XFS4IOT Sample Messaging for KeyManagement Draft 0.0.4 documentation

Introduction

- Basic Information

Documentation

- Appendix-A
- Remote Key Loading Using Signatures
- Initialization Phase – Signature Issuer and ATM PIN
- Initialization Phase – Signature Issuer and Host
- Key Exchange – Host and ATM PIN
- Key Exchange (with random number) – Host and ATM PIN
- Enhanced RKL, Key Exchange (with random number) – Host and ATM PIN
- Remote Key Loading Using Certificates
- Certificate Exchange and Authentication
- Remote Key Exchange
- Replace Certificate
- Primary and Secondary Certificate
- TR34 BIND To Host
- TR34 Key Transport
- TR34 REBIND To New Host
- TR34 Force REBIND To New Host
- TR34 UNBIND From Host
- TR34 Force UNBIND From Host
- EMV Support
- ImportKey command Input-Output Parameters
- Appendix–D (TR-31 Key Use)

Commands

- KeyManagement.PowerSaveControl
- KeyManagement.SynchronizeCommand
- KeyManagement.GetStatus
- KeyManagement.GetCapabilities
- KeyManagement.GetKeyDetail
- KeyManagement.Initialization
- KeyManagement.DeriveKey
- KeyManagement.Reset
- KeyManagement.ImportKey
- KeyManagement.ExportRSAIssuerSignedItem
- KeyManagement.GenerateRSAKeyPair
- KeyManagement.ExportRSAEPPSignedItem
- KeyManagement.GetCertificate
- KeyManagement.ReplaceCertificate
- KeyManagement.StartKeyExchange
- KeyManagement.GenerateKCV
- KeyManagement.LoadCertificate
- KeyManagement.StartAuthenticate

Unsolicited Events

- KeyManagement.DevicePositionEvent

- KeyManagement.DevicePositionEvent
- KeyManagement.PowerSaveChangeEvent
- KeyManagement.InitializedEvent
- KeyManagement.IllegalKeyAccessEvent
- KeyManagement.CertificateChangeEvent

Events

- Common.PowerSaveChangeEvent

XFS4IOT Sample Messaging for KeyManagement Draft 0.0.4

This section describes the general interface for the following functions:

- Loading of encryption keys
- EMV 4.0 PIN blocks, EMV 4.0 public key loading, static and dynamic data verification Important Notes: * This revision of this specification does not define all key management procedures; some key management is still vendor-specific.
- Key space management is customer-specific, and is therefore handled by vendor-specific mechanisms. Key values are passed to the API as binary hexadecimal values, for example: 0123456789ABCDEF = 0x01 0x23 0x45 0x67 0x89 0xAB 0xCD 0xEF When hex values are passed to the API within strings, the hex digits 0xA to 0xF can be represented by characters in the ranges 'a' to 'f' or 'A' to 'F'. The following commands and events were initially added to support the German ZKA standard, but may also be used for other national standards: Certain levels of the PCI EPP security standards specify that if a key encryption key is deleted or replaced, then all keys in the hierarchy under that key encryption key are also removed. Key encryption keys have the keyEncKey type of access. Applications can check impact of key deletion using Keydetail.

Documentation

Appendix-A

This section provides extended explanation of concepts and functionality needing further clarification. The terminology as described below is used within the following sections.

Definitions and Abbreviations	
ATM	Automated Teller Machine, used here for any type of self-service terminal, regardless whether it actually dispenses cash
CA	Certificate Authority
Certificate	A data structure that contains a public key and a name that allows certification of a public key belonging to a specific individual. This is certified using digital signatures.
HOST	The remote system that an ATM communicates with.
KTK	Key Transport Key
PKI	Public Key Infrastructure
Private Key	That key of an entity's key pair that should only be used by that entity.
Public Key	That key of an entity's key pair that can be made public.
Symmetric Key	A key used with symmetric cryptography
verification Key	A key that is used to verify the validity of a certificate
signatureIssuer	An entity that signs the ATM´s public key at production time, may be the ATM manufacturer

