HSM Documentation

HSM Host Interface Protocol
<table>
<thead>
<tr>
<th>Version</th>
<th>Name(s)</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01.00</td>
<td>F. Demaertelaere</td>
<td>10/02/2000</td>
<td>Initial version</td>
</tr>
<tr>
<td>01.01</td>
<td>F. Demaertelaere</td>
<td>29/02/2000</td>
<td>Review remarks included</td>
</tr>
<tr>
<td>01.02</td>
<td>F. Demaertelaere</td>
<td>10/03/2000</td>
<td>Extend the Application Address</td>
</tr>
<tr>
<td>02.00</td>
<td>F. Demaertelaere</td>
<td>24/11/2000</td>
<td>Final version</td>
</tr>
<tr>
<td>02.01</td>
<td>F. Demaertelaere</td>
<td>12/01/2001</td>
<td>Modifications new logo</td>
</tr>
<tr>
<td>03.00</td>
<td>F. Demaertelaere</td>
<td>20/01/2003</td>
<td>Documentation Platform Independent</td>
</tr>
<tr>
<td>03.01</td>
<td>F. Demaertelaere</td>
<td>22/01/2003</td>
<td>After review</td>
</tr>
<tr>
<td>03.02</td>
<td>P.Stienon, F.Wouters</td>
<td>12/04/2006</td>
<td>Few updates, new disclaimer, DEP/T6</td>
</tr>
<tr>
<td>04.00</td>
<td>Anna Papayan</td>
<td>18/03/2011</td>
<td>Change the template into Atos Worldline, minor changes.</td>
</tr>
<tr>
<td>04.01</td>
<td>David Lheureux</td>
<td>02/10/2013</td>
<td>Make this document compatible with DEP and ADYTON</td>
</tr>
<tr>
<td>04.02</td>
<td>Marc Haest</td>
<td>10/10/2013</td>
<td>Mainly typo</td>
</tr>
<tr>
<td>04.03</td>
<td>Pierre BODIN</td>
<td>16/02/2016</td>
<td>Updated error codes list</td>
</tr>
</tbody>
</table>
CONFIDENTIALITY

The information in this document is confidential and shall not be disclosed to any third party in whole or in part without the prior written consent of Atos Worldline S.A./N.V.

COPYRIGHT

The information in this document is subject to change without notice and shall not be construed as a commitment by Atos Worldline S.A./N.V.

The content of this document, including but not limited to trademarks, designs, logos, text, images, is the property of Atos Worldline S.A/N.V. and is protected by the Belgian Act of 30.06.1994 related to author’s right and by the other applicable Acts.

The contents of this document must not be reproduced in any form whatsoever, by or on behalf of third parties, without the prior written consent of Atos Worldline S.A./N.V.

Except with respect to the limited license to download and print certain material from this document for non-commercial and personal use only, nothing contained in this document shall grant any license or right to use any of Atos Worldline S.A./N.V.’s proprietary material.

LEGAL DISCLAIMER

While Atos Worldline S.A./N.V. has made every attempt to ensure that the information contained in this document is correct, Atos Worldline S.A./N.V. does not provide any legal or commercial warranty on the document that is described in this specification. The technology is thus provided “as is” without warranties of any kind, expressed or implied, included those of merchantability and fitness for a particular purpose. Atos Worldline S.A./N.V. does not warrant or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, product or process disclosed.

To the fullest extent permitted under applicable law, neither Atos Worldline S.A./N.V. nor its affiliates, directors, employees and agents shall be liable to any party for any damages that might result from the use of the technology as described in this document (including without limitation direct, indirect, incidental, special, consequential and punitive damages, lost profits).

JURISDICTION AND APPLICABLE LAW

These terms shall be governed by and construed in accordance with the laws of Belgium. You irrevocably consent to the jurisdiction of the courts located in Brussels for any action arising from or related to the use of this document.
# TABLE OF CONTENTS

TABLE OF CONTENTS ........................................................................................................... 4

1. SCOPE OF THE DOCUMENT .................................................................................. 5
   1.1. REFERENCES ........................................................................................................ 5
   1.2. CONTACTING ATOS WORLDLINE ................................................................. 5

