

Commands of Centralized Security System for the BIT Project

3940349E-BIT-DOC-CON-CTM-AIT-A ❖ 16/12/2009 ❖ Version: 0.1.0



**Billeteaje Inteligente para
el Transporte de Madrid**

CONFIDENCIAL

© 2002 Consorcio Regional de Transportes de Madrid.

Todos los derechos reservados.

La información contenida en este documento es
confidencial; no podrá ser reproducida total o
parcialmente ni ser hecha disponible a terceros sin
autorización escrita del CRTM.

CRTM – Pza. Descubridor Diego de Ordás 3 – 28003 –
Madrid - España

Índex

1. DOCUMENT CONTROL	5
2. GENERAL INFORMATION FOR HSM CONNECTION	6
2.1. Command information	6
2.2. Address access	7
3. COMMANDS FOR THE PREPERSONALIZATION APPLICATION	8
3.1. Start Operation Command	8
3.2. End Operation Command	10
3.3. Diversified Key Generation Command	11
3.4. Transaction Record Generation Command	12
4. COMMANDS FOR THE PERSONALIZATION APPLICATION	14
4.1. Start Operation Command	14
4.2. End Operation Command	14
4.3. Transaction Generation Record Command	14
4.4. Session Key Generation (Authentication) – Part 1	15
4.5. Session Key Generation (Authentication) – Part 2	17
4.6. Sube-T Card Number Generation Command	18
4.7. HSM Serial Number Supply Command	19
5. COMMANDS FOR THE TICKET-VENDING APPLICATION	20
5.1. Start Operation Command	20
5.2. End Operation Command	20
5.3. Transaction Generation Command	20
5.4. Session Key Generation (Authentication) – Part 1	20
5.5. Session Key Generation (Authentication) – Part 2	20
6. COMMAND EXECUTION	21
7. LIST OF COMMAND RESPONSE CODES	22
8. IMPORTANT COMMENTS	23

Table list

Table 1 Version control	5
Table 2 Distribution list	5
Table 3 Command response codes	22

CONFIDENCIAL

1. DOCUMENT CONTROL

Table 1 Version control			
No.	Version	Date	Comments
1	0.1.0	16/12/2009	Document creation.
2	-	-	-
3	-	-	-
4	-	-	-
5	-	-	-

Table 2 Distribution list			
No.	Version	Date	Distribution details
1	0.1.0	16/12/2009	Peter Helderma, Olivier Hupe, Chandak Mukesh Damien Bordron, Frederic Noyer (GEMALTO), Pedro Martinez, Juan Moreno (AVANT STUDIO).
2	-	-	-
3	-	-	-

2. GENERAL INFORMATION FOR HSM CONNECTION

2.1. Command information

The HSM is the core element of the Centralized Security System. In fact, the HSM is a remote security module. As a result, in order to execute its commands, it is necessary to encapsulate them in some type of protocol that permits to communicate with the HSM. It is applied HTTP, and the communication between the remote location and the security module is done through POST, using the POST command with the context type: **application/x-www-form-urlencoded**. The necessary variables of each command are sent to it.

The format of the response will have a header consisting of the two following variables:

```
Content-type: text/plain
Content-length: <length>
```

The length specified by **content-length** indicates the length of the message.

Each line is finished with **<CR><LF>**. The command return follows, leaving a blank line between the header and main part:

```
Content-type: text/plain <CR><LF>
Content-length: 50 <CR><LF>
<CR><LF>
<Main part of command return>
```

The main part of the command consists of a set of variables that the command returns. Each variable is followed by the symbol "=" and then the value is specified. At the end of each **<variable>=<valor>** goes **<CR><LF>**.

The value of the variable will always be a string. When there is data in binary system, a notation in hexadecimal system will be used, without any other indication. For example, the card serial number, which consists of 7 Bytes, will be encoded (in the command call) as a string of 14 characters. Care has to be taken with the syntax of the commands, because the system is key sensitive. In all cases, the command result code contains 0, if there is a successful execution. In the unlikely event of receiving a different code, the command should be executed again.

