**Demo Applet**

Specification

Contents

[1 Introduction 3](#_Toc387765699)

[2 Services 4](#_Toc387765700)

[2.1 Authenticate Service 4](#_Toc387765701)

[2.2 Update Demo Key Service 4](#_Toc387765702)

[2.2.1 Update DEM-AUTH Key Container 5](#_Toc387765703)

[2.2.2 Update DEM-MAC Key Container 5](#_Toc387765704)

[2.2.3 Update DEM-KEY-WRAP Container 5](#_Toc387765705)

[2.2.4 Update DEM-3P-PUB key containers 5](#_Toc387765706)

[2.3 GET Challenge 6](#_Toc387765707)

[2.4 Key Pair Generation 7](#_Toc387765708)

[2.5 GET DATA 7](#_Toc387765709)

[2.6 Manage Security Environment (MSE) 8](#_Toc387765710)

[2.6.1 Restore 8](#_Toc387765711)

[2.6.1 SET 9](#_Toc387765712)

[2.7 Perform Security Operation (PSO) 9](#_Toc387765713)

[2.7.1 WRAP 9](#_Toc387765714)

[2.7.2 UNWRAP 9](#_Toc387765715)

[2.7.3 Digital Signature 9](#_Toc387765716)

[2.7.4 HASH 9](#_Toc387765717)

[2.7.5 Verify Digital Signature 9](#_Toc387765718)

[2.8 Key Agreement 10](#_Toc387765719)

[2.9 Destroy 10](#_Toc387765720)

# 1 Introduction

This document describes all of the services in Demo Applet v1.0. The services have some dependency on each other and these dependencies also describe here.

The command set of the applet is based on ISO/IEC 7816-4, ISO/IEC 7816-8, and some proprietary command according to these standards.

The applet uses Java Card v3.0.4 and Global Platform v2.2.1 APIs as the development platform.

The main purpose of this Applet is to perform many of the basic crypto operations such as Wrap/Un-wrap, Signature/Verify, Hash and Mac.

The applet supports five different types of keys: RSA, DES, AES, EC and HMAC keys of different length and corresponding many algorithms discussed later in the documentation.

The applet can contain several objects as its internal resources: Key-Refs, Cipher-Refs, Digest-Refs, Signature-Refs and Key-containers. This Key-container keeps all of the public and private keys with their basic information like, key type, length etc. This objects references will use frequently.

The applet has its security policy in many terms. The first level security is to External authenticate. The next level is operation sate. For all operations, the state is changed and some operation is dependent on other operations state. And finally, the applet state: (probably) INSTALLED, INITIALIZED, PERSONALIZED, BLOCKED and DEAD.

The applet has many services:

1. Authentication.
2. Update Demo Keys.
3. Random Number Generation.
4. Key Pair Generation.
5. Digital Signature.
6. Message Authentication.
7. Message Digest.
8. Key Wrap.
9. Key Agreement.
10. Destroy Demo Applet CSPs.

# 2 Services

The Demo Applet can perform many services. These services are described below.

# 2.1 Authenticate Service

To perform any operation in Demo Applet, the applet user needs to authenticate first. Demo Applet use SCP-02 for external authentication. Initially the base key for external authentication is 16 byte constant value. But applet user can use own base key (16 byte) by Update Demo Key service (will discuss later).

The authentication service has two parts:

1. Initialize update.
2. External authenticate.

See Demo Applet APDU specification for more details.

# 2.2 Update Demo Keys Service

This service is for put data into key containers. Demo Applet has four Key containers to contain different types of keys:

1. DEM-AUTH key containers.
2. DEM-MAC key containers.
3. DEM-KEY-WRAP key containers.
4. DEM-3P-PUB key containers.

Update demo key service is for put keys into these Key Containers. Demo Applet supports different types of keys: RSA Key Pair, EC Key Pair, DES Keys, AES Keys, HMAC keys.

To perform different types of crypto operation, the appropriate key will be put into these key containers according to this service (for key pair, only the third party public key can be put into key containers). See Demo Applet APDU specification for the APDU structure of PUT KEY service.