2. COMMUNICATION ARCHITECTURE ..................................................................... 6

3. COMMUNICATION PROTOCOLS ......................................................................... 7
   3.1. POINT-TO-POINT PROTOCOLS .................................................................. 7
       3.1.1. PDP Asynchronous on RS232 (not ADYTON) ........................................... 7
           3.1.1.1. Overall RS232 Characteristics ................................................................. 7
           3.1.1.2. Asynchronous PDP Protocol with LRC .................................................... 7
           3.1.1.3. Asynchronous PDP Protocol with CRC ................................................... 8
           3.1.1.4. Protocol Flowchart ................................................................................ 9
           3.1.1.5. CRC Sample Code ................................................................................ 10
       3.1.2. Other Protocols .......................................................................................... 11
   3.2. NETWORK PROTOCOLS .............................................................................. 11
       3.2.1. TCP/IP on Ethernet .................................................................................... 11
           3.2.1.1. Overall TCP/IP Characteristics .............................................................. 11
           3.2.1.2. Additional Message Information ............................................................ 11
       3.2.2. Other Protocols ........................................................................................ 12

4. APPLICATION PROTOCOLS .............................................................................. 12
   4.1. HSM COMMAND STRUCTURES ................................................................. 12
       4.1.1. DS2 Command Structure ............................................................................ 12
       4.1.2. DS3 Command Structure ............................................................................ 12
       4.1.3. DS4 Command Structure ............................................................................ 13
       4.1.4. DS5 Command Structure (only for ADYTON) ............................................. 13
   4.2. DEP PROTOCOL (DP) ..................................................................................... 14
       4.2.1. DP for DEP Platforms ............................................................................... 14
       4.2.2. DP for ADYTON ....................................................................................... 15
   4.3. ENHANCED DEP PROTOCOL (EDP) .......................................................... 15
       4.3.1. Magic Number ........................................................................................... 16
           4.3.1.1. DEP Platforms ..................................................................................... 16
           4.3.1.2. ADYTON ............................................................................................. 16
       4.3.2. Header Version ........................................................................................... 16
       4.3.3. Destination/Source Address ..................................................................... 17
       4.3.4. Host Message Identification Length ........................................................... 19
       4.3.5. Host Message Identification ....................................................................... 19
       4.3.6. Command .................................................................................................. 19
       4.3.7. Summary Table ........................................................................................ 19
       4.3.8. Example ................................................................................................... 20
   4.4. MIX OF PROTOCOL ....................................................................................... 20
       4.4.1. Rules .......................................................................................................... 20
       4.4.2. Treatment of Messages .............................................................................. 21
       4.4.3. Recommendation ...................................................................................... 21

5. BUFFERING COMMAND MESSAGES ................................................................ 22
1. SCOPE OF THE DOCUMENT

This HSM Host Interface Protocol document defines the interface between the HSM and the host.

In this document, the term HSM is used for both DEP Platforms and ADYTON.

It describes in detail the structures of the messages that have to be sent at application level by the host to the HSM. Moreover, this document gives also a description of the messages at communication level.

The document is required when implementing the host driver that communicates with the HSM.

1.1. REFERENCES

This document contains references to other documents about the DEP. This paragraph gives a list of all the documents referred to.

- DEP/Linux User Manual
- HSM DS3, DS4 and DS5 Principles
- DEP/NMS User Manual
- DEP/T6 Owner’s Manual
- ADYTON Reference Guide

There are no references made to the following documents, but they could be useful to understand this document.

- DEP Introduction to DEP
- DEP General Architecture
- DEP Glossary

1.2. CONTACTING ATOS WORLDLINE

You can visit Atos Worldline on the World Wide Web to find out about new products and about various other fields of interest.


For the documentation visit http://www.banksys.com web page.

For support on issues related to DEP, customers, partners, resellers, and distributors can send an email to the DEP Hotline:

mailto:dephotline-atosworldline@atosorigin.com.
2. COMMUNICATION ARCHITECTURE

The communication between the host and the HSM is based on a client-server architecture where

- the host is the client part of the architecture it sends commands to the Software of the HSM and waits for responses
- the HSM is the server part of the architecture: it waits for commands, executes them (by means of its Software) and sends the responses back to the client part.

The communication between the two parts has to pass through a single bi-directional communication channel. Depending on the underlying communication protocols, the channel is a point-to-point communication line or a network.

In some circumstances (see paragraph 3.2.1 on page 11), there may be more than one host communicating with a HSM at the time.

The communication between the client and the server is organized in several communication protocol levels. Considering the stack of protocol layers, we can divide it in two levels:

- The bottom level consists of standard communication protocols. This level allows messages to be transported back and forth between the host and the HSM. This level, called the Communication Protocol Level, handles the communication channel connecting the host and the HSM.
- The top level is a single layer level, consisting of an Atos Worldline-defined application protocol. This level, called the Application Protocol Level, allows messages to be exchanged between a host application and a particular Application Software of the HSM.

