Security Processor Options

Table C-2 Option Locator

Option	Name	Purpose	Default Setting	Additional Cost (Premium Value)
20	Append MFK Name to all responses.	Append the Master File Key name to all responses except the response of the status command, 9A; default – do not append name. Enable this option using either the SCA or command 101.	Off	No
21	Append detailed error information	Append the detailed error information to the error response, 00; default – do not append detailed error. Enable this option using either the SCA or command 101.	Off	No
23	Remove CR/LF from responses	Remove the carriage return and line feed from all responses; default – CR/LF appended to all responses. Enable this option using either the SCA or command 101.	Off	No
27	Use the rightmost 4 PIN digits for Diebold PIN verification. (Default uses leftmost 4 PIN digits)	Use the rightmost 4 PIN digits for Diebold PIN verification; default is to use the leftmost 4 PIN digits. Enable this option using either the SCA or command 101.	Off	No
28	Modify response string in commands 1101 and 1110.	Return "HP Atalla" in the responses to the 1101 and 1110 commands. The default action, when this option is disabled, is to return "HPE Atalla" in the responses to the 1101 and 1110 commands.	Off	No
42	Generate a 2key-3DES key in command 3FA	When enabled the Network Security Processor will generate a random 2key-3DES PIN Encryption Key in command 3FA.	Off	No
43	Use "R" for root public key headers	When enabled, command 123 requires that the header of the certificate authority's public key must have the letter "R" in byte 1 (Key Usage). If this option is not enabled, the key usage byte can contain either of these letters: "R" or "S".	Off	No

Table C-2 Option Locator (continued)

Option	Name	Purpose	Default Setting	Additional Cost (Premium Value)
44	Capture NSP Command and Error Response to System Log	When this option is enabled, any host application command sent to the NSP that results in an error will be logged to the system log along with the corresponding error response. This option is useful for troubleshooting host applications. Be sure to disable this option after the problem has been identified. Enable this option using either the SCA or command 101.	Off	No
46	Restrict PIN block types in PIN translate commands	See Option 46 - ANSI and ISO-3 PIN block	Off	No
47	Restrict outgoing PIN block types in PIN translate commands	See Option 47 - ANSI and ISO-3 Outgoing PIN block	Off	No
48	Require encrypted conversion tables	See Option 48 - Encrypted Conversion Tables	Off	No
49	Outgoing PIN Encryption Key (KPEo) length check	See Option 49 - Outgoing PIN Encryption Key length	Off	No
4A	PIN Verification Key length	See Option 4A - PIN Verification Key length	Off	No
4B	Prevent ANSI X9.8 PIN block attack	See Option 4B - Modified PIN Sanity Test	Off	No
4C	PIN validation digits match ANSI PAN digits	See Option 4C - Validation Data equals ANSI PIN block data	Off	No
4D	CVV/CVCs length check	See Option 4D - CVV/CVC length	Off	No
4E	Conversion Table Restriction	See Option 4E - Conversion Table restrictions	Off	No
4F	Check Digit Methods in command 7E	If this option is disabled method R is disabled and method I is enabled. If this option is enabled method R is enabled and method I is disabled.	Off	No

Table C-2 Option Locator (continued)

Option	Name	Purpose	Default Setting	Additional Cost (Premium Value)
60	Clear PIN Compare	Controls the ability to verify clear PINs. Refer to Verify PIN – Clear-PIN Comparison (Command 32). You must purchase this option in the form of a command 105, and enable it in your security policy using either the SCA directly, or commands 108 and 109 and the SCA. If this option is enabled, the clear PIN Compare command will be enabled.	Off	Yes
61	Encrypted PIN Compare	Controls the ability to verify encrypted PINs by comparison. Refer to Verify PIN – PIN-Block Comparison (Command 32).	Off	Yes
62	Allow command 31 DUKPT	Controls the ability to translate PINs that have been encrypted using the VISA DUKPT PIN encryption key. Refer to Translate PIN – VISA DUKPT (Command 31).	On	No
63	Allow command 32 DUKPT	Controls the ability to verify PINs that have been encrypted using the VISA DUKPT PIN encryption key. Refer to Verify PIN – Atalla 2x2 (Command 32).	On	No
66	Do not validate old PIN for command 37	Controls the ability to validate the old PIN in command 37.	Off	Yes
68	Allow Command 32 option E	Controls the ability to verify EBCDIC PINs.	Off	Yes
69	Allow Command 37A to not verify the old PIN.	This is an option for a custom command.	Off	Yes
6B	Requires the Incoming PAN/ICV to match the outgoing PAN/ICV and be non-zero	Requires the Incoming PAN/ICV to match the outgoing PAN/ICV and be non-zero. This option is only used in PIN and PIN-Block Translate (Command 335).	Off	No
6C	Allow 1key-3DES (single-length) working keys	Allows certain commands to accept a 1key-3DES (single-length) working keys. The length of PIN Verification keys in the IBM 3624, Atalla DES-Bilevel, NCR, and Atalla 2x2 PIN algorithms is controlled by option 4A.	Off	No
6E	Disable sequence number validation for command 109	Controls the ability to validate the sequence number when accepting a new security policy.	Off	No