Notation of Cryptographic Items and Functions	
SKE	The private key belonging to entity E
PKE	The public belonging to entity E
SKATM	The private key belonging to the ATM/PIN
PKATM	The public key belonging to the ATM/PIN
SKHOST	The private key belonging to the Host
PKHOST	The public key belonging to the Host
SKSI	The private key belonging to Signature Issuer
PKSI	The public key belonging to Signature Issuer
SKROOT	The root private key belonging to the Host
PKROOT	The root public key belonging to the Host
KNAME	A symmetric key
CertHOST	A Certificate that contains the public verification of the host and is signed by a trusted Certificate Authority.
CertATM	A Certificate that contains the ATM/PIN public verification or encipherment key, which is signed by a trusted Certificate Authority.
CertCA	The Certificate of a new Certificate Authority
RATM	Random Number of the ATM/PIN
IHOST	Identifier of the Host
KKTK	Key Transport Key
RHOST	Random number of the Host
IATM	Identifier of the ATM/PIN
TPATM	Thumb Print of the ATM/PIN
Sign(SKE)[D]	The signing of data block D, using the private key SKE
Recover(PKE)[S]	The recovery of the data block D from the signature S, using the private key PKE
RSACrypt(PKE)[D]	RSA Encryption of the data block D using the public key PKE
Hash [M]	Hashing of a message M of arbitrary length to a 20 Byte hash value
Des(K) [D]	DES encipherment of an 8 byte data block D using the secret key K
Des-1(K)[D]	DES decipherment of an 8 byte data block D using the 8 byte secret key K
Des3(K)[D]	Triple DES encipherment of an 8 byte data block D using the 16 byte secret key K = (KL
Des3-1 (K) [D]	Triple DES decipherment of an 8 byte data block D using the 16 byte secret key K = (KL
RndE	A random number created by entity E

Notation of Cryptographic Items and Functions	
UIE	Unique Identifier for entity E
(A || B)	Concatenation of A and B

Remote Key Loading Using Signatures

RSA Data Authentication and Digital Signatures

Digital signatures rely on a public key infrastructure (PKI). The PKI model involves an entity, such as a Host, having a pair of encryption keys – one private, one public. These keys work in consort to encrypt, decrypt and authenticate data. One way authentication occurs is through the application of a digital signature. For example:

1. The Host creates some data that it would like to digitally sign;
2. Host runs the data through a hashing algorithm to produce a hash or digest of the data. The digest is unique to every block of data – a digital fingerprint of the data, much smaller and therefore more economical to encrypt than the data itself.
3. Digest is encrypted with the Host's private key.

This is the digital signature – a data block digest encrypted with the private key. The Host then sends the following to the ATM:

1. Data block.
2. Digital signature.
3. Host's public key.

To validate the signature, the ATM performs the following:

1. ATM runs data through the standard hashing algorithm – the same one used by the Host – to produce a digest of the data received. Consider this digest2;
2. ATM uses the Host's public key to decrypt the digital signature. The digital signature was produced using the Host's private key to encrypt the data digest; therefore, when decrypted with the Host's public key it produces the same digest. Consider this digest1. Incidentally, no other public key in the world would work to decrypt digest1 – only the public key corresponding to the signing private key.
3. ATM compares digest1 with digest2.

If digest1 matches digest2 exactly, the ATM has confirmed the following:

• Data was not tampered with in transit. Changing a single bit in the data sent from the Host to the ATM would cause digest2 to be different than digest1. Every data block has a unique digest; therefore, an altered data block is detected by the ATM.

• Public key used to decrypt the digital signature corresponds to the private key used to create it. No other public key could possibly work to decrypt the digital signature, so the ATM was not handed someone else's public key. This gives an overview of how Digital Signatures can be used in Data Authentication. In particular, Signatures can be used to validate and securely install Encryption Keys. The following section describes Key Exchange and the use of Digital signatures.