The figure presented below illustrates the two levels for a particular case: the HSM protocol over TCP/IP on Ethernet.
3. COMMUNICATION PROTOCOLS

As described previously, this level ensures the communication between the host and the HSM, regardless of the final application destination. This level is implemented by different protocols. They can be classified in two different groups:

- Point-to-Point Protocols (not supported by ADYTON)
- Network Protocols

For the supported network protocols, initially, there was only one virtual circuit connecting the host to the HSM. This implied that there was no possibility to connect two host applications to a HSM at the same time. Recently, Atos Worldline added a multi-connection property so that multiple hosts can be connected to one HSM.

3.1. POINT-TO-POINT PROTOCOLS

3.1.1. PDP Asynchronous on RS232 (not ADYTON)

Asynchronous RS232 communication is available on DEP/XP and DEP/T6.

3.1.1.1. Overall RS232 Characteristics

The overall RS232 communication characteristics of the PDP Asynchronous protocol are defined here.

- baudrate: 4800, 9600, 19200, 38400, 56000, 57600 or 115200,
- stop bits: 1 stop bit,
- characters: 8 bits, no parity.

There are two possible checksums in the PDP Asynchronous protocol:

- LRC
- CRC

The PDP Asynchronous protocol parameters need to be specified in the DEP/NMS application (refer to DEP/NMS User Manual for more information).

Because the CRC checksum is stronger than the LRC, it is recommended to use the CRC checksum.

3.1.1.2. Asynchronous PDP Protocol with LRC

The byte stream that has to be transferred between the host and the HSM is defined in the following way:
The `<LRC>` should be calculated from the first `<DLE>` to the last `<ETX>`, both included. `<LRC>` calculation is done before duplicating the `<DLE>` characters in the `<MESSAGE>`.

For example, if the following command (I_STD_ECHO) shall be sent to the DEP Platform:

```
FF 01000100 0005010A102530 02000900 01000200
```

the following byte streams will be exchanged between the host and the DEP Platform. All the added protocol characters are underlined and the characters included in the `<LRC>` are indicated in bold.

```
Host to DEP Platform:
10 02 FF 01000100 0005010A102530 02000900 01000200 10 03 FD

DEP Platform to Host:
10 02 00 01000200 0005010A102530 10 03 09
```

### 3.1.1.3. Asynchronous PDP Protocol with CRC

The byte stream that has to be exchanged between the host and the DEP Platform is defined in the following way:
The `<CRC>` should be calculated only over the `<MESSAGE>`. `<CRC>` calculation is done before duplicating the `<DLE>` characters in the `<MESSAGE>`.

For example, if the following command (`I_STD_ECHO`) shall be sent to the DEP Platform:

```
FF 01000100 0005010A102530 02000900 01000200,
```

the following byte streams will be exchanged between the host and the DEP Platform. All the added protocol characters are underlined and the characters included in the CRC are indicated in bold.

```
Host to DEP Platform:
10 02 FF 01000100 0005010A102530 02000900 01000200 10 03 34 B3

DEP Platform to Host:
10 02 00 01000200 0005010A10102530 10 03 97 0F
```

### 3.1.1.4. Protocol Flowchart

When communicating with the DEP Platform by using the PDP Asynchronous protocol, the following flow has to be taken into account.
3.1.1.5. CRC Sample Code

This paragraph gives a sample C-code for the calculation of the CRC.

```c
#include <stdio.h>

#define word unsigned short
#define byte unsigned char
#define dword unsigned long

void CRCUpdate ( byte in_data, dword *accum )
{
    word i;
    word c1, c2, c3, c4;
    c1 = 65535;
    c2 = 32768;
    for (i=0; i<=7; i++) {
        c3 = (in_data*256) & c1;
        c4 = (*accum ^ c3) & c2;
        if ( c4 != 0 ) {
            *accum ^= 0x0810;
            *accum *= 2;
            *accum |= 1;
            *accum &= c1;
        }
        else {
            *accum *= 2;
            *accum &= c1;
        }
        in_data = (in_data*2) & 0xFF;
    }
}

int CRCCalc ( byte *mes, word len, word *CRC )
{
    word i;
    dword DCRC;
    DCRC = 65535;
    for (i=0; i<len; i++)
        CRCUpdate (mes[i], &DCRC);
    *CRC = DCRC & 0xFFFF;
    return(0);
}

void main ( void )
{
    byte mes1[20];
    byte mes2[20];
    word CRCFinal;
    mes1[0] = 0x02;
    mes2[0] = 0x00;
    CRCFinal = 0x00;
    CRCCalc (&mes1[0], 1, &CRCFinal);
    printf ("CRC1= %02X %02X
\n", CRCFinal>>8, CRCFinal&0xFF);
    CRCCalc (&mes2[0], 2, &CRCFinal);
    printf ("CRC2= %02X %02X\n", CRCFinal>>8, CRCFinal&0xFF);
}
```
3.1.2. Other Protocols

Other protocols could possibly be available in the future (on request).