Table C-2 Option Locator (continued)

Option	Name	Purpose	Default Setting	Additional Cost (Premium Value)
6F	Disable serial number validation for command 109	Controls the ability to validate the serial number when accepting a new security policy.	Off	No
80	Command 3F, option 1-6	This is an option for a custom command.	Off	Yes
81	Command 3F, option 7	This is an option for a custom command.	Off	Yes
82	Command 3F, option 8 and 9	This is an option for a custom command.	Off	Yes
83	Command 3F, option SG, SA, and SC	This is an option for a custom command.	Off	Yes
84	Command 32A, option B	This is an option for a custom command.	Off	Yes
87	Enable NIC2	When this option is enabled the NSP will enable NIC2 per the keyword/value pairs present in the config.prm file. This option must be enabled to use the printing commands 161 or 16F.	Off	Yes
88	Return 6 check digits	If this option is enabled the NSP will return six instead of four check digits in certain commands.	Off	No
89	Controls Command 32#E	This is an option for a custom command.	Off	Yes
8A	Enable commands 32#G and 32#H	This is an option for custom commands.	Off	Yes
8B	Enable command 14#5#	This option is used to enable command 14#5#.	Off	Yes
8C	Allow Single-Length VISA KPVs	When this option is enabled commands that process VISA PVVs will allow the KPV to be a 1key-3DES (single-length) key if the KPV uses any of these headers 1VVNE000, 1VVNN000, 1VVGE000, 1VVGN000, 1VVVE000, or 1VVVN000.	Off	Yes
8E	Allow 3DES to DES DUKPT PIN translation	This is an option for command 373 a customer specific command.	Off	No

Table C-2 Option Locator (continued)

Option	Name	Purpose	Default Setting	Additional Cost (Premium Value)
8F	Reduce the minimum allowed RSA key length to 768 bits	When this option is enabled the minimum allowed RSA key length is 768 bits. When this option is disabled (default value) the minimum RSA key length is 1024 bits. This option applies to the following commands: 120, 121, 122, 124, 125, 127, 12B, 12C, 12F, 138, and 358.	Off	Yes
A0	Sets minimum PIN length	The minimum PIN length can be from 0 to 12. The default is 4. Minimum PIN lengths of 10, 11, or 12 are defined as A, B, and C, respectively. For example: To change the minimum PIN length to 10, set (A0)="A". To change the minimum PIN length to 6, set (A0)="6". This option replaces option 40 that was set using command 101.	4	No
A1	Defines the Sanity indicator	Controls the value of the sanity indicator that will be returned if the sanity error is caused by an invalid PIN length. The default is "S". To return an "L" in this situation, set (A1)="L". To return this value to the default "S", set (A1)="S". This option replaces option 24 that was set using command 101.	S	No

Table C-2 Option Locator (continued)