RSA Secure Key Exchange using Digital Signatures

In summary, both end points, the ATM and the Host, inform each other of their Public Keys. This information is then used to securely send the PIN device Master Key to the ATM. A trusted third party, the Signature Issuer, is used to generate the signatures for the Public keys of each end point, ensuring their validity.

The detail of this is as follows:

Purpose: The Host wishes to install a new master key (KM) on the ATM securely.

Assumptions:

1. The Host has obtained the Public Key (PK SI) from the Signature Issuer.
2. The Host has provided the Signature Issuer with its Public Key (PK HOST), and receives the corresponding signature Sign(SK SI)[PK HOST]. The Signature Issuer uses its own Private Key (SK SI) to create this signature.
3. In the case where Enhanced Remote Key Loading is used, the host has provided the Signature Issuer with its Public Key (PK ROOT), and receives the corresponding signature Sign(SK SI)[PK ROOT]. The host has generated another key pair PKHOST and SKHOST and signs the PKHOST with the SKROOT.
4. (Optional) The host obtains a list of the valid PIN device's Unique Identifiers. The Signature Issuer installs a Signature Sign(SK SI)[UI ATM] for the Unique Id (UI ATM) on the ATM PIN. The Signature Issuer uses SKSI to do this.
5. The Signature Issuer installs its Public Key (PK SI) on the ATM PIN. It also derives and installs the Signature Sign(SK SI)[PK ATM] of the ATM PIN's Public Key (PK ATM) on the ATM PIN. The Signature Issuer uses SKSI to do this.
6. The ATM PIN device additionally contains its own Public (PK ATM) and Private Key (SK ATM).

Step 1

The ATM PIN sends its Public Key to the Host in a secure structure:

The ATM PIN sends its ATM Public Key with its associated Signature. When the Host receives this information it will use the Signature Issuer's Public Key to validate the signature and obtain the ATM Public Key.

The command used to export the PIN public key securely as described above is ExportRsaIssuerSignedItem.

Step 2 (Optional)

The Host verifies that the key it has just received is from a valid sender.

It does this by obtaining the PIN device unique identifier. The ATM PIN sends its Unique Identifier with its associated Signature. When the Host receives this information it will use the Signature Issuer's Public Key to validate the signature and retrieve the PIN Unique Identifier. It can then check this against the list it received from the Signature Issuer.

The command used to export the PIN Unique Identifier is ExportRsaIssuerSignedItem.

Step 3 (Enhanced Remote Key Loading only)

The Host sends its root public key to the ATM PIN:

The Host sends its Root Public Key (PKROOT) and associated Signature. The ATM PIN verifies the signature using PKSI and stores the key.

The command used to import the host root public key securely as described above is ImportRsaPublicKey.

Step 4

The Host sends its public key to the ATM PIN:

The Host sends its Public Key (PK HOST) and associated Signature. The ATM PIN verifies the signature using PKSI (or PKROOT in the Enhanced Remote Key Loading Scheme) and stores the key.

The command used to import the host public key securely as described above is ImportRsaPublicKey.

Step 5 The ATM PIN receives its Master Key from the Host:

The Host encrypts the Master Key (KM) with PKATM. A signature for this is then created using SKHOST. The ATM PIN will then validate the signature using PKHOST and then obtain the master key by decrypting using SKATM.

The commands used to exchange master symmetric keys as described above are:

• StartKeyExchange

• ImportRsaSignedDesKey

Step 6 – Alternative including random number The host requests the ATM PIN to begin the DES key transfer process and generate a random number.

The Host encrypts the Master Key (KM) with PKATM. A signature for the random number and encrypted key is then created using SKHOST.

The ATM PIN will then validate the signature using PKHOST, verify the random number and then obtain the master key by decrypting using SKATM.