3.2. NETWORK PROTOCOLS

3.2.1. TCP/IP on Ethernet

3.2.1.1. Overall TCP/IP Characteristics

The standard TCP/IP protocol could be used for establishing communication with the HSM. This assumes that a standard TCP/IP stack has been installed. Connecting to a HSM should be done by the standard TCP/IP connection procedure.

The TCP/IP address and port number can be defined on the DEP/XP, DEP/Linux or DEP/T6. Refer to the DEP/Linux User Manual or the DEP/T6 Owner’s Manual to know how these setting have to be defined.

The TCP/IP address can be defined on the ADYTON but the TCP/IP port cannot be set. Refer to the ADYTON Reference Guide for more information.

In case of multiple connections, every host should connect to the HSM by using the same TCP/IP address and port number. Remark that there are some restrictions when using multiple connections (e.g. the number of simultaneous connections).

3.2.1.2. Additional Message Information

Because TCP is a byte-stream protocol that provides no message boundaries, all application messages transmitted through the TCP connection are preceded by a length field of four bytes, the Application Message Length field.

```
<table>
<thead>
<tr>
<th>Application Message Length</th>
<th>Application Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSB</td>
<td>MSB</td>
</tr>
</tbody>
</table>
```

The Application Message Length field is a hexadecimal field with the least significant byte (LSB) as the first byte and the most significant byte (MSB) as the last byte.

For example, if the following command (I_STD_ECHO) shall be sent to the HSM:

```
FF 01000100 0005010A102530 02000900 01000200,
```

the following byte streams will be exchanged between host and HSM. The added Application Message Length field bytes are underlined.
3.2.2. Other Protocols

Other protocols could possibly be available in the future (on request).

4. APPLICATION PROTOCOLS

4.1. HSM COMMAND STRUCTURES

4.1.1. DS2 Command Structure

All DS2 commands and replies are composed of a fixed sequence of fields.

Such HSM command always starts with a one-byte command code, ranging from 0x00 to 0xFD. Depending on the value of the command code, several pre-defined data objects need to follow the command code.

Each data object has a fixed length value and a dedicated place in the command. In case of variable length fields, the length of the data value is always available in the data object.

The HSM has all the information about the structure of the data objects.

<table>
<thead>
<tr>
<th>Field</th>
<th>Command Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00-0xFD</td>
<td>FIELD1 FIELD2 FIELD3 ... FIELD n</td>
</tr>
</tbody>
</table>

The reply of the HSM contains a one-byte reply code, followed by several pre-defined data objects.

<table>
<thead>
<tr>
<th>Field</th>
<th>Reply Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>reply dependent</td>
<td>FIELD1 FIELD2 FIELD3 ... FIELD n</td>
</tr>
</tbody>
</table>

4.1.2. DS3 Command Structure

A HSM DS3 command always starts with a one-byte DS3 identifier, i.e. 0xFF. This identifier is followed by a list of input data tags immediately followed by their corresponding actual data, a list of functions (tags) to execute and a list of output data tags.

<table>
<thead>
<tr>
<th>Field</th>
<th>Command Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFF</td>
<td>DATA TAG1 VALUE1 DATA TAG n VALUE n FUNC TAG1 FUNC TAG m DATA TAG1 DATA TAG</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Command Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>reply dependent</td>
<td>DATA TAG1 VALUE1 DATA TAG n VALUE n FUNC TAG1 FUNC TAG m DATA TAG1 DATA TAG</td>
</tr>
</tbody>
</table>
The reply of the HSM contains a one-byte response code, followed by several tags and their corresponding data.

<table>
<thead>
<tr>
<th>Reply Code (1 byte)</th>
<th>Reply Data Structure (variable length)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>FIELD1 FIELD2 FIELD3 ... FIELD n</td>
</tr>
</tbody>
</table>

Refer to the document *HSM DS3, DS4 and DS5 Principles* for more information.

### 4.1.3. DS4 Command Structure

A HSM DS4 command always starts with a one-byte identifier, i.e. 0xFF. This identifier is followed by one function tag and a structure of data objects.

