https://verifone.cloud/docs/sca-functional-specification/payment_func/adminstration/message_auth
Updated: 20-May-2025

Message Authentication

This section describes the process of establishing a secured channel between POS and the Device. It describes the security protocol packet exchanges to establish the secured channel between POS and the Verifone device. It is during this process the key exchange happens and the device is securely paired with the POS

Pairing and Key Exchange

The first communication between a point of sale application and a device is a pairing that will also serve as a key exchange. The purpose of this pairing is to establish a level of trust that transaction requests are coming from a POS system that is authorized to send transaction requests to the Verifone platform. It is not meant to protect cardholder data, as no cardholder data is ever captured by the POS system. This pairing is only meant to reduce fraud risk associated with a rogue POS system in a merchant environment.

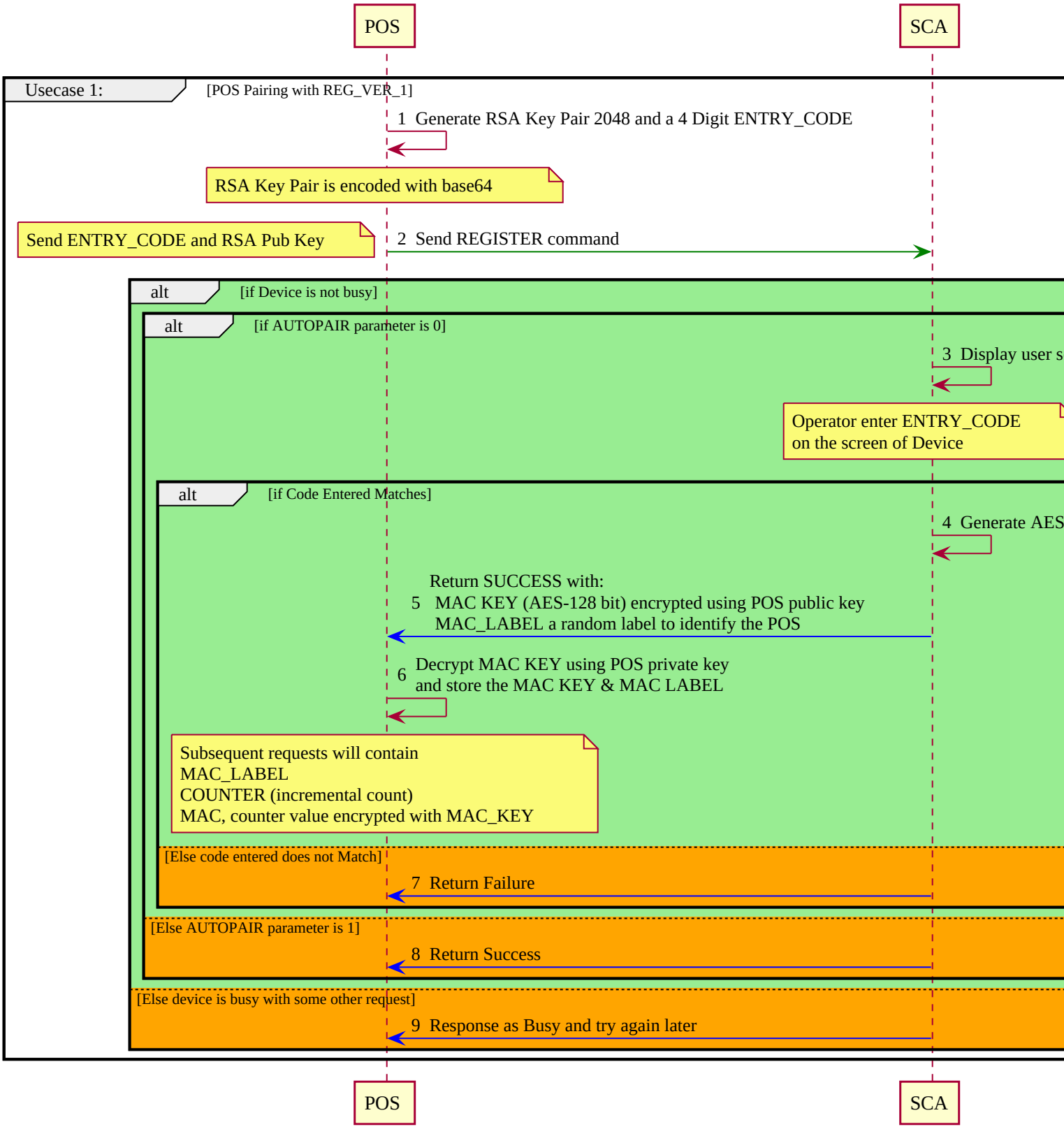
The pairing of the POS device is initiated through a REGISTER command.
Refer to the sequence diagram below for each security level. These diagrams describe the steps involved in initiating a secured channel.
After this process is complete, the device and the POS are paired and the POS can successfully send the messages to the payment device.