For example if the client application sends:

```
POST /pagename HTTP/1.0<CR><LF>
Content-type: application/x-www-form-urlencoded<CR><LF>
Content-length: 86 <CR><LF>
<CR><LF>
Variable1=value%20of%20the%20variable%20one&
Variable2=value%20of%20the%20variable%20two
```

<CR><LF>

The server application responds:

```
Content-type:text/plain <CR><LF>
Content-length:16 <CR><LF>
<CR><LF>
Variable1=Test<CR><LF>
```

2.2. Address access

The direction for the HSM access is:

<http://IP:PR/Application/Version/Command>

Where the term **Command**, represents the command name.

If an error occurs during a command execution, this command should be executed again.

3. COMMANDS FOR THE PREPERSONALIZATION APPLICATION

3.1. Start Operation Command

This command is invoked whenever the other part launches a new operation.

COMMAND

<http://IP:PR/Application/Version/InitOperation>

PARAMETERS

- Id** It represents the identifier of the process that accesses to the HSM. In case of GEMALTO contains the text "**GEMALTO**".
- Rol** The role determines the use of various commands. For GEMALTO contains the value 0x06.

RESPONSE

- CodResponse** This is the response code. It contains 0 if there is no error.
- jsessionId** This is the code that identifies the operation. The session identifier is necessary for authentication purposes in the rest of the operations.
- OperationNumber** This parameter contains the number assigned to this operation.
- HsmSerial** This is the serial number (in hexadecimal system) of the HSM used for this operation.

EXAMPLE

http://10.10.50.14:80/BIT/HSM20/InitOperation?id=toolsHSM_ver2.2&rol=06

RESULT

```
CodResponse = 0
JsessionId = 3F21EFB049B26073441E9A2714CC0542.worker2
OperationNumber = 00000000000000387
HsmSerial = ab604d
```

The value of the variable **jsessionId**, obtained from the command response of **InitOperation**, should be passed to the rest of command calls, until the operation is finished (**FinishOperation**). The right way to do it is using a technique known as URL writing. The following string should be added to the URL used to invoke the command:

"jsessionId=[valor de variable jsessionId]"

The input parameters (passed to the command, through the **GET** method), using the URL (in order to include them) will follow the session identifier, as presented below:

[http://10.10.50.14:80/BIT/HSM20/VerifyMac;jsessionid=3F21...42.worker2?\[parameters\]](http://10.10.50.14:80/BIT/HSM20/VerifyMac;jsessionid=3F21...42.worker2?[parameters])

CONFIDENCIAL

3.2. End Operation Command

It is necessary to invoke this command whenever an operation is completed.

COMMAND

[http://IP:PR/Application/Version/FinishOperation;jsessionId=\[session value\]](#)

PARAMETERS

[jsessionid](#) This is the code that identifies the operation obtained using the **InitOperation** command.

RESPONSE

CodResponse: This is the response code. The values of 0x00 and 0x20 indicate that there is no error, and that there is no session respectively.

EXAMPLE

[http://10.10.50.14:80/BIT/HSM20/FinishOperation;jsessionId=5C08523D33ADCEFE1859F4AC8AE85EB5.worker1](#)

RESULT

CodResponse = 0

3.3. Diversified Key Generation Command

This command is used in order to obtain the diversified keys of a card, with only one command call to the HSM.

COMMAND

[http://IP:PR/Application/Version/GetAllDiversifiedKey;jsessionid=\[SessionValue\]?SerialNumber=\[SerialNumber\]](http://IP:PR/Application/Version/GetAllDiversifiedKey;jsessionid=[SessionValue]?SerialNumber=[SerialNumber])

PARAMETERS

- SerialNumber** It represents the card serial number, and contains 14 characters in hexadecimal system (or 7 Bytes).
- jsessionId** This is the code that identifies the operation obtained using the **InitOperation** command.

The command returns:

- CodResponse** This is the return code. Value 0 indicates correct execution.
- KeyDiversified0, keyDiversified1, ..., keyDiversified6** It contains packs of 32 characters in a hexadecimal system representation of the diversified key, or 0 if an error occurs.