Each data object has a fixed length value and a dedicated place in the command. In case of variable length fields, the length of the data value is always available in the data object.

The HSM has all the information about the structure of the data objects.

<table>
<thead>
<tr>
<th>DS4 Identifier</th>
<th>Command Data Structure (variable length)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFF</td>
<td>FUNCTION TAG FIELD1 FIELD2 FIELD3 ... FIELD n</td>
</tr>
</tbody>
</table>

The reply of the HSM contains a one-byte response code, followed by several tags and their corresponding data. In case of a correct answer (Reply Code is 0x00), the HSM will return a DS2-like answer.

<table>
<thead>
<tr>
<th>Reply Code (1 byte)</th>
<th>Reply Data Structure (variable length)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>FIELD1 FIELD2 FIELD3 ... FIELD n</td>
</tr>
</tbody>
</table>

Error cases are replied by a DS3-like answer.

<table>
<thead>
<tr>
<th>Reply Code (1 byte)</th>
<th>Reply Data Structure (variable length)</th>
</tr>
</thead>
<tbody>
<tr>
<td>reply dependent</td>
<td>TAG 1 VALUE1 ... TAG n VALUE n</td>
</tr>
</tbody>
</table>

Refer to the document *HSM DS3, DS4 and DS5 Principles* for more information.

### 4.1.4. DS5 Command Structure (only for ADYTON)

To get more information about DS5, refer to the specific documentation of the loaded software.
4.2. DEP PROTOCOL (DP)

4.2.1. DP for DEP Platforms

The DEP Platform offers the possibility that a sequence number precedes the DS2/DS3/DS4 command. The reply will be preceded with the same sequence number.

The sequence number field needs to be managed by the host and offers the possibility for the host to identify command and reply messages. Although this field is called a sequence number field, the use of a sequence number is just one of the possible implementations for the message and reply identification. The feature offers a general possibility…

The sequence number field could be useful for hosts accessing different DEP Platforms to make the link between the command sent and the reply received.

The sequence number is optional and its length may vary from 0 to 15 bytes. The length of this field must be defined in the DEP/NMS application and in the DEPD Daemon Configuration File for the DEP/Linux.

![Diagram of DEP protocol](image)

The command field contains the instructions to be processed by the DEP Crypto Module and can either have a DS2 command structure, a DS3 command structure or a DS4 command structure.

When the DP is used for sending messages to a DEP Platform containing different DEP Crypto Modules, these messages are always forwarded internally to the DEP POOL (see paragraph 4.3.3 on page 17).

For example, if the following command (**I_STD_ECHO**) will be sent to the DEP Platform in DP with a sequence number of 4 bytes

```
FF 01000100 0005010A102530 02000900 01000200,
```

the following byte streams (without communication protocol) will be exchanged between the host and the DEP Platform. The added sequence number field bytes are underlined.
4.2.2. DP for ADYTON

The command field contains the instructions to be processed by the ADYTON and can either have a DS2 command structure, a DS3 command structure, a DS4 command structure or a DS5 command structure.

For example, if the following command \textit{(I\_STD\_ECHO)} will be sent to the ADYTON

\begin{verbatim}
FF 01000100 0005010A102530 02000900 01000200
\end{verbatim}

the following byte streams (without communication protocol) will be exchanged between the host and the DEP Platform.

\begin{verbatim}
Host to ADYTON:
FF 01000100 0005010A102530 02000900 01000200
\end{verbatim}

\begin{verbatim}
ADYTON to Host:
00 01000200 0005010A102530
\end{verbatim}

4.3. ENHANCED DEP PROTOCOL (EDP)

The DEP Protocol described in paragraph 4.2 on page 14 offers only a basic principle for communicating with a HSM.

The Enhanced DEP Protocol is extended with some new concepts:

- allowing the host to specify some more information concerning the destination of the message

- allowing additional security applications and security hardware modules to be integrated in the system.
The ADYTON supports the EDP for backward compatibility reasons. It means that some parameters cannot be set. It will be described in this section.

This protocol consists of command messages sent by the host applications to the HSM and the reply messages returned by the HSM to the host applications. To each command corresponds one and only one reply.

4.3.1. Magic Number

4.3.1.1. DEP Platforms

The Magic Number can be configured in the DEP/NMS application and the DEP Daemon Configuration File of the DEP/Linux. It allows making a difference between the DEP Protocol and the Enhanced DEP Protocol.

The Magic Number received from the host application should match the Magic Number that is configured on the DEP Platform. The same value of the Magic Number is returned in the reply message.

The length of the Magic Number can range from 0 to 16 bytes.