The commands used to exchange master symmetric keys as described above are:

• StartKeyExchange

• ImportRsaSignedDesKey

The following diagrams summaries the key exchange process described above:

Default Keys and Security Item loaded during manufacture

Several keys and a security item which are mandatory for the 2 party/Signature authentication scheme are installed during manufacture. These items are given fixed names so multi-vendor applications can be developed without the need for vendor specific configuration tools.

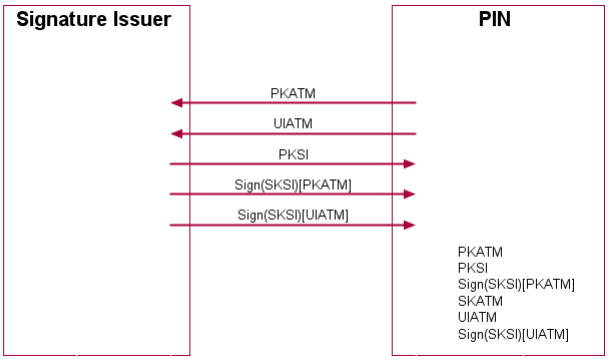
Item Name	Item Type	Signed by	Description
“sigIssuerVendor”	Public Key	N/A	The public key of the signature issuer, i.e. PKSI
“eppCryptKey”	Public/Private key-pair	The private key associated with sigIssuerVendor	The key-pair used to encrypt and encrypt the symmetric. key, i.e SK ATM and PK ATM . The public key is used for encryption by the host and the private for decryption by the epp.

In addition the following optional keys can be loaded during manufacture.

Item Name	Item Type	Signed by	Description
“eppSignKey”	Public/Private key-pair	The private key associated with sigIssuerVendor	A key-pair where the private key is used to sign data, e.g. other generated key pairs

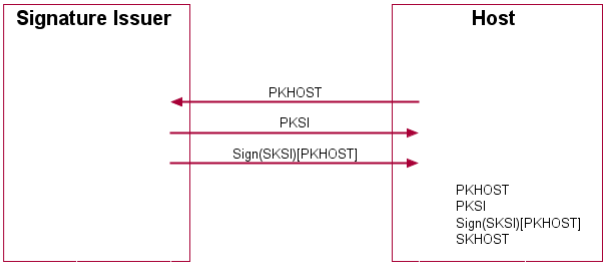
Initialization Phase – Signature Issuer and ATM PIN

This would typically occur in a secure manufacturing environment.



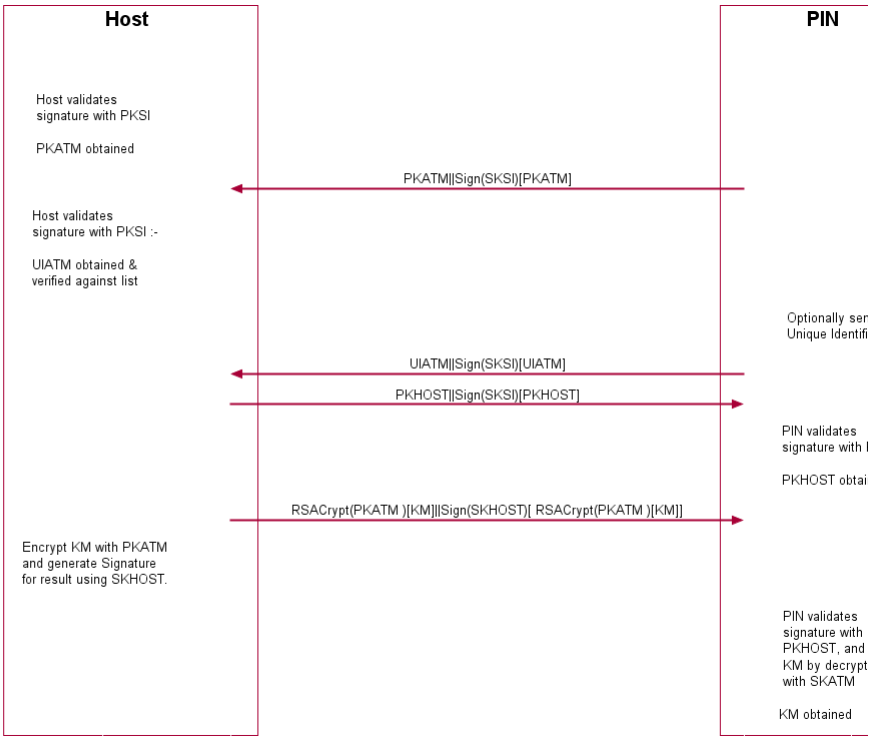
Initialization Phase – Signature Issuer and Host