Option	**Name**	**Purpose**	**Default Setting**	**Additional Cost (Premium Value)**
A2	DUKPT Session Key length	This option is used to control the length of the generated session key in commands 1E, 31, 32, and 5C, 346, 347, and 348. And custom commands 308 and 309. The default value is "S" for a 1key-3DES (single-length). To generate sessions keys that are 2key-3DES (double-length), set this option to "D". To allow the host application to specify the length of the session key to be generated in the command, set this option to "B". The length of the Base Derivation Key must be greater than or equal to the length of the session key.	S	No
C1	SCA Screen Display	This option is used to control what screens are displayed on the SCA-3.	Off	Yes
E0	Defines the type of working keys that can be imported	Defines the key types that can be imported in commands 13, 117, 119, and 11B. The values defined for this option must match that of Byte 1, Key Usage of the working key to be imported.	Off	No
E1	Defines the type of working keys that can be exported	Defines the key types that can be exported in commands 11, 1A, 113, 118, and 11A. The values defined for this option must match that of Byte 1, Key Usage of the working key to be exported.	Off	No
E2	Allows a KEK to be used for both import and export	Enable this option when the same KEK value is used to import and export working keys. If this option is enabled, the KEK header byte 3 can contain the value "N".	Off	No
E3	Allow ATM Master Key Header as KEK	Enable this option when commands 10 and 1A must support ATM Master Key headers.	Off	No

Table C-2 Option Locator (continued)

Option	Name	Purpose	Default Setting	Additional Cost (Premium Value)
E4	Allows command 11B to import customer specific base derivation keys	This is an option for a custom command.	Off	Yes
E5	Enables the printing commands	When enabled, this option allows the Network Security Processor to process printing commands for an 8 hour period which starts when this option is enabled in the Network Security Processor's security policy. This option is automatically disabled when the Network Security Processor is powered on.	Off	Yes

Recommended settings for security options

These options give security officers the ability to restrict the input data supplied in PIN processing commands, as well as in the Verify Card Verification Value or Card Validation Code command. The default value may not be the most secure choice. Carefully review each of these options and then decide which ones should be enabled in the Network Security Processor's security policy.

Option 46 - ANSI and ISO-3 PIN block

This option is used to restrict PIN block types in the following PIN translate commands: 31, 33#11, 33#13, 33#19, 33#22, 33#23, 33#33, 33#39, 33#93, 33#99, 35, 39, BA, BB, BD and 335.

The default setting for this option is disabled (OFF), which means that PIN translate commands will allow all PIN block types supported by that command.

When this option is enabled (ON), the Network Security Processor enforces these two requirements:

- Only ANSI (also referred to as ISO-0) or ISO-3 PIN blocks are allowed in the PIN translate command. This requirement is enforced for both the incoming and outgoing PIN block.
- In PIN translate commands 33#11, 35#...#...#11#, BB and 335 which contain both an incoming and an outgoing ANSI or ISO-3 Primary Account Number (PAN) field, both the incoming and outgoing ANSI or ISO-3 PAN values must be identical.

Recommendation: Review your PIN processing environment to determine what PIN translate commands are in use; disable all unnecessary PIN translate commands. Check with your processing partners to determine what types of PIN blocks should be allowed, and if the incoming ANSI or ISO-3 PAN data should be different than the outgoing ANSI or ISO-3 PAN data. Enable this option if only ANSI or ISO-3 PIN blocks should be allowed, and there is no legitimate business reason to support different values for the incoming and outgoing ANSI or ISO-3 PAN data.

Option 47 - ANSI and ISO-3 Outgoing PIN block

This option is used to restrict the types of outgoing PIN blocks in the following PIN translate command: 33#11, 33#13, 33#19, 33#22, 33#23, 33#33, 33#39, 33#93, 33#99, 35, BA, BB and 335.

The default setting for this option is disabled (OFF), which means that PIN translate commands will allow all outgoing PIN block types supported by that command.

When this option is enabled (ON), the Network Security Processor enforces these two requirements:

- Only ANSI (also referred to as ISO-0) or ISO-3 outgoing PIN blocks are allowed in PIN translate commands. All incoming PIN block types supported by the PIN translate command are allowed.
- In PIN translate commands 33#11, 35#...#...#11#, BB and 335 which contain both an incoming and an outgoing ANSI or ISO-3 Primary Account Number (PAN) field, both the incoming and outgoing ANSI or ISO-3 PAN values must be identical.

note When option 46 is enabled it supersedes this option.