EXAMPLE

<http://10.10.50.14:80/BIT/HSM20/GetAllDiversifiedKey;jsessionId=5C08523D33ADCFE1859F4AC8AE85EB5.worker1?SerialNumber=04404040404040>

RESULT

```
CodResponse = 0
KeyDiversified0 = 49dc91c2bbd62b1e734b16669752a27d
KeyDiversified1 = 484ab101c47719ea6143bf15c010ada2
KeyDiversified2 = f42a5bdf20b56ad60817e068b9900b2c
KeyDiversified3 = 3bf93c391f3b281133041f5e2d7c8350
KeyDiversified4 = d16305c9b0f93cdf5631397149f083f7
KeyDiversified5 = 48099de638e575d0fbb7e0e7d7c72f5c
KeyDiversified6 = eb0fd27c798bccc243ecea65c738a435
```

3.4. Transaction Record Generation Command

This command generates the cryptographic checksum MAC (or digital signature) of 4 Bytes, which is finally concatenated at the end of the transaction record. The command returns the result code (correct or incorrect), the operation counter value, the transaction counter value and the cryptographic checksum.

COMMAND

http://IP:PR/Application/Version/DoMac;jsessionid=[SessionValue]?Data=[Data]&Tlv=[TLV Code]

PARAMETERS

- jsessionid** This is the code that identifies the operation obtained using the **InitOperation** command.
- Data** This parameter contains the data that will be digitally signed (expressed in hexadecimal system). This data packet does not include the [HSMNumber]. In the case of pre-personalization process, the length of this data packet is 36 Bytes. In other words, this is like the pre-personalization transaction record (of 59 Bytes), excluding (in total 8+8+3+4 Bytes): the first two counters (8+8 Bytes, that are not yet known), the HSM serial number, and of course the digital signature (the last 4 bytes, which is not yet known).
- Tlv** This parameter indicates the TLV type (in hexadecimal system) that has to be digitally signed. In the case of pre-personalization process, contains the value **0xC3**.

RESPONSE

- CodResponse** This is the response code (correct or incorrect). It contains 0 if there is no error.
- MAC** This is the cryptographic checksum (or a digital signature of 4 Bytes in hexadecimal system).
- OperationCounter** This is the operations counter of 8 Bytes in hexadecimal system.
- TransacCounter** This is the transactions counter of 8 Bytes in hexadecimal system.
- CipherString** Encrypted data (It is not used in prepersonalization). If there is not encrypted data, it returns 0x00.

EXAMPLE

http://10.10.50.14:80/BIT/HSM20/DoMac;jsessionid=1BE1ECFF36C0F4ED3D860E20FA0C5350.worker1?Data=112233445566778899001122334455667788990011223344556677889900112233445566&Tlv=C3

RESPONSE EXAMPLE

```
CodResponse = 0
Mac = 28DF953F
OperationCounter = 0000000000006070
```

```
TransacCounter = 0000000000005E43  
CipherString = 00
```

The data received can then be used in order to compose the registration of the operation, which in our example will be:

```
0000000000006070000000000005E4316E361044E16C96D1C80055200020006035A5  
A5A0186A101A7D1127A897DCE01000000001010028DF953F
```

CONFIDENCIAL

4. COMMANDS FOR THE PERSONALIZATION APPLICATION

4.1. Start Operation Command

As it is described in section 3.1.

4.2. End Operation Command

As it is described in section 3.2.

4.3. Transaction Generation Record Command

As it is described in section 3.4.

CONFIDENCIAL

4.4. Session Key Generation (Authentication) – Part 1

COMMAND DESCRIPTION

This command is used in order to apply the first part of the authentication process according to the NXP DESFire chip. The reader sends to the security element, the ciphered random number (generated by the DESFire chip).