This would typically occur in a secure offline environment.



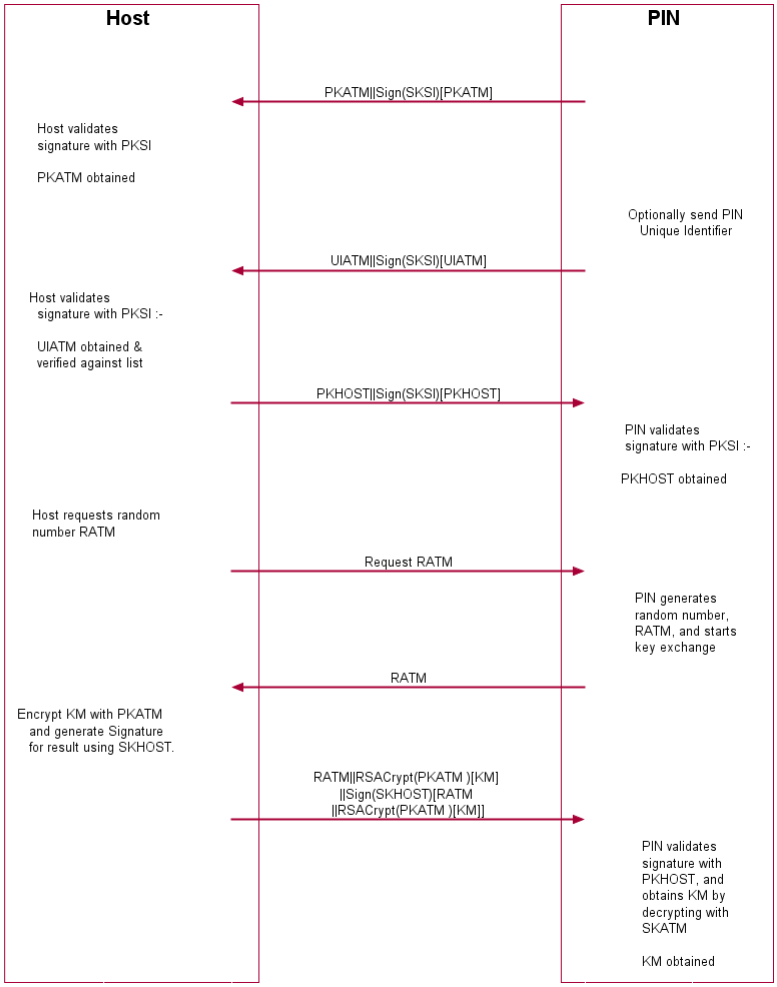
Key Exchange – Host and ATM PIN

This following is a typical interaction for the exchange of the initial symmetric master key in a typical ATM Network. The following is the recommended sequence of interchanges.