Recommendation: Review your PIN processing environment to determine what PIN translate commands are in use; disable all unnecessary PIN translate commands. Check with your processing partners to determine what types of PIN blocks should be allowed, and if the incoming ANSI or ISO-3 PAN data should be different than the outgoing ANSI or ISO-3 PAN data. Enable this option if only ANSI or ISO-3 outgoing PIN blocks should be allowed, and there is no legitimate business reason to support different values for the incoming and outgoing ANSI or ISO-3 PAN data.

Option 48 - Encrypted Conversion Tables

This option affects PIN verification and PIN change commands that support conversion tables. The commands affected by this option are: 32#2, 32#6, 36#...#2#, 36#...#6#, 37#2, 37#6, 3A#2,x#, 3A#6,x#, 3D, D0#2 and 30A.

The default setting for this option is disabled (OFF), which means that only clear-text conversion tables or volatile table locations that contain encrypted conversion tables are supported in PIN verification and PIN change commands.

When this option is enabled (ON), the Network Security Processor enforces either of these two requirements:

- Conversion tables must be supplied in AKB format using a header value of 1nCNE000 or 1nCNN000.
- The volatile table location that contains the conversion table must be provided in the command. The conversion table must be loaded into the volatile table using command 70 or 7F.

note Conversion Tables contain 16 digits and can be mistaken for a DES key. Conversion tables are not DES keys and therefore are not controlled by option 6C.

Recommendation: Review your PIN processing environment to determine if conversion tables are in use. Confirm that your host application can support encrypted conversion tables, and if so, enable this option. If the host application cannot support encrypted conversion tables consider enabling option 4E.

Option 49 - Outgoing PIN Encryption Key length

This option is used to restrict the length of the outgoing PIN encryption key in the following PIN translate commands: 31, 33#11, 33#13, 33#19, 33#22, 33#23, 33#33, 33#39, 33#93, 33#99, 35, 39, BA, BB, BD, and 335.

The default setting for this option is disabled (OFF), which means that the length of the outgoing PIN Encryption Key (KPEo) is only restricted by the option 6C.

When this option is enabled (ON), the length of the KPEo must be equal to, or greater than, the length of the incoming PIN Encryption Key (KPEi). This option does not restrict the length of the KPEi.

note 1key-3DES (single-length) PIN encryption keys are allowed only when option 6C is enabled.

Recommendation: Review your PIN processing environment to determine what PIN translate commands are in use; disable all unnecessary PIN translate commands. Enable this option if all your processing partners require PIN blocks encrypted under 2key-3DES (double-length) or 3key-3DES (triple-length) keys.

Option 4A - PIN Verification Key length

This option affects PIN verification and PIN change commands that support IBM 3624, Atalla DES-Bilevel, NCR, and Atalla 2x2 PIN verification keys. The commands affected by this option are: 32#2, 32#4, 32#6, 32#I, 36, 37#2, 37#4, 37#6, 3A#2,x#, 3A#6,x#, 3D, 11E and 30A.

When this option is disabled (OFF), 1key-3DES (single-length) KPVs are not allowed.

When this option is enabled (ON), 1key-3DES (single-length) KPVs are allowed, regardless of the setting of option 6C.

note Conversion Tables contain 16 digits and can be mistaken for a key, these tables are not keys and therefore are not controlled by option 4A or 6C.

This option is not supported in the Variant Network Security Processor personality

Recommendation: Review your PIN processing environment to determine what key lengths are required for PIN verification and PIN change commands. Enable this option if 1key-3DES (single-length) KPVs should be allowed. If, after careful review, you determine that option 6C was previously enabled only to allow 1key-3DES (single-length) KPVs, you can enable option 4A and disable option 6C.

Option 4B - Modified PIN Sanity Test

This option affects PIN change, PIN translate, and PIN verification commands that support ANSI (also referred to as ISO-0) and ISO-3 PIN blocks. The commands affected by this option are: 31, 32#1, 32#2, 32# 3, 32#4, 32#5,32#6, 32#7, 32#F, 32#I 33#11, 33#13, 33#19, 35, 36, all forms of 37, 39, all forms of 3A, 3D, BA, BB, BD, 335, 346 and 347.