COMMAND SYNTAX

[http://IP:PR/Application/Version/InitSession;jsessionid=\[SessionValue\]?SerialNumber=\[CardSerialNumber\]&RndB'=\[CardGeneratedRandomNumber\]&KeyIndex=\[KeyIndex\]](#)

PARAMETERS

jsessionid	This is the code that identifies the operation obtained using the InitOperation command.
SerialNumber	This is the serial number of the chip. Its size is 7 Bytes (in hexadecimal system).
RndB'	It contains the random number generated by the chip (according to the NXP DESFire authentication procedure). It occupies 8 bytes (in hexadecimal system), and it is ciphered with the key selected for the authentication process (see the next parameter).
KeyIndex	It specifies the key index or in other words the key that should be used for this authentication process.

RESPONSE

CodResponse	This is the response code (correct or incorrect). It contains 0 if there is no error.
VersionLNS	It represents version of the card black list that is used by the security element.
RndABgCif	It contains the concatenation of the ciphered random numbers selected by the card RndB (previously rotated left by 8 bits), and the reader RndA. Each number occupies 8 bytes (in hexadecimal system).
SessionKeys	This is the session key that will be used for the current authentication.

EXAMPLE

[http://10.10.50.14:80/BIT/HSM21/InitSession;jsessionid=92E467907252A6CBF71E2CEDE40A5876.worker1?SerialNumber=04444040664077&RndB'=154E8A942EDA4BBB&KeyIndex=03](#)

RESPONSE EXAMPLE

```
CodResponse = 0
VersionLNS= 0102
RndABgCif= 9e56ef204579798ad7fc4dc9b9a9ac0e
```

SessionKeys= 990a60da4dd7441c8f856686721f5bf1

NEXT STEP

If the response to this command is equals to 0, this means that the operation has been successful, and the reader has obtained the ciphered concatenated random numbers RndABgCif and the session key. In other words, the security element has generated a random number RndA and it has concatenated it with the random number generated by the card RndB' (which has already been rotated left by 8 bits). The two concatenated numbers are ciphered using 3DES. The information that the reader obtains during this operation should be saved in order to be used in the second part of the authentication procedure.

CONFIDENCIAL

4.5. Session Key Generation (Authentication) – Part 2

COMMAND DESCRIPTION

The card receives the information generated during the execution of the first part (4.4). It is possible for the DESFire to verify that the random number **RndB** received (after it is rotated left by 8 bits) is equal to the one selected during the beginning of the operation (4.4) by the card. In other words, the card knows that the other side possesses the same secret key for the operation. Next, the card decipheres the **RndA** (selected by the security element), rotates it left by 8 bits and sends it (**RndA'**) to the reader. Moreover, this data is transferred to the security element where it is processed. If at the end the security element verifies that the data just received (**RndA**, after it is rotated left by 8 bits) is equal to the number initially selected by the same element, then it is demonstrated that the card possesses the right key.

COMMAND SYNTAX

[http://IP:PR/Application/Version/InitSession;jsessionid=\[SessionValue\]?
RndA'=\[HSMGeneratedRandomNumberRotated8bits\]](http://IP:PR/Application/Version/InitSession;jsessionid=[SessionValue]?RndA'=[HSMGeneratedRandomNumberRotated8bits])

PARAMETERS

jsessionid	This is the code that identifies the operation obtained using the InitOperation command.
RndA'	It contains the random number generated by the security element rotated left by 8 bits (according to the NXP DESFire authentication procedure). It occupies 8 bytes (in hexadecimal system), and it is ciphered with the key selected for the authentication process.

RESPONSE

CodResponse This is the response code (correct or incorrect). It contains 0 if there is no error.

4.6. Sube-T Card Number Generation Command

COMMAND DESCRIPTION

This command returns the Sube-T number which is a unique identification code for each personalized card. It is the CRTM that specifies this code during the personalization process and it is completely independent from the chip UID.

COMMAND SYNTAX

[http://IP:PR/Application/Version/GetSubeTNumber;jsessionid=\[SessionValue\]](http://IP:PR/Application/Version/GetSubeTNumber;jsessionid=[SessionValue])

PARAMETERS

jsessionid This is the code that identifies the operation.

RESPONSE

CodResponse This is the response code (correct or incorrect). It contains 0 if there is no error.

SubeT SubeT number.