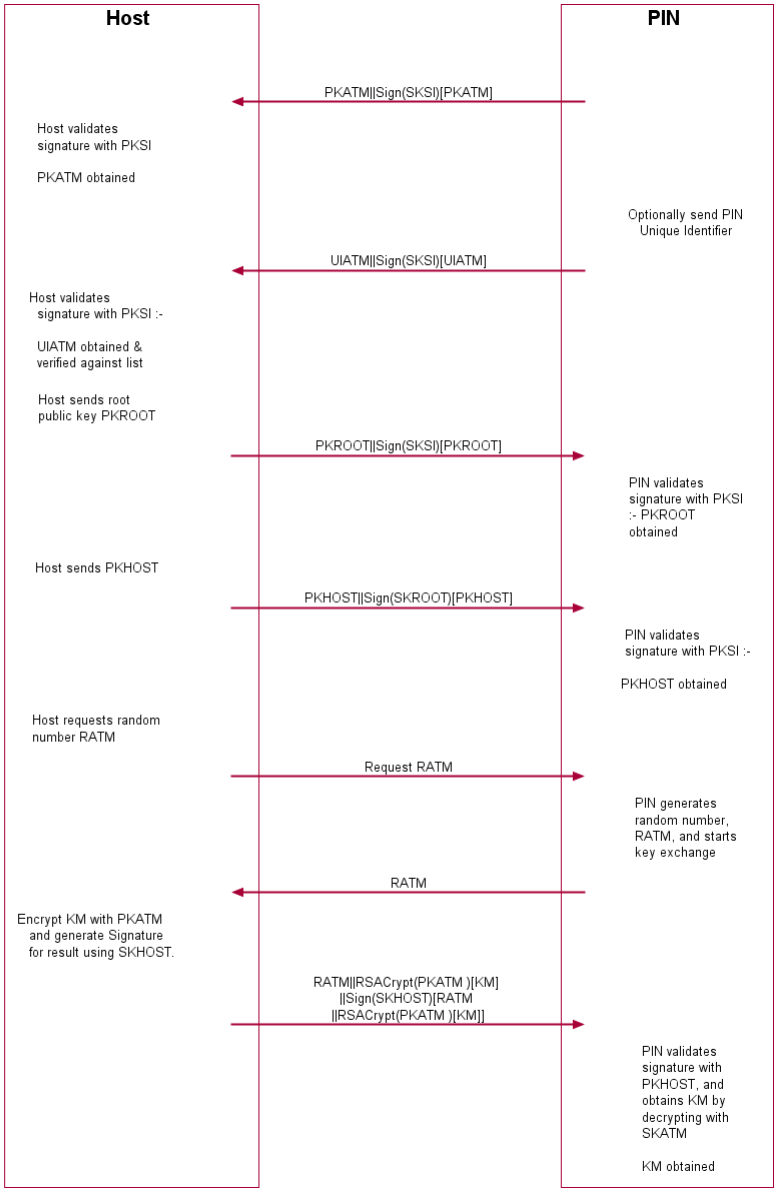
Key Exchange (with random number) – Host and ATM PIN

This following is a typical interaction for the exchange of the initial symmetric master key when the PIN device Service Provider supports the StartKeyExchange command.



Enhanced RKL, Key Exchange (with random number) – Host and ATM PIN

This following is a typical interaction for the exchange of the initial symmetric master key when the PIN device and Service Provider supports the Enhanced Signature Remote Key Loading scheme.



Remote Key Loading Using Certificates

The following sections demonstrate the proper usage of the CEN pin interface to accomplish Remote Key Loading using Certificates. Beginning with Section 8.2.5, there are sequence diagrams to demonstrate how the CEN pin interface can be used to complete each of the TR34 operations.