The default setting for this option is disabled (OFF), which means that all 16 hexadecimal characters of the decrypted ANSI or ISO-3 PIN block will be evaluated for sanity. And in the case of a PIN translate command, if the decrypted PIN block fails the sanity test, the value of the decrypted PIN block will be encrypted under the outgoing PIN Encryption Key (KPEo) and returned in the response.

When this option is enabled (ON), a modified PIN sanity test, which checks a subset of the decrypted PIN block, is performed. And in the case of a PIN translate command, if the decrypted PIN block fails the modified sanity test, 16 zeros will be returned in the response. This modified PIN sanity test block does not reveal information about the PIN when the Primary Account Number (PAN) digits, used to form the ANSI or ISO-3 PIN block, are manipulated.

Since all of the decrypted PIN block digits are not checked for sanity, there is the potential that in certain rare key synchronization conditions a PIN will not verify, or in the case of a PIN translate command an incorrect encrypted PIN will be returned in the response.

Recommendation: Enable this option so no useful information about a PIN is returned when incorrect primary account numbers are sent to the Network Security Processor in a PIN change, PIN translate, or PIN verification command.

Option 4C - Validation Data equals ANSI PIN block data

This option affects IBM3624, NCR, and Visa PIN change and PIN verification commands. Customer specific account digits are used in the PIN validation process. These same digits may, or may not, be used to form the ANSI PIN block. The commands affected by this option are: 32#2, 32#3, 32#6, 36#2, 36#3, 36#6, 37#2, 37#3, 37#6, all forms of 3A, and 3D.

The default setting for this option is disabled (OFF), which means that no comparison of the PIN validation data and ANSI PAN digits is performed.

When this option is enabled (ON), the IBM3624 and NCR validation digits must match the 12 digits used to form the ANSI PIN block. If there are less than 12 validation data digits an error response will be returned. When there are more than 12 validation digits the Network Security Processor will compare the rightmost 12 digits of the validation data to the ANSI PAN digits. If these digits do not match, the rightmost digit of the validation data is dropped and the comparison is performed again. If both comparisons fail an error is returned. In a Visa PIN change or PIN verification command the Network Security Processor compares the 11 digits of the verification data to the rightmost 11 ANSI PAN digits. If that test fails, it compares the 10 rightmost ANSI PAN digits with the 10 leftmost digits of the VISA verification data. If both comparisons fail an error is returned.

Recommendation: Review your PIN processing environment to determine what validation data lengths are supported. Enable this option if the validation data is 12 (11 digits for VISA) or more digits, and the validation data is equal to the digits used to form the ANSI PIN block.

Option 4D - CVV/CVC length

This option affects command 5E and all forms of command 3A.

The default value for this option is OFF, which means that the Visa Card Verification Value or the MasterCard Card Validation Code to be verified can be 1 - 8 digits in length.

When this option is enabled (ON), the Visa Card Verification Value or the MasterCard Card Validation Code to be verified must be 3 - 8 digits in length.

note In command 5E, this option applies only when field one (algorithm identifier) is set to 3.

Recommendation: Review your processing environment to determine the lengths of the Visa Card Verification Value or the MasterCard Card Validation Code to be verified. Enable this option if all the Visa Card Verification Value or the MasterCard Card Validation Code to be verified are at least three digits in length.

Option 4E - Conversion Table restrictions

This option affects PIN change and PIN verification commands that support either the IBM3624 or NCR PIN algorithms. The commands affected by this option are: 32#2, 32#6, 36#...#2#, 36#...#6#, 37#2, 37#6, 3A#2,x#, 3A#6,x#, 3D, D0#2 and 30A.

The default setting for this option is disabled (OFF), which means that any clear-text conversion table is allowed.

When this option is enabled (ON), two restrictions are placed on the conversion table, they are:

- The numeric conversion table must have at least eight unique digits.
- No single digit can occur more than four times.

Here are some examples of conversion tables that adhere to these restrictions:

0123456789012345, 9876543210543321, 8351296477461538

Here are some examples of conversion tables that do not adhere to these restrictions:

- 1234567123456712, does not contain eight unique digits

- 2437528797671271, the number 7 appears more than 4 times.

note When both options 48 and 4E are enabled, the conversion table restrictions are applied to the decrypted conversion table.