EXAMPLE

<http://10.10.50.14:80/BIT/HSM21/GetSubeTNumber;jsessionid=6A04AC4036F3A5662BBA4BF63F2821C2.worker2>

RESPONSE EXAMPLE

```
CodResponse = 0
SubeT= 0000000000000130
```

4.7. HSM Serial Number Supply Command

COMMAND DESCRIPTION

This command returns the HSM serial number (assigned to the HSM by the CRTM during the particularization process).

COMMAND SYNTAX

[http://IP:PR/Application/Version/GetHSMNumber;jsessionid=\[SessionValue\]](http://IP:PR/Application/Version/GetHSMNumber;jsessionid=[SessionValue])

PARAMETERS

jsessionid This is the code that identifies the operation.

RESPONSE

CodResponse This is the response code (correct or incorrect). It contains 0 if there is no error.
HSMSerial Serial number of the HSM.

EXAMPLE

<http://10.10.50.14:80/BIT/HSM21/GetHSMNumber;jsessionid=F9261965C03985490C252D4B5778F42F.worker1>

RESPONSE EXAMPLE

```
CodResponse = 0  
HSMSerial= 16e361
```

5. COMMANDS FOR THE TICKET-VENDING APPLICATION

5.1. Start Operation Command

As it is described in section 3.1.

5.2. End Operation Command

As it is described in section 3.2.

5.3. Transaction Generation Command

As it is described in section 3.4.

5.4. Session Key Generation (Authentication) – Part 1

As it is described in section 4.4.

5.5. Session Key Generation (Authentication) – Part 2

As it is described in section 4.5.

6. COMMAND EXECUTION

Every time there is an operation with a card within an application it is necessary to implement the following commands:

1. Execute command **InitOperation**.
2. Execute the commands related to security in order to implement the applications according to specifications. The latter are included in the following documents:
 - 2.1. Aplicación de Pre_personalización V2-ea02ebed-BIT-DOC-CON-TEK.doc.
 - 2.2. Aplicación de Personalización de TMI 2 V2 -96f9bcdb-BIT-DOC-CON-TEK.doc.
 - 2.3. Aplicación de Venta de Títulos para TMI 2-0d2debfc-BIT-DOC-CON-TEK.doc.
 - 2.4. Funciones Avanzadas de Aplicación de Venta de Títulos SubeT-1b0e9130-BIT-DOC-CON-CTM-AIT.doc.

As a result the commands executed, specifying details like the times and the order of execution, are all indicated in the aforementioned documents.

3. Execute command **FinishOperation**.

7. LIST OF COMMAND RESPONSE CODES

The codes that are used to represent the command responses are given below.

Table 3 Command response codes	
Response	Explanation
Correct	
00	Command executed correctly
Errors	
01	Parameter length incorrect
02	Parameter length incorrect
03	Not possible to convert parameter to bytes – Error in hexadecimal notation
04	Not possible to convert parameter to integer
06	A command parameter is missing
20	Session identification not valid – Check jsessionid value
30	Card black list not available
61	Command executed correctly, but the HSM returns a quota different from the requested
62	Command executed correctly, but a very high quota has been requested. The HSM limits the quota to the maximum limit allowed
70	HSM expired
72	Not possible to assign a value to SubeT counter
73	SAM included in black list
74	Error in TLV length
81	Action 01 form card included in black list
82	Action 02 form card included in black list
83	Action 03 form card included in black list
84	Action 04 form card included in black list
85	Action 05 form card included in black list
86	Action 06 form card included in black list
500	Internal application error
Warning	
60	Command executed correctly. Upper limit of command SubeT Counter is reached or exceeded (although the maximum still has not been reached)
75	The authentication process (during a quote request) has failed

8. IMPORTANT COMMENTS

In case of an interrupt during an HSM command execution (after a session code has been obtained), the load-balancer will automatically assign other HSM for the operation. However, the remotely connected terminal will not be able to continue with the same session code (it will receive the return code 0x20). In this case, it will have to start a new session, and as a result, a new session code will be generated.

CONFIDENCIAL