Refer to the DEP/NMS User Manual to know how to set the Magic Number for the DEP/T6 and to the DEP/Linux User Manual for the DEP/Linux.

4.3.1.2. ADYTON

The Magic Number allows making a difference between the DEP Protocol and the Enhanced DEP Protocol.

The length of the Magic Number is fixed to 1 and its value is fixed to 0xFE.

The Magic Number received from the host application should match the Magic Number that is defined on the ADYTON. The same value of the Magic Number is returned in the reply message.

4.3.2. Header Version

The Header Version field is created to allow handling different versions of the protocol. It has a fixed length of one byte.
The most significant nibble (4 bits) indicates the release number and the least significant nibble is the version number. The *Header Version* is also returned in the reply message.

Actually, the value of the *Header Version* should be 0x30.

### 4.3.3. Destination/Source Address

The *Destination/Source Address* consists of two bytes. The *Destination Address* is used in command messages to the HSM; the *Source Address* is used in the reply messages from the HSM.

The first byte of the *Destination/Source Address* is the *Application Address* and identifies the application that must receive the message (in a command) or that replied (in a reply).

For applications controlling several hardware modules, the second byte of the *Destination/Source Address* (*Device Address*) identifies the hardware module for which the message was sent (or the one that replied the response message). The interpretation of the second byte is application dependent.

**Important note:** in ADYTON the concept of Crypto Module doesn’t exist. So the *application address* and *device address* are not relevant. But for backward compatibility reasons, the only possible values are:

- Application address: 0x01
- Device address: 0x00 and 0x01

In the reply, the *application address* will be always 0x01 and *device address* will be always 0x01.

The following table gives an overview of the supported *Destination Addresses* and their meaning.

<table>
<thead>
<tr>
<th>APPLICATION ADDRESS</th>
<th>DEVICE ADDRESS</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01 - DEP Handler or DEP Daemon</td>
<td>0x00</td>
<td>(DEP POOL) DEP Handler makes a load balancing between the available pool of DEP Crypto Modules</td>
</tr>
<tr>
<td></td>
<td>0x01</td>
<td>(FIRST) Command must be treated by the first DEP Crypto Module</td>
</tr>
<tr>
<td></td>
<td>0x02</td>
<td>(SECOND) Command must be treated by the second DEP Crypto Module</td>
</tr>
<tr>
<td></td>
<td>0x03</td>
<td>(THIRD) Command must be treated by the third DEP Crypto Module</td>
</tr>
<tr>
<td></td>
<td>0x04</td>
<td>(FOURTH) Command must be treated by the fourth DEP Crypto Module</td>
</tr>
</tbody>
</table>

The *Destination Address* in the command is duplicated in the response message, but should then be interpreted by the host as a *Source Address* instead of a *Destination Address*. The *Application Address* in the reply message will be the same as the *Application Address* in the command message, except if a protocol error has been detected.
The following table gives an overview of the possible returned *Source Addresses* and their meaning.

<table>
<thead>
<tr>
<th>APPLICATION ADDRESS</th>
<th>DEVICE ADDRESS</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01 - DEP Handler or DEP Daemon</td>
<td>0x01-0x04</td>
<td>(OK) The Device Address indicates the DEP Crypto Module that treated the command message (0x01=FIRST, 0x02=SECOND, 0x03=THIRD, 0x04=FOURTH)</td>
</tr>
<tr>
<td></td>
<td>0x80</td>
<td>(ERROR) A command message was sent to the DEP POOL, but there is no DEP Crypto Module in the pool</td>
</tr>
</tbody>
</table>
|                      | 0x81 | (ERROR) Invalid message:
- Invalid length (<= 0)
- Invalid ICS (Internal Communication System) version
- Invalid message type
- Destination APP address is not “DEP Handler” |
|                      | 0x82 | (ERROR) An invalid Device Address was specified |
|                      | 0xA0-0xA3 | (ERROR) The accessed DEP Crypto Module was in the FATAL mode (0xA0=FIRST, 0xA1=SECOND, 0xA2=THIRD, 0xA3=FOURTH) |
|                      | 0xA4-0xA7 | (ERROR) The accessed DEP Crypto Module was in the OFF-LINE mode (0xA4=FIRST, 0xA5=SECOND, 0xA6=THIRD, 0xA7=FOURTH) |
|                      | 0xA8-0xAB | (ERROR) The accessed DEP Crypto Module did not return a response within the maximum response time specified (0xA8=FIRST, 0xA9=SECOND, 0xAA=THIRD, 0xAB=FOURTH) |
|                      | 0xAC-0xAF | (ERROR) The accessed DEP Crypto Module returned an invalid response to the DEP Handler or DEP Daemon (0xAC=FIRST, 0xAD=SECOND, 0xAE=THIRD, 0xAF=FOURTH) |
|                      | 0xB0-0xB3 | (ERROR) The Host Messages attribute was not set for the DEP Crypto Module (0xB0=FIRST, 0xB1=SECOND, 0xB2=THIRD, 0xB3=FOURTH) |
|                      | 0xB8-0xBB | (ERROR) There was an error sending the command to the DEP Crypto Module (0xB8=FIRST, 0xB9=SECOND, 0xBA=THIRD, 0xBB=FOURTH) |
|                      | 0xC0 | (PRINT ERROR) in Open command |
|                      | 0xC1 | (PRINT ERROR) in Send command |
|                      | 0xC2 | (PRINT ERROR) in Receive command |
|                      | 0xC4 | (PRINT ERROR) Timeout |
|                      | 0xC5 | (PRINT ERROR) in Close command |
|                      | 0x9C-0x9F | (ERROR) There was an error receiving the reply from the DEP Crypto Module (0x9C=FIRST, 0x9D=SECOND, 0x9E=THIRD, 0x9F=FOURTH) |
| others               | 0xFF | (PRINT ERROR) in Close command |