Recommendation: Review your PIN processing environment to determine what, if any, conversion tables are used. Enable this option if all conversion tables meet the requirements mentioned above.

D

Obtaining technical support

HPE Security - Data Security technical support

For technical questions, contact HPE Security - Data Security Technical Support:

- E-mail: DataSecurity.Atalla.Support@hpe.com
- Telephone: **+1-800-500-7858 (U.S.)** or **+1-916-414-0216 (outside of U.S.)**

Before contacting HPE, collect the following information:

- Product model names and numbers
- Technical support registration number or NonStop system number (if applicable)
- Service Agreement ID number (SAID)
- Product serial numbers
- Error messages
- Software version number
- Detailed questions

24-hour support

24-hour emergency support is available to those customers who have valid service contracts. Use this service for product and system emergencies that occur after normal working hours or on weekends and U.S. holidays. Questions about product installation and setup are supported during normal working hours.

For 24-hour emergency support call: **+1-800-255-5010**

Customers located outside the U.S. can obtain local support contact information from the Country phone numbers section of this document:

http://h20195.www2.hp.com/V2/GetPDF.aspx/c02083951.pdf

G

Glossary

1Key-3DES Key	A DES key that has the same value for all three key blocks. Also referred to as a single-length DES key
2Key-3DES Key	A DES key that has the same value for the first and third key blocks. Also referred to as a double-length DES key.
3DES	A term used to indicate that three DES cycles are used to either encrypt or decrypt information, also referred to as triple DES.
3Key-3DES Key	A DES key that has unique values for each of the three key blocks. Also referred to as a triple-length DES key.
AATMKEY	This working key is used to encrypt the ATM B key before the B key's cryptogram is loaded into an ATM machine.
Acquirer Node	The computer that has attached to it, automatic teller machines or PIN pads that introduce transactions into the network.
ANSI	American National Standards Institute.
ATM	Automated Teller Machine.
Authenticate	To establish the validity of a claimed identify.
BATM	ATM B-Key. This working key is used to encrypt the ATM master key before it is transmitted to an ATM machine.
Check Digit	An ending digit that is derived from the preceding digits in a number using an algorithm. Usually appended to the Primary Account Number (PAN).
Check Digits	A four to six hexadecimal character value used to ensure both entities have the same secret value without knowing the actual value.
Clear-Text	Data or a key value in un-encrypted form.
CMDID	The two, three, or four-character Command ID.
CRLF	Carriage Return Line Feed. Added to the end of the response. It can be removed by enabling option 23 in the CONFIG_COMMANDS section of the CONFIG.PRM file.
CONFIG.PRM	The file used to configure the Network Security Processor.
Conversion Table	Another name for a decimalization table, used to convert hexadecimal values to decimal values. Conversion tables are used in the IBM 3624, NCR, and Atalla 2x2 PIN verification algorithms.

CVC	Card Validation Code. Check values that confirm the validity of a MasterCard bankcard's magnetic stripe.
CVV	Card Verification Value. Check value that confirms the validity of a VISA bankcard's magnetic stripe.
Decryption	The process of using a key to unscramble data.
DES	Data Encryption Standard. A cryptographic algorithm which employs a 56-bit secret key, adopted by the National Bureau of Standards for data security.
DK	Derivation Key. A working key which is used in a cryptographic process to derive other keys.
DNT	Diebold Number Table.
Double-Length Key	A DES Key used in the 3DES process where the first and third key blocks have the same value.
DUKPT	Derived Unique Key Per Transaction. A key management scheme developed by VISA, used in Point-Of-Sale devices. As the name implies the key used to encrypt is derived by the host security module based on data sent from the device.
Encryption	The process of using a data encryption key to scramble data so that it cannot be read by someone who does not have the key.
Exclusive Or	A process of combining two values on a bit-by-bit basis. If both values contain a one bit, the resulting bit will be zero. If only one of the values contain a one bit the resulting bit will be one. If neither of the values contain a one bit the resulting value will be zero.
Hexadecimal	The character set of 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. When representing a DES key or encrypted DES key (cryptogram), each character contains 4 bits. Therefore a 3key-3DES key which contains192 bits is represented as 48 hexadecimal characters.
IKSN	Initial Key Serial Number. A non-secret value loaded by the acquirer into each PIN Pad to generate the initial PIN encryption key. Each PIN Pad has a unique initial key serial number.
IPEK	Initial PIN Encryption Key. The result of encrypting the IKSN with the Derivation Key.
Issuer Node	The computer that belongs to the financial institution that has an account relationship with the consumer. An issuer can have ATMs or PIN Pads attached to it, enabling it to act as both an issuer and an acquirer.
IV	Initialization Vector. A value that is exclusive or'd with data prior to encryption/decryption.
IVN	Input Verification Number.
KC	Communications Key. Used in ATMs to encrypt information, such as a PIN.
KD	Data Encryption Key. Used to encrypt or decrypt transaction data.
KEK	Key Exchange Key. A cryptographic key used to encrypt working keys.
Key Block	A 3DES key consists of three key blocks, each key block is used in one DES cycle.
Key Component	A portion of the key. It is used to prevent one person from knowing the key. Key components are exclusive or'd together to produce the final secret key. In a 3DES system a key component is made up of three key blocks each block is 16 hexadecimal characters.
KI	Initial Master Key.