Certificate Exchange and Authentication

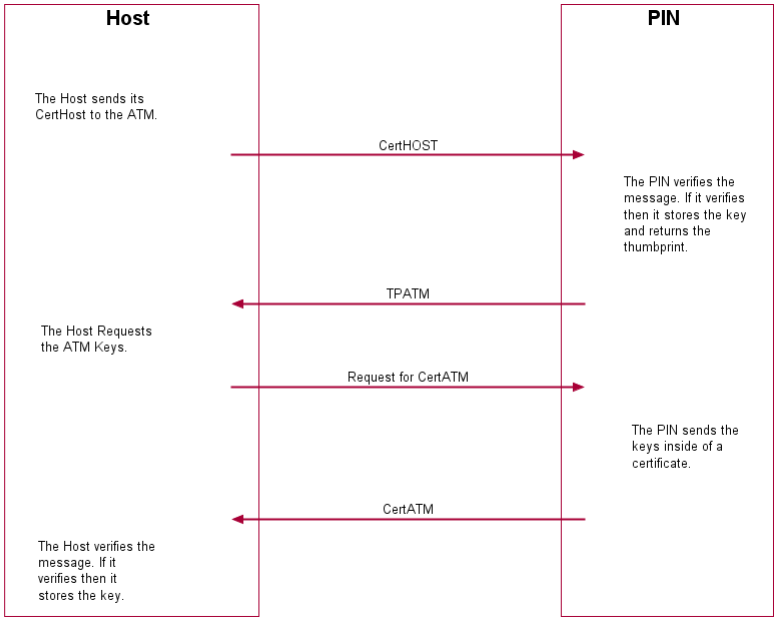
In summary, both end points, the ATM and the Host, inform each other of their Public Keys. This information is then used to securely send the PINS device Master Key to the ATM. A trusted third party, Certificate Authority (or a HOST if it becomes the new CA), is used to generate the certificates for the Public Keys of each end point, ensuring their validity. NOTE: The LoadCertificate and GetCertificate do not necessarily need to be called in the order below. This way though is the recommend way.

The following flow is how the exchange authentication takes place:

• LoadCertificate is called. In this message contains the host certificate, which has been signed by the trusted CA. The encryptor uses the Public Key of the CA (loaded at the time of production) to verify the validity of the certificate. If the certificate is valid, the encryptor stores the HOST's Public Verification Key.

• Next, GetCertificate is called. The encryptor then sends a message that contains a certificate, which is signed by the CA and is sent to the HOST. The HOST uses the Public Key from the
CA to verify the certificate. If valid then the HOST stores the encryptor's verification or encryption key
(primary or secondary this depends on the state of the encryptor).

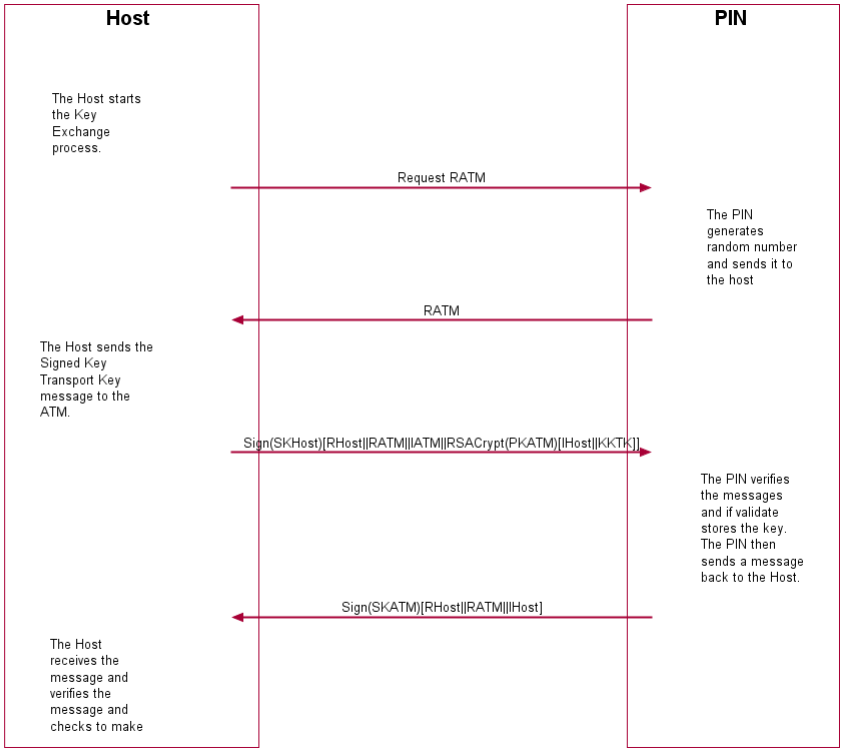
The following diagram shows how the Host and ATM Load and Get each other's information to make Remote Key Loading possible:



Remote Key Exchange

After the above has been completed, the HOST is ready to load the key into the encryptor. The following is done to complete this and the application must complete the Remote Key Exchange in this order: First, the StartKeyExchange is called. This returns RATM from the encryptor to be used in the authenticating the ImportRSAEnchiperdPKCS7Key message.
Next, ImportRSAEnchiperdPKCS7Key is called. This command sends down the KTK to the encryptor. The following items below show how this is accomplished. a) HOST has obtained a Key Transport Key and wants to transfer it to the encryptor. HOST constructs a
key block containing an identifier of the HOST, IHOST, and the key, KKTK, and enciphers the block, using the encryptor's Public Encryption Key from the GetCertificate command. b) After completing the above, the HOST generates random data and builds the outer message containing the random number of the host, RHOST, the random number of the encryptor returned in the StartKeyExchange command, RATM , the identifier of the encryptor, IENC, and the enciphered key block. The HOST signs the whole block using its private signature key and sends the message down to the encryptor. The encryptor then verifies the HOST's signature on the message by using the HOST's Public Verification Key. Then the encryptor checks the identifier and the random number of the encryptor passed in the message to make sure that the encryptor is talking to the right HOST. The encryptor then deciphers the enciphered block using its private verification key. After the message has been deciphered, the encryptor checks the Identifier of the HOST. Finally, if everything checks out to this point the encryptor will load the Key Transport Key. NOTE: If one step of this verification occurs the encryptor will return the proper error to the HOST. c) After the Key Transport Key has been accepted, the encryptor constructs a message that contains the random number of the host, the random number of the encryptor and the HOST identifier all signed by the private signature key of the encryptor. This message is sent to the host. d) The HOST verifies the message sent from the encryptor by using the ATM's public verification key. The HOST then checks the identifier of the host and then compares the identifier in the message with the one stored in the HOST. Then checks the random number sent in the message and to the one stored in the HOST. The HOST finally checks the encryptor's random number with the one received in received in the StartKeyExchange command.

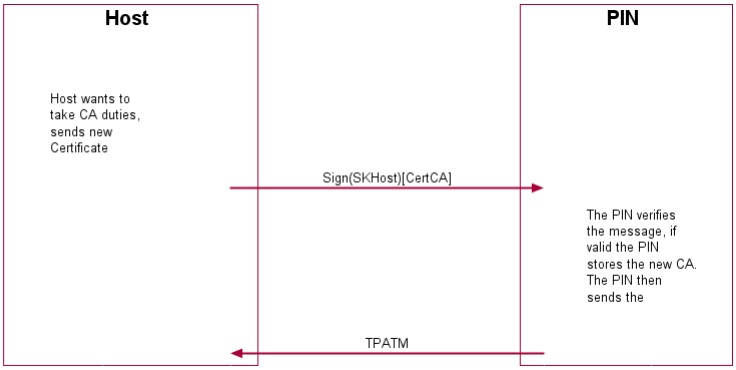
The following diagram below shows how the Host and ATM transmit the Key Transport Key.



Replace Certificate

After the key is been loaded into the encryptor, the following could be completed:

• (Optional) ReplaceCertificate. This is called by entity that would like to take over the job of being the CA. The new CA requests a Certificate from the previous Certificate Authority. The HOST must over-sign the message to take over the role of the CA to ensure that the encryptor accepts the new Certificate Authority. The HOST sends the message to the encryptor. The encryptor uses the HOST's Public Verification Key to verify the HOST's signature. The encryptor uses the previous CA's Public Verification Key to verify the signature on the new Certificate sent down in the message. If valid, the