0xFF - Protocol Error

<table>
<thead>
<tr>
<th>DEVICE ADDRESS</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>(ERROR) An invalid Header Version was sent to the DEP Platform</td>
</tr>
<tr>
<td>0x02</td>
<td>(ERROR) An invalid Application Address was sent to the DEP Platform</td>
</tr>
<tr>
<td>0x03</td>
<td>(ERROR) The addressed application was not available (either not installed or not started)</td>
</tr>
<tr>
<td>0x05</td>
<td>(ERROR) An incorrect Magic Number was sent to the DEP Platform</td>
</tr>
<tr>
<td>0x06</td>
<td>(ERROR) An invalid Host Message Identification Length was sent to the DEP Platform</td>
</tr>
<tr>
<td>0x07</td>
<td>(ERROR) The length of the sent message was not correct</td>
</tr>
<tr>
<td>others</td>
<td>(ERROR) An undefined protocol error was returned by the DEP Platform</td>
</tr>
</tbody>
</table>
4.3.4. Host Message Identification Length

This one-byte field identifies the length of the Host Message Identification field. The value ranges from 0 to 15.

When no Host Message Identification is used, the Host Message Identification Length field should be set to 0. When a non-zero value is specified, this field should be followed by a Host Message Identification field of the specified length (see paragraph 4.3.5 on page 19).

4.3.5. Host Message Identification

The Host Message Identification field needs to be managed by the host and offers the possibility for the host to identify (e.g. keep together) command and reply messages. The Host Message Identification field could be useful for hosts accessing several HSMs (e.g. routing) to make the link between the command sent and the reply received.

The HSM will duplicate the received Host Message Identification information in its reply message.

The length of this field is identified in the Host Message Identification Length field that is preceded (see paragraph 4.3.4 on page 19).

4.3.6. Command

The command field contains the instructions to be processed by the DEP Crypto Module, and can be formatted either in DS2, DS3, DS4 or DS5.

4.3.7. Summary Table

This table gives an overview of all the parameters used in the Enhanced DEP Protocol.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>LENGTH</th>
<th>VALUE (DEP Platforms)</th>
<th>VALUE (ADYTON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAGIC NUMBER</td>
<td>DEP Platforms: specified in the DEP/NMS application or DEPD Daemon Configuration File - (0-16) ADYTON: fixed to 1</td>
<td>Specified in the DEP/NMS application or DEPD Daemon Configuration File</td>
<td>Fixed to 0xFE</td>
</tr>
<tr>
<td>HEADER VERSION</td>
<td>1 byte</td>
<td>0x30</td>
<td>0x30</td>
</tr>
<tr>
<td>DESTINATION ADDRESS</td>
<td>2 bytes</td>
<td>Application Address: 0x00-0x06 Device Address: application dependent</td>
<td>Application Address: 0x01 Device Address: 0x01</td>
</tr>
<tr>
<td>HOST MESSAGE IDENTIFICATION LENGTH</td>
<td>1 byte</td>
<td>0x00-0x0F</td>
<td>0x00-0x0F</td>
</tr>
<tr>
<td>HOST MESSAGE IDENTIFICATION</td>
<td>Specified in Host Message Identification Length</td>
<td>Managed by the host</td>
<td>Managed by the host</td>
</tr>
</tbody>
</table>
4.3.8. Example

For example, if the following command (I_STD_ECHO) will be sent to the HSM in DP with a Host Message Identification Length of 4 bytes

```
FF 01000100 0005010A102530 02000900 01000200,
```

the following byte streams (without communication protocol) will be exchanged between host and HSM. The added protocol field bytes are underlined.