Following are the types of authentication based on the security level

- REG_VER 1
- REG_VER 2
- Full Packet Encryption

REG_VER 1

Following diagram explains the steps involved in establishing the secured channel for REG_VER 1 security level



1. The POS has to generate the following parameters
    - RSA key pair of 2048 bit length. These keys need to be stored internally and will be used for further transactions. The key pairs are normally generated in binary format and have to be converted to base 64. RSA key pair needs to be in base 64 format as it needs to be sent to the SCA device as part of the protocol message.

    - 4 digit ENTRY_CODE. This is a pairing code which is sent to the device and also the user has to manually enter the code on the device.
2. Send a REGISTER message with ENTRY_CODE. Refer to the protocol specification document for details on the REGISTER message.
3. SCA displays a user screen prompting for the 4 digit code to be entered manually by the operator.
4. SCA generates a AES-128 bit MAC key and encrypts the MAC key with the public key sent by the POS
5. Returns the response with following parameters
    - Encrypted MAC Key
    - A randomly generated Mac label to identify the POS
6. POS receives the response, decrypts the MAC Key with the private key component and stores the Mac Key and Mac label for future transactions
7. A failure case if the operator enters a wrong code
8. An AUTOPAIR parameter determines if the operator has to enter to code or the pairing is automatically done
9. A failure if the device is processing some other command

Example Request For REV_VER_1

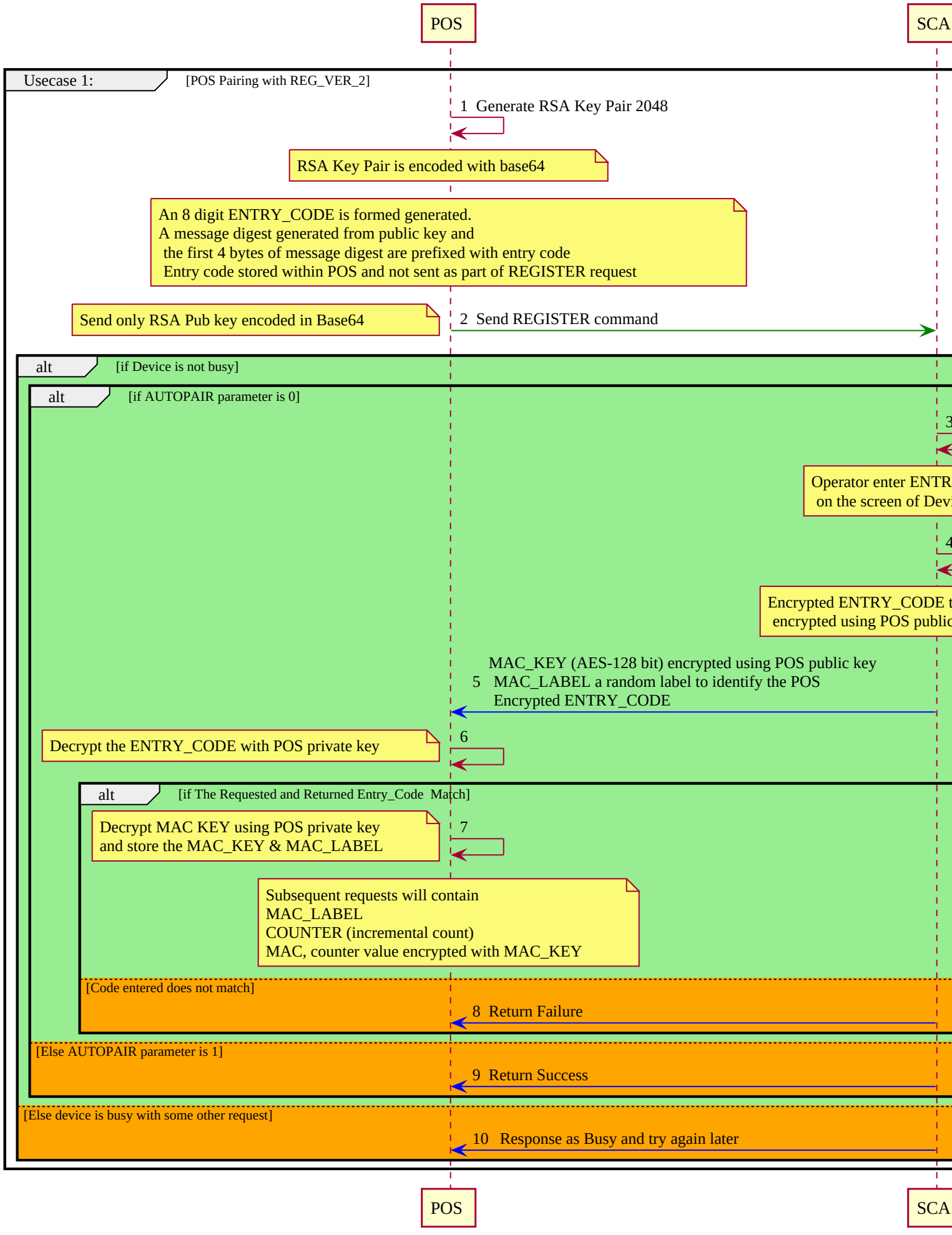
```
<TRANSACTION>
    <FUNCTION_TYPE>SECURITY</FUNCTION_TYPE>
    <COMMAND>REGISTER</COMMAND>
    <ENTRY_CODE>2884</ENTRY_CODE>
    <KEY>MIIBIjANBg....V9QIDAQAB</KEY>
</TRANSACTION>
```

Example Response For REV_VER_1

```
<RESPONSE>
    <RESPONSE_TEXT>Registered P_V58BIL</RESPONSE_TEXT>
    <RESULT>OK</RESULT>
    <RESULT_CODE>-1</RESULT_CODE>
    <TERMINATION_STATUS>SUCCESS</TERMINATION_STATUS>
    <MAC_KEY>VrRC33G4....f1qEZw==</MAC_KEY>
    <MAC_LABEL>P_V58BIL</MAC_LABEL>
    <ENTRY_CODE>2884</ENTRY_CODE>
</RESPONSE>
```

REG_VER 2

Following diagram explains the steps involved in establishing the secured channel for REG_VER 2 security level.


POS
SCA
Usecase 1:
[POS Pairing with REG_VER_2]
1 Generate RSA Key Pair 2048
RSA Key Pair is encoded with base64
An 8 digit ENTRY_CODE is formed generated.
A message digest generated from public key and
the first 4 bytes of message digest are prefixed with entry code
Entry code stored within POS and not sent as part of REGISTER request
Send only RSA Pub key encoded in Base64
2 Send REGISTER command
alt
[if Device is not busy]
alt
[if AUTOPAIR parameter is 0]
3
Operator enter ENTR
on the screen of Dev
4
Encrypted ENTRY_CODE t
encrypted using POS public
MAC_KEY (AES-128 bit) encrypted using POS public key
5 MAC_LABEL a random label to identify the POS
Encrypted ENTRY_CODE
Decrypt the ENTRY_CODE with POS private key
6
alt
[if The Requested and Returned Entry_Code Match]
Decrypt MAC KEY using POS private key
and store the MAC_KEY & MAC_LABEL
7
Subsequent requests will contain
MAC_LABEL
COUNTER (incremental count)
MAC, counter value encrypted with MAC_KEY
[Code entered does not match]
8 Return Failure
[Else AUTOPAIR parameter is 1]
9 Return Success
[Else device is busy with some other request]
10 Response as Busy and try again later
POS
SCA

1. The POS has to generate the following parameters
    - RSA key pair of 2048 bit length. These keys need to be stored internally and will be used for further transactions. The key pairs are normally generated in binary format and have to be converted to base 64. RSA key pair needs to be in base 64 format as it needs to be sent to the SCA device as part of the protocol message.
    - 8 digit ENTRY_CODE. This is a pairing code which is sent to the device and also the user has to manually enter the code on the device.
2. Send a REGISTER message **without ENTRY_CODE**. Refer to the protocol specification document for details on the REGISTER message.
3. SCA display a user screen prompting for the 4 digit code to be entered manually by the operator.
4. SCA generates a AES-128 bit MAC key and encrypts the MAC key with the public key sent by the POS
5. Returns the response with following parameters
    - Encrypted MAC Key
    - A randomly generated Mac label to identify the POS
    - ENTRY_CODE entered by the user encrypted using POS public key
6. POS receives the response, decrypts the ENTRY_CODE with the private key component. Compare the ENTRY_CODE with the code randomly generated by POS
7. Here the ENTRY_CODE entered matches the randomly generated code by POS. Hence decrypt the MAC KEY using POS private key and store the MAC KEY and MAC LABEL.
8. The code entered by the user on SCA does not match the randomly generated code by POS. Hence return failure.
9. If AUTOPAIR parameter was set to 1, then immediately return success without asking user to enter the pairing code.
10. If the device is busy processing some other requests, then SCA returns a busy try again later response.

Example Request For REV_VER_2

```
<TRANSACTION>
    <FUNCTION_TYPE>SECURITY</FUNCTION_TYPE>
    <COMMAND>REGISTER</COMMAND>
    <REG_VER>2.0</REG_VER>
    <KEY>MIIBIjANBg....UTQIDAQAB</KEY>
</TRANSACTION>
```

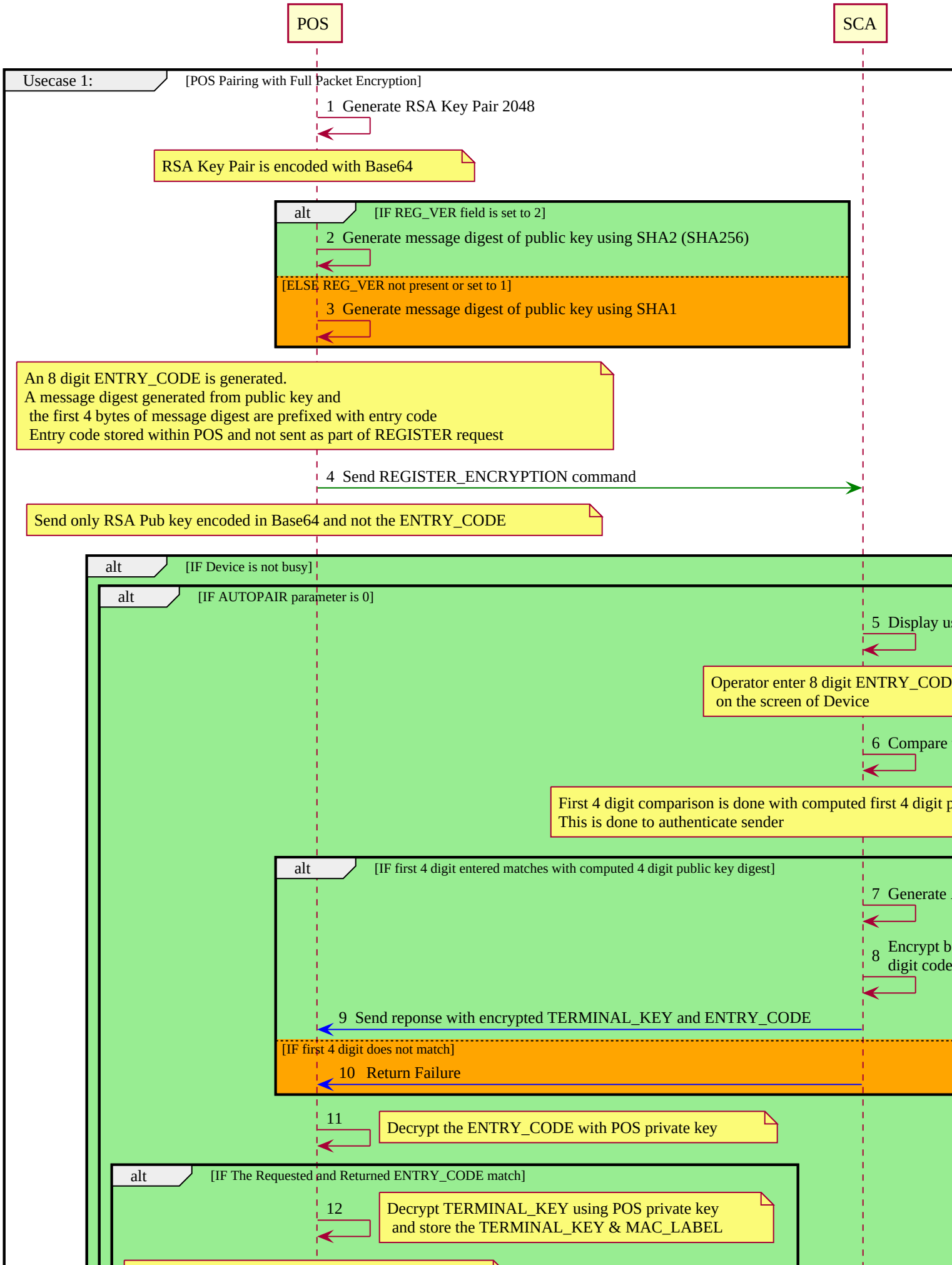
Example Response For REV_VER_2

```
<RESPONSE>
    <RESPONSE_TEXT>Registered P_V7NUMC</RESPONSE_TEXT>
    <RESULT>OK</RESULT>
    <RESULT_CODE>-1</RESULT_CODE>
    <TERMINATION_STATUS>SUCCESS</TERMINATION_STATUS>
    <MAC_KEY>K8+71HyZF....MwYg==</MAC_KEY>
    <MAC_LABEL>P_V7NUMC</MAC_LABEL>
    <ENTRY_CODE>ceFQUY5....SDFxGA==</ENTRY_CODE>
</RESPONSE>
```

Full Packet Encryption

An available option to integrators is the ability to exchange an AES encryption key and encrypt the SCA transaction data between POS and Secure Commerce Application on the device. The solution models much of the key exchange process outlined above, but contains better authentication and security surrounding the exchange process.

Following diagram explains the steps invovled in establishing the secured channel for Full Packet Encryption security level


POS
SCA
Usecase 1:
[POS Pairing with Full Packet Encryption]
1 Generate RSA Key Pair 2048
RSA Key Pair is encoded with Base64
alt
[IF REG_VER field is set to 2]
2 Generate message digest of public key using SHA2 (SHA256)
[ELSE REG_VER not present or set to 1]
3 Generate message digest of public key using SHA1
An 8 digit ENTRY_CODE is generated.
A message digest generated from public key and
the first 4 bytes of message digest are prefixed with entry code
Entry code stored within POS and not sent as part of REGISTER request
4 Send REGISTER_ENCRYPTION command
Send only RSA Pub key encoded in Base64 and not the ENTRY_CODE
alt
[IF Device is not busy]
alt
[IF AUTOPAIR parameter is 0]
5 Display us
Operator enter 8 digit ENTRY_COD
on the screen of Device
6 Compare
First 4 digit comparison is done with computed first 4 digit p
This is done to authenticate sender
alt
[IF first 4 digit entered matches with computed 4 digit public key digest]
7 Generate
8 Encrypt b
digit code
9 Send reponse with encrypted TERMINAL_KEY and ENTRY_CODE
[IF first 4 digit does not match]
10 Return Failure
11
Decrypt the ENTRY_CODE with POS private key
alt
[IF The Requested and Returned ENTRY_CODE match]
12
Decrypt TERMINAL_KEY using POS private key
and store the TERMINAL_KEY & MAC_LABEL

1. The POS has to generate the following parameters
    - RSA key pair of 2048 bit length. These keys need to be stored internally and will be used for further transactions. The key pairs are normally generated in binary format and have to be converted to base 64. RSA key pair needs to be in base 64 format as it needs to be sent to the SCA device as part of the protocol message.
    - 8 digit ENTRY_CODE. This is a pairing code which is sent to the device and also the user has to manually enter the code on the device.
2. Generate message digest of public key using SHA2 (SHA256) if REG_VER field is to be set to 2 in the protocol packet
3. Generate message digest of public key using SHA1 if REG_VER field is to be set to 1 in the protocol packet
4. Send a REGISTER_ENCRYPTION message **without ENTRY_CODE**. Refer to the protocol specification document for details on the REGISTER_ENCRYPTION message.
5. SCA display a user screen prompting for the 8 digit code to be entered manually by the operator.
6. SCA validates the first 4 digits entered by the user. Compares it against the 4 bytes of message digest of public key. If they match, the sender is authenticated.
7. SCA generates a AES-128 bit TERMINAL_KEY
8. Encrypts both the TERMINAL_KEY and 8 digit code with the public key sent by the POS
9. Returns the response with following parameters
    - TERMINAL_KEY
    - A randomly generated Mac label to identify the POS
    - ENTRY_CODE entered by the user encrypted using POS public key
10. If the 4 digit entered by user does not match the 4 bytes of message digest of public key, then return failure.
11. POS receives the response, decrypts the ENTRY_CODE with the POS private key and compares against randomly generated 4 digit in step 1.
12. If the 4 digit ENTRY_CODE matches, then POS decrypts the TERMINAL_KEY with the private key component. Stores TERMINAL_KEY and MAC_LABEL
13. The code entered by the user on SCA does not match the randomly generated code by POS, then show pairing failure.
14. If AUTOPAIR parameter was set to 1, then immediately return success without asking user to enter the pairing code.
15. If the device is busy processing some other requests, then SCA returns a busy try again later response.

Note

It is recommended to use POS Registration with FULL Encryption for Credit card transactions, where Account Number is sent from POS to ensure that the card details are encoded and is not compromised. Below is the example of Account Passthrough Transaction:

-

```
    <TRANSACTION>
    <FUNCTION_TYPE>PAYMENT</FUNCTION_TYPE>
    <COMMAND>CAPTURE</COMMAND>
    <TRANS_AMOUNT>51.00</TRANS_AMOUNT>
    <MANUAL_ENTRY>FALSE</MANUAL_ENTRY>
    <PAYMENT_TYPE>CREDIT</PAYMENT_TYPE>
    <FORCE_FLAG>FALSE</FORCE_FLAG>
    <ACCT_NUM>4242424242424242</ACCT_NUM>
    </TRANSACTION>
```

Example Request For Full Packet Encryption

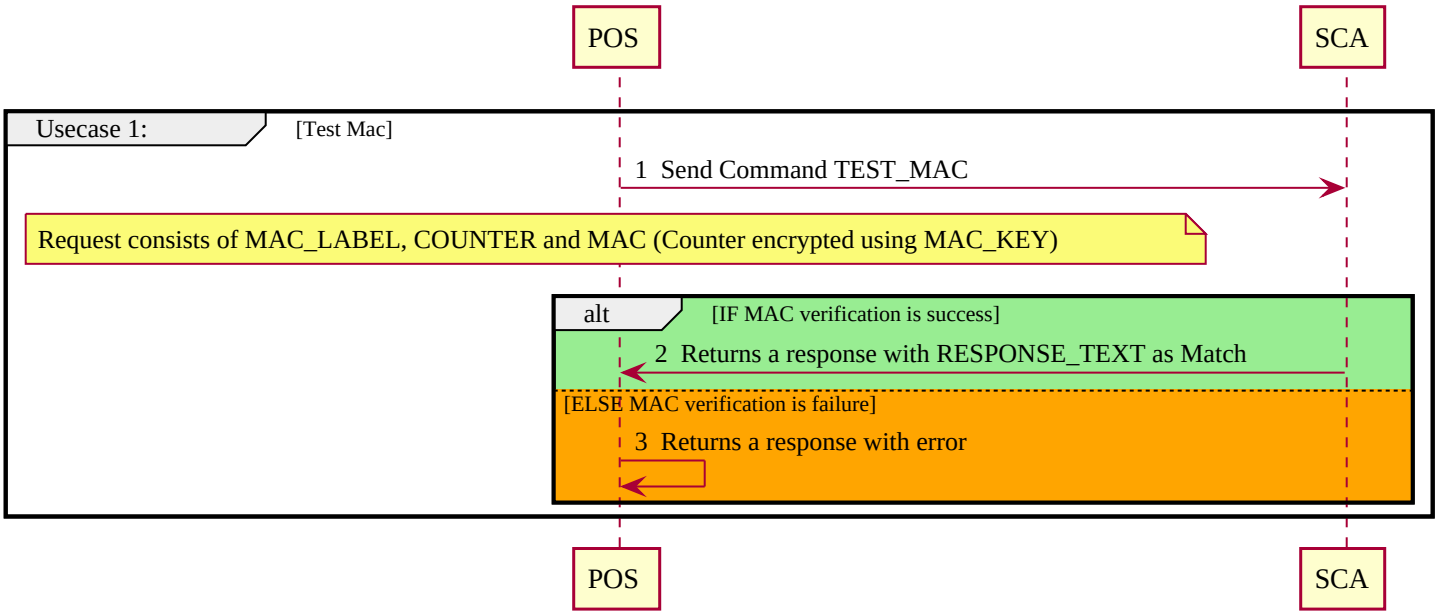
```
<TRANSACTION>
    <FUNCTION_TYPE>SECURITY</FUNCTION_TYPE>
    <COMMAND>REGISTER_ENCRYPTION</COMMAND>
    <KEY>MIIBIjANBg....FmQIDAQAB</KEY>
</TRANSACTION>
```

Example Response For Full Packet Encryption

```
<RESPONSE>
    <RESPONSE_TEXT>Registered P_XTZF6Y</RESPONSE_TEXT>
    <RESULT>OK</RESULT>
    <RESULT_CODE>-1</RESULT_CODE>
    <TERMINATION_STATUS>SUCCESS</TERMINATION_STATUS>
    <MAC_LABEL>P_XTZF6Y</MAC_LABEL>
    <ENTRY_CODE>A5/DryKzi....WDRWoZmJRhgw==</ENTRY_CODE>
    <TERMINAL_KEY>t4lEl5F....uAfhjzrU/MFA==</TERMINAL_KEY>
</RESPONSE>
```

TEST MAC

The command validates that a MAC is correct and that a POS has be successfully added. Performing this command will increate the counter to the next possible value.



Example Request For TEST_MAC

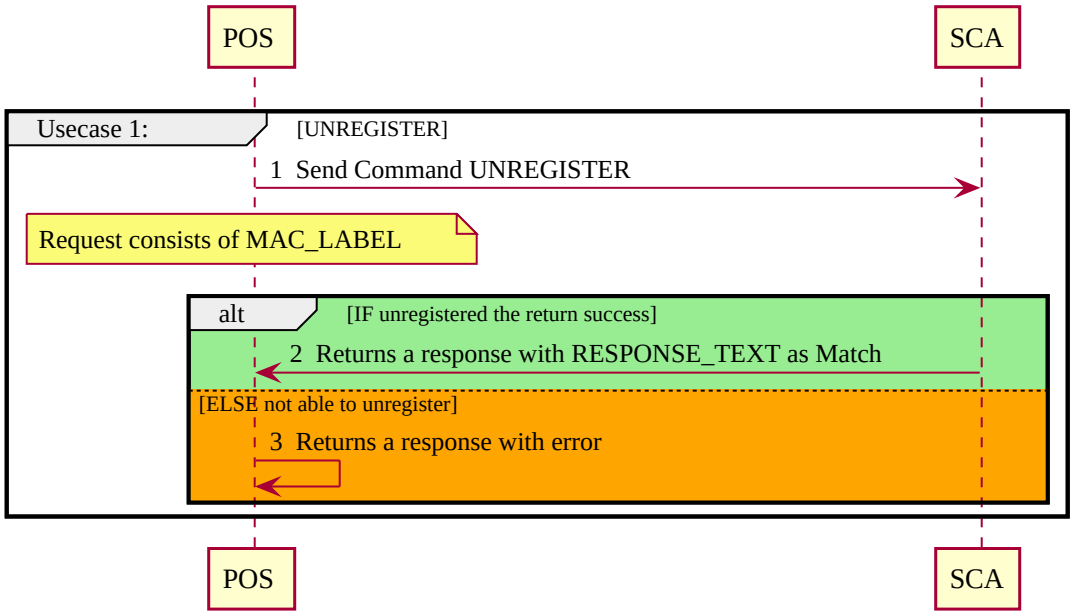
```
<TRANSACTION>
    <FUNCTION_TYPE>SECURITY</FUNCTION_TYPE>
    <COMMAND>TEST_MAC</COMMAND>
    <MAC_LABEL>P_94G1J1</MAC_LABEL>
    <COUNTER>2</COUNTER>
    <MAC>knKTLqMNKdhEqQxn1wJO5NcMc4xsME6THtMB89HOT2U=</MAC>
</TRANSACTION>
```

Example Response For TEST_MAC

```
<RESPONSE>
    <RESPONSE_TEXT>Match</RESPONSE_TEXT>
    <RESULT>OK</RESULT>
    <RESULT_CODE>-1</RESULT_CODE>
    <TERMINATION_STATUS>SUCCESS</TERMINATION_STATUS>
    <COUNTER>2</COUNTER>
</RESPONSE>
```

UNREGISTER

This command removes the POS registration from device's trusted clients list and removes the data used to verify authenticity of the sender.



Example Request For TEST_MAC

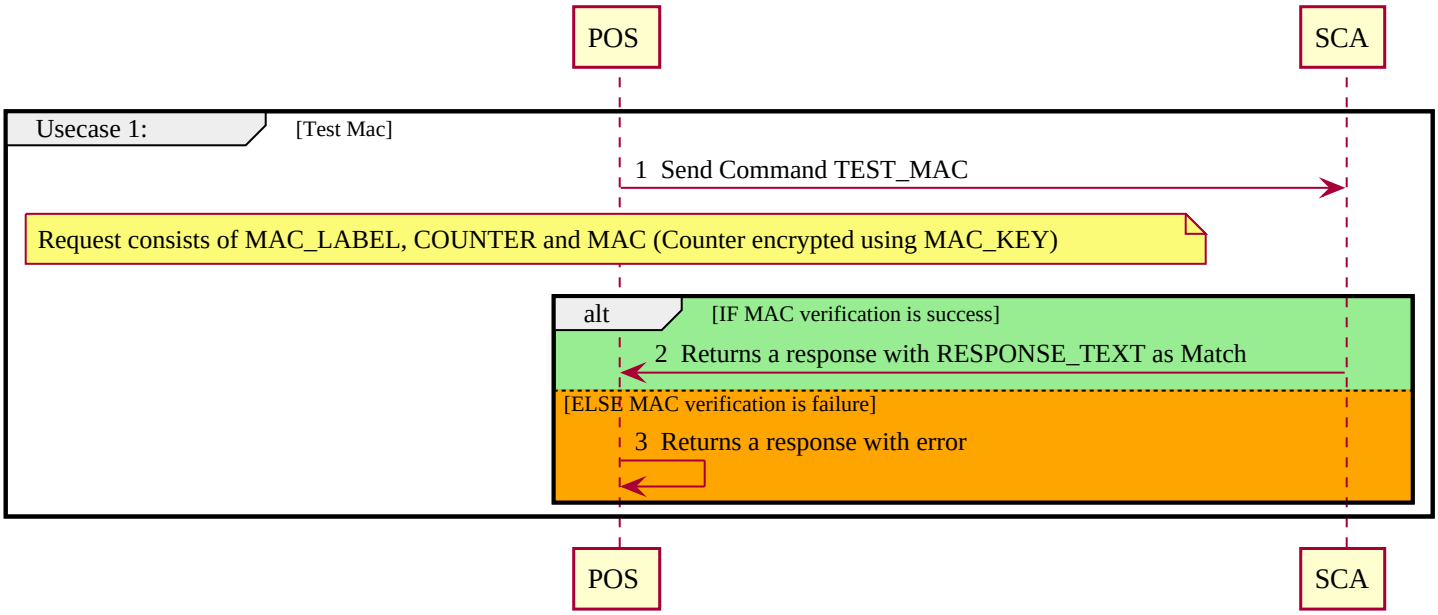
```
<TRANSACTION>
    <FUNCTION_TYPE>SECURITY</FUNCTION_TYPE>
    <COMMAND>UNREGISTER</COMMAND>
    <MAC_LABEL>P_A01W7B</MAC_LABEL>
</TRANSACTION>
```

Example Response For TEST_MAC

```
<RESPONSE>
    <RESPONSE_TEXT>Unregistered P_A01W7B</RESPONSE_TEXT>
    <RESULT>OK</RESULT>
    <RESULT_CODE>-1</RESULT_CODE>
    <TERMINATION_STATUS>SUCCESS</TERMINATION_STATUS>
    <MAC_LABEL>P_A01W7B</MAC_LABEL>
</RESPONSE>
```

UNREGISTERALL

This command removes the POS registration from device’s trusted clients list and removes the data used to verify authenticity of the sender.



Example Request For TEST_MAC

```
<TRANSACTION>
    <FUNCTION_TYPE>SECURITY</FUNCTION_TYPE>
    <COMMAND>UNREGISTERALL</COMMAND>
</TRANSACTION>
```

Example Response For TEST_MAC

```
<RESPONSE>
    <RESPONSE_TEXT>Operation SUCCESSFUL</RESPONSE_TEXT>
    <RESULT>OK</RESULT>
    <RESULT_CODE>-1</RESULT_CODE>
    <TERMINATION_STATUS>SUCCESS</TERMINATION_STATUS>
</RESPONSE>
```

Message Authentication Code (MAC)

SCA uses a Hash-based message authentication code (HMAC) to verify message authenticity. The hashing algorithm is SHA-256.

Example 1 (Empty)
MAC_KEY: (empty)
COUNTER: (empty)
MAC: 0xb613679a0814d9ec772f95d778c35fc5ff1697c493715653c6c712144292c5ad

Example 2
MAC_KEY (not encrypted, base 64 encoded): /K33o/TkT8g4093cUw1K+w==
COUNTER: 1

MAC (base 64 encoded): liGpuYqVWmh+AQ/wmI2tQE03ImSIeCbYH1rrHshZMv8=

Rules

1. If a device receives a pairing request and it is currently processing some other request of any kind, it will respond to the POS with a message indicating that it is busy and to try again later.
2. If the device is moved to a different lane (physical or virtual) and needs to perform a new pairing, the POS will simply perform the pairing process again using the REGISTER command.
3. If the POS loses any of its key material, it can perform the pairing process again using the REGISTER command. Best practice: perform an UNREGISTER command before re-registering.
4. The POS will need to increment a counter (COUNTER) for each transaction; this could also be used as a transaction identifier if necessary.
5. An un-pairing message is available that removes knowledge of the POS from a particular device (UNREGISTER).
6. It is not necessary to re-pair following a normal POS reboot.
7. Subsequent message between the POS and the payment device will contain all the required data elements to perform a transaction, plus it must also include the counter and a MAC of the counter.
8. The payment device does not need to track counters, per se, but will validate that the counter value is greater than the previous counter.
9. The counter will be MACéd by the POS so that the payment device can validate each MAC for each message.

Configuration Parameter

Following configuration parameter which affect the operation. Refer to Application Parameters table for more details on the below parameter.

- AUTOPAIR

Message Interfaces

Refer to the below Protocol sections for the command fields description and examples.

- REGISTER
- REGISTER_ENCRYPTION
- UNREGISTER
- UNREGISTERALL
- TESTMAC