# 2.2.1 Update DEM-AUTH Key Container

To perform Authentication service (see section: 2.1 Authenticate Service) the base key can be put into the DEM-AUTH key container through this service. This base key will be used to generate internal keys for Initialize update and External authenticate.

The base key is Two Key Triple DES (16 byte).

# 2.2.2 Update DEM-MAC Key Container

To generate and verify MAC (See section: 2.7.3 Digital Signature), the MAC key will be put into DEM-MAC key container. The type of MAC keys are:

1. DES key (16 bytes).
2. AES-128 key (16 bytes).
3. HMAC key (Not more than 64 bytes).

# 2.2.3 Update DEM-KEY-WRAP Container

To perform wrap/unwrap (See section: 2.7.1 WRAP, 2.7.2 UNWRAP) of any plain texts or keys the wrap/unwrap key will be put into DEM-KEY-WRAP Container. The type of wrap/unwrap keys are:

1. DES key (16 or 24 bytes).
2. AES key (16, 24 or 32 bytes).

# 2.2.4 Update DEM-3P-PUB key containers

For Digital signature verification (See section: 2.7.3 Digital Signature) and Key Agreement (See section: 2.8 Key Agreement) service, the public key will be put into DEM-3P-PUB key containers. Two types of public key can be put into the containers:

1. RSA public key (both 1024 and 2048 bits).
2. EC public key (all EC keys approved by NIST and FIPS).

**Put RSA public key:**

To put RSA-1024 public key two APDU command is needed.

* One’s for putting modulus of the keys.
* And the other for exponent!

But in RSA-2048, three APDU command is needed.

* First one is for putting the first part of modulus of the keys (1st 128 bytes) staring with a tag “81”.
* The second one is for putting the last part of modulus of the keys (2nd 128 bytes) staring with a tag “82”..
* And the last APDU is for exponent!

In every case the modulus of the public key will be put first into the key container than the exponent. User cannot put exponent first if the modulus is not already put into the key container.

For RSA-2048, 2nd part of the modulus cannot be put into the key container if the 1st part of the modulus is not put yet into the key container.

The public key cannot be initialized properly if the exponent is not put into the key container. Hence no crypto operation can be performed by this public key!

There is a sample APDU for Update DEM-AUTH Key Container:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CLA | INS | P1 | P2 | LC | Data | LE |
| 00 | DA | 01 | 01 | 10 | 16 byte DES key | -- |

For more understanding about APDU structure, see Demo Applet APDU specification.

**Some restriction:**

* AUTH key must be 16 bytes.
* HMAC key must not be more than 64 bytes.
* For Wrap/Unwrap service, DES key must be 16 or 24 bytes.
* For Wrap/Unwrap, if NOPAD algorithm is used (such as: ALG\_DES\_CBC\_NOPAD), then the input message must be explicitly padded according to multiple of the key length. For example: if 16 bytes DES key with NOPAD algorithm (ALG\_DES\_CBC\_NOPAD) is used, then the plain text for wrap/unwrap must be multiple of 16 bytes (explicit padding is needed). Otherwise 6F00 status word will be thrown.

# 2.3 Random Number Generation

This service is for generating random number of given size. To generate random number, the Get Challenge service has two ways:

1. Generate random number with seeding
2. Generate random number without seeding.

In generating random number with seeding the seed value is required and its minimum size is 8 (eight) bytes.

Sample APDU for generating random number with 8 bytes seeding:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CLA | INS | P1 | P2 | LC | Data | LE |
| 00 | 84 | 00 | 00 | 0A | 08 20 eight bytes seed value | -- |

# 

# 2.4 Key Pair Generation

Key Pair Generation service generates two types of key pair:

1. RSA key pair.
2. EC key pair.

**RSA key pair:**

Key Pair Generation service only generates RSA-2048 key pair.

**EC key pair:**

Demo Applet supports all the NIST and FIPS approved EC key pair generation:

1. EC P-224.
2. EC P-256.
3. EC P-384.
4. EC P-521.

This Applet preserves the generated key pair into its internal key objects for further use until the Destroy (See section: 2.9 Destroy) service is called or a new key pair is generated! When a new key pair is generated the previous key pair is rewrite with the new key pair. After the key pair is generated, only the public key can be retrieved by GET DATA APDU command.