- using the POOL of DEP Crypto Modules

  **Host to the POOL HSM:**
  ```
  FE 30 0100 04 12345678 FF 01000100 0005010A102530 02000900 01000200
  ```

  **HSM (DEP Crypto Module 2) to Host:**
  ```
  FE 30 0102 04 12345678 00 01000200 0005010A102530
  ```

- addressing directly the DEP Crypto Module 3

  **Host to the DEP Crypto Module 3 HSM:**
  ```
  FE 30 0103 04 12345678 FF 01000100 0005010A102530 02000900 01000200
  ```

  **HSM (DEP Crypto Module 3) to Host:**
  ```
  FE 30 0103 04 12345678 00 01000200 0005010A102530
  ```

4.4. MIX OF PROTOCOL

4.4.1. Rules

When mixing both the DEP Protocol and the Enhanced DEP Protocol, some rules have to be respected to allow the HSM to unambiguously recognize the protocol used for each message. Not respecting these rules may cause a wrong interpretation of some message headers.

- The **Sequence Number** field of the DEP Protocol should not start with the **Magic Number** defined in the Enhanced DEP Protocol. This can be achieved by setting the length of the **Magic Number** field (Enhanced DEP Protocol) equal to the length of the **Sequence Number** field (DEP Protocol) plus one.

- The last byte of the **Magic Number** field should be 0xFE.

When the length of the **Magic Number** field is set to zero, no mix of protocols is allowed and all messages are assumed to respect the Enhanced DEP Protocol.
4.4.2. Treatment of Messages

Each command message received from the host is analyzed using the following rules:

1. When the length of the Magic Number field is set to zero or when the first byte(s) of the message match the Magic Number, the format of the message is assumed to correspond to the Enhanced DEP Protocol. The other fields are analyzed:

   1.1. If the Header Version is invalid, the protocol error Invalid Header Version in Host Command is returned.

   1.2. If the Host Message Identification Length is invalid (longer than 15 bytes), the protocol error Invalid Length Host Message ID Field in Host Command is returned.

   1.3. The Application Address is checked:

       1.3.1. If the address corresponds to a configured and started application of the HSM, the message is passed to it. The command is processed by the application and the reply is sent back to the host.

       1.3.2. If the address corresponds to a non-installed, a non-configured or a not started application, the protocol error Destination in Host Command not Available is returned.

       1.3.3. Otherwise, the address is considered as an invalid address and the protocol error Invalid Destination in Host Command is returned.

2. Otherwise, it is assumed not to be an Enhanced DEP Protocol command message. The Device Address is assumed to be POOL, allowing the message command to be treated by any DEP Crypto Module of the DEP Platform. The concept of POOL doesn’t exist in ADYTON, the message will be treated by the Platform itself.

4.4.3. Recommendation

To take the advantage of some specific functionality, it is recommended that the Enhanced DEP Protocol be used for new developments.

E.g. the Enhanced DEP Protocol allows the host to access a dedicated DEP Crypto Module of one HSM. This could be interesting when the DEP Crypto Modules are loaded with different Application Software and/or keys.

Although retro-compatibility is guaranteed in the future, it is not excluded that the Enhanced DEP Protocol is extended with additional functionality, or that the HSM functionality is extended with new features by using the Enhanced DEP Protocol.
5. BUFFERING COMMAND MESSAGES

Several command messages may be sent by the host to the HSM before receiving the first response from the HSM. The HSM will memorize each command message until the resource needed to execute the command message becomes free. If several command messages need the same resource, they will be scheduled. The response message corresponding to the command message is sent as soon as it is available.

At the host side, the relationship between the command messages and response messages can be ensured by the Sequence Number field (DEP Protocol) (only for DEP Platforms) or the Host Message Identification (Enhanced DEP Protocol).

The goal of this mechanism is to maximize the throughput between the host and the HSM. The improvement will particularly be significant on low speed communication lines.

In the DEP Platforms, the number of command messages simultaneously accepted by the HSM is limited to a maximal value, to be defined in the DEP/NMS application (see the document DEP/NMS User Manual).and the DEPD Daemon Configuration File (see the document DEP/Linux User Manual).

Not relevant for ADYTON.