KM	ATM Master Key.
KMAC	Message Authentication Code Key. Used to generate or verify the integrity of transmitted data.
KMATM	ATM Master Key. This key is downloaded to an ATM machine from the host computer during initialization to facilitate PIN encryption.
KPE	PIN Encryption Key. Used to encrypt or decrypt PINs.
KPEn	Unique Transaction Key. The key that encrypts the PIN from all but the first transaction.
KPV	PIN Verification Key. Used in an algorithm to verify PINs.
KSN	Key Serial Number. A non-secret value generated from the initial key serial number and an encryption counter, used in the VISA DUKPT key management scheme.
KX	Exchange Key. Another term for Key Exchange Key (KEK),
MAC	See Message Authentication Code.
Master File Key (MFK)	The 3key-3DES key cryptographic key under which all working keys are protected. It is stored in the non-volatile key table. It is not erased if power is removed.
Message Authentication Code	A code derived from applying the DES algorithm and cryptographic key to a message to protect it from alteration.
MFK	Master File Key.
MFK Check Digits	The Master File Key's check digits. Are produced by encrypting zeros with the MFK. The check digits are the leftmost four characters of the result.
Non-volatile Key Table	An area of battery backed up memory used to hold the Master File Key and Pending Master File Key. The contents of this non-volatile key table are maintained during a power outage.
NSP	Network Security Processor. A hardware security module used to perform cryptographic operations.
PAN	Primary Account Number.
Pending Master File Key	A 3key-3DES key that is stored in the non-volatile key table. It is promoted to the current MFK using command 9F.
PIN	Personal Identification Number.
PMK	PIN Master Key.
POS	Point Of Sale.
PVN	PIN Verification Number. The result of processing a PIN through the Identikey algorithm.
PVV	PIN Verification Value. The result of processing a PIN through the VISA algorithm.

Sanity Indicator	A flag returned in a response to indicate where or not the decrypted PIN block is in a valid format. When processing an encrypted PIN, the encrypted PIN block is decrypted inside the NSP. If the NSP determines that this decrypted PIN block is invalid it will set the sanity indicator to N. If the NSP determines that the decrypted PIN block is valid it will set the sanity indicator to Y.
Secure Configuration Assistant-3 (SCA-3)	The device that is used with Atalla Smart cards to initialize and configure the Network Security Processor.
Security Processor	See Network Security Processor (NSP).
Single-Length Key	A DES key that contains 64 bits.
Switch Node	The computer that directs transactions from multiple acquirers to the appropriate issuer. A switch can have ATMs or PIN Pads attached to it.
TMK	Terminal Master Key.
Variant	A value, based on the type of key that is being encrypted, that is exclusive or'd with the encrypting key.
Volatile Table	An area of RAM memory used to hold up to 1000 3key-3DES working keys or conversion tables. The contents of this table are not maintained during a power outage.
Working Keys	A category of keys used to perform specific cryptographic operations. Every working key is encrypted by a unique version of the MFK or a KEK. They are not stored in the non-volatile key table. However they may be stored in the volatile table.
ZCMK	Zone Control Master Key. A VISA term for a Key Exchange Key.