Sample APDU for generating EC-521 key pair:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CLA | INS | P1 | P2 | LC | Data | LE |
| 00 | 46 | 00 | 00 | 01 | 15 | -- |

# 2.5 GET DATA

GET DATA service is to retrieve the public key from the generated key pair (See section: 2.4 Key Pair Generation). Only the public key is retrievable from the key pair.

GET DATA service retrieve public key in two ways:

1. To retrieve RSA public key, one APDU command’s response will return the public key modulus (1024 or 2048bits). And other APDU command’s response will return the public key exponent.
2. To retrieve EC public key one APDU command’s response will return the EC public key.

**Public key format:**

* APDU command to retrieve modulus of RSA public key returns the modulus value of the key in plain text. The data format is big-endian and right-aligned (the least significant bit is the least significant bit of last byte).
* APDU command to retrieve exponent of RSA public key returns the public exponent value of the key in plain text. The data format is big-endian and right-aligned (the least significant bit is the least significant bit of last byte).
* APDU command to retrieve EC public key returns the point of the EC curve comprising the public key in plain text form. The point is represented as an octet string in compressed or uncompressed forms as per ANSI X9.62. The data format is big-endian and right-aligned (the least significant bit is the least significant bit of last byte).

Sample APDU for getting the EC public key:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CLA | INS | P1 | P2 | LC | Data | LE |
| 00 | CA | 01 | 60 | -- | -- | -- |

# 2.6 Manage Security Environment (MSE)

To perform any security operation appropriate algorithm is needed. Security Environment (SE) keeps the information of algorithm. MSE service will set the SE with user defined algorithm. See Demo Applet APDU specification for algorithm lists. Then the cryptographic operation will be done based on the algorithm settled in SE.

MSE has two main parts:k

1. Restore SE
2. SET SE

# 2.6.1 Restore

In a session, the cards SE must be restored first at least one time. Without restoring the SE, one cannot SET it!

In restore SE process, some predefined algorithm will be set for different cryptographic operation.

* Security Environment for *HASH(Message Digest)* operation will be restored with algorithm *ALG\_SHA\_256*
* Security Environment for *CONFIDENTIALITY (wrap/unwrap)* operation will be restored with algorithm *ALG\_AES\_BLOCK\_128\_CBC\_NOPAD*
* Security Environment for *DS (Digital Signature and MAC)* operation will be restored with algorithm *ALG\_AES\_MAC\_128\_NOPAD*.

# 2.6.1 SET

Applet user can SET SE with any preferred algorithm for appropriate security operation. But SE must be restored once in a session before set. If the SE is not resorted at least once in a session, it cannot be SET!

To perform Wrap operation SE need to be SET for Confidentiality operation with appropriate algorithm. See Demo Applet APDU specification for algorithm list.

Whenever a security operation is going to perform, first, the SE for that operation is checked for proper algorithm. According to the algorithm found in SE, appropriate Key container is searched to initialize the Key.

For example, if SE is set with algorithm ALG\_DES\_CBC\_PKCS5 for Confidentiality (wrap/unwrap) operation, then, to initialize appropriate Key for wrap/unwrap operation, the DEM-KEY-WRAP key containers will be searched for key data. If DEM-KEY-WRAP key containers are not initialized with DES key, then Key initialization will be failed.

See section 2.2 Update Demo Keys Service for which Key containers will be searched for Key data to perform what kind of security operation.

So, if applet user updates DEM-3P-PUB key containers with RSA-1024 public key to perform DS verification operation (See section: 2.7.5 Verify Digital Signature) but do not SET SE with appropriate algorithm for DS operation then unexpected behavior may be found!

Sample APDU for setting SE with ALG\_HMAC\_SHA1 for DS operation:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CLA | INS | P1 | P2 | LC | Data | LE |
| 00 | 22 | F1 | B6 | 03 | 8001 0C | -- |

# 2.7 Perform Security Operation (PSO)

All kind of security operation will be performed with Perform Security Operation (PSO) service. But to use this service SE must be Restored or Set first. So without MSE, PSO service cannot be performed. The PSO service mainly has the following parts:

1. Digital Signature.
2. Message Authentication.
3. Message Digest.
4. Key Wrap.

Description of these services is given below.

# 2.7.1 Digital Signature

Digital Signature (DS) service mainly does two things:

1. Signature generation.
2. Signature verification.

**Signature generation:**

First the applet user Restore or SET the SE for DS operation with appropriate algorithm. Then, the Key for Signature generation will be initialized according to the algorithm in SE.

/\* If the algorithm is for DES, AES or HMAC then the DEM-MAC key containers will be searched for key data to initialize the Signature Key. So Update Demo Keys Service must be performed before DS service. \*/

If the algorithm is for RSA or EC then the Signature Key will be initialized from the corresponding RSA or EC private key, generated in Key Pair Generation service (See section: ). So Key Pair Generation service must be performed before DS service. Otherwise unexpected behavior may be found.

The highest length of the data to be signed is 255 bytes.

Sample APDU for generating DS:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CLA | INS | P1 | P2 | LC | Data | LE |
| 00 | 2A | 9E | 9A | Data length | Data to be signed | -- |

**Signature verification:**

First the applet user Restore or SET the SE for DS operation with appropriate algorithm. Then, the Key for Signature generation will be initialized according to the algorithm in SE.

/\*If the algorithm is for DES, AES or HMAC then the DEM-MAC key containers will be searched for key data to initialize the Signature Key.\*/

If the algorithm is for RSA or EC then the DEM-3P-PUB key containers will be searched for key data to initialize the Signature Key with corresponding key type: RSA public key or EC public key.

So Update Demo Keys Service must be performed before DS service. Otherwise unexpected behavior may be found.

To verify a signed data, multiple APDU command is needed. Demo applet do not support update operation in sign verification. The highest length of signed data can be verified is 508 bytes. And the highest length of the data to be verified is 254 bytes.

**If the signed data is no more than 254 bytes then:**

* First APDU command will send the signed data started with a tag “20”.
* Second APDU command will send the data to be verified started with a tag “10”.

**If the signed data length is larger than 254 bytes then:**

Two APDU command will needed to send the signed data and third APDU command will send the data to be verified.

* First APDU command will send the first part (254 bytes without tag) of signed data started with a tag “20”.
* Second APDU command will send the last part (254 bytes without tag) of signed data started with a tag “40”.
* And the third APDU command will send the data to be verified, started with a tag “10”.

User cannot send last part of signed data before the first part of signed data is already sent. Moreover, the sign verification process starts when the data to be verified is sent! So user cannot send the data to be verified before the first part of signed data is already sent.

Sample APDU for verify DS:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CLA | INS | P1 | P2 | LC | Data | LE |
| 00 | 2A | 00 | A8 | Data length | 20 + signed data | -- |
| 00 | 2A | 00 | A8 | Data length | 10 + data to be verified | -- |

# 2.7.2 Message Authentication

Message Authentication (MAC) service mainly does two things:

1. MAC generation.
2. MAC verification.

**MAC generation:**

First the applet user Restore or SET the SE for DS operation with appropriate algorithm. Then, the Key for MAC generation will be initialized according to the algorithm in SE.

If the algorithm is for DES, AES or HMAC then the DEM-MAC key containers will be searched for key data with corresponding key types like DES, AES or HMAC, to initialize the MAC Key. So Update Demo Keys Service must be performed before DS service.

Sample APDU for generating MAC:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CLA | INS | P1 | P2 | LC | Data | LE |
| 00 | 2A | 9E | 9A | Data length | Data for generating MAC | -- |

**MAC verification:**

First the applet user Restore or SET the SE for DS operation with appropriate algorithm. Then, the Key for MAC generation will be initialized according to the algorithm in SE.

If the algorithm is for DES, AES or HMAC then the DEM-MAC key containers will be searched for key data with corresponding key types like DES, AES or HMAC, to initialize the MAC Key. So Update Demo Keys Service must be performed before DS service.

To verify MAC, multiple APDU command is needed. Demo applet do not support update operation in MAC verification. The highest length of MAC to be verified is 508 bytes. And the highest length of the data to be verified is 254 bytes.

**If the MAC is no more than 254 bytes then:**

* First APDU command will send the MAC with a tag “20”.
* Second APDU command will send the data to be verified started with a tag “10”.

**If the MAC data length is larger than 254 bytes then:**

Two APDU command will needed to send the MAC and third APDU command will send the data to be verified.

* First APDU command will send the first part (254 bytes without tag) of MAC started with a tag “20”.
* Second APDU command will send the last part (254 bytes without tag) of MAC started with a tag “40”.
* And the third APDU command will send the data to be verified, started with a tag “10”.

User cannot send last part of MAC before the first part of MAC is already sent. Moreover, the MAC verification process starts when the data to be verified is sent! So user cannot send the data to be verified before the first part of MAC is already sent.

Sample APDU for verify MAC:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CLA | INS | P1 | P2 | LC | Data | LE |
| 00 | 2A | 00 | A8 | Data length | 20 + MAC | -- |
| 00 | 2A | 00 | A8 | Data length | 10 + data to be verified | -- |

# 2.7.3 Message Digest

This service is for generating message digest with preferred algorithm. See Demo Applet APDU specification for algorithm list.

First the applet user Restore or SET the SE for HASH operation with appropriate algorithm. Then an APDU command with maximum 255 bytes data for making digest will be send.

Sample APDU for Message Digest:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CLA | INS | P1 | P2 | LC | Data | LE |
| 00 | 2A | 90 | 80 | Data length | Data for message digest | -- |

Response: Digest.

# 2.7.4 Key Wrap

This service is mainly encrypts or decrypts a Key or plain text. So the service is divided into two parts:

1. Wrap.
2. Unwrap.

**Wrap:**

Wrapping means encrypting plain text. The plain text may be a key or plain data!

To perform this service properly, applet user must Restore or SET SE for Confidentiality with appropriate algorithm first.

According to the algorithm settled in SE, the DEM-KEY-WRAP key containers will be searched for key data to initialize the Wrap Key. So Update Demo Keys Service must be performed before Wrap service.

Sample APDU for Wrap:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CLA | INS | P1 | P2 | LC | Data | LE |
| 00 | 2A | 86 | 80 | Data length | Plain data | -- |

Response: Encrypted data.

**Unwrap:**

Unwrapping means decrypt text (plain data/text). The plain text may be an encrypted key or plain data!

To perform this service properly, applet user must Restore or SET SE for Confidentiality with appropriate algorithm first.

According to the algorithm settled in SE, the DEM-KEY-WRAP key containers will be searched for key data to initialize the Unwrap Key. So Update Demo Keys Service must be performed before Unwrap service.

Sample APDU for Unwrap:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CLA | INS | P1 | P2 | LC | Data | LE |
| 00 | 2A | 80 | 86 | Data length | Plain/encrypted data | -- |

Response: Decrypted data.

# 2.8 Key Agreement

Demo Applet’s Key Agreement service uses all EC key pairs suggest by NIST and FIPS.

Through Key Agreement service two or more parties, like off-card and on-card, can agree on a Key/Secret in such a way that both influence the outcome.

To generate the Secret, Key Agreement service needs the EC curves name, like EC-384 or EC-521 and EC public key in uncompressed forms as per ANSI X9.62 by an APDU command.

Key Agreement service generates an EC Key Pair with the EC curve name and use the private key and public key from APDU to generate the Secret.

In response the Key Agreement service returns the Public key of the key pair it generates!

Sample APDU for Key Agreement with EC-224:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CLA | INS | P1 | P2 | LC | Data | LE |
| 00 | 86 | 12 | 00 | Data length | Public key of EC-224 curve | -- |

# 2.9 Destroy

This service will initialize all the Key Containers with zeros! And all of the Demo Applet’s internal key objects that preserve the keys/key pairs for further use will clear!

Hence no operation can be done using those objects or key containers.

Sample APDU for Destroy:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CLA | INS | P1 | P2 | LC | Data | LE |
| 00 | 30 | 00 | 00 | -- | -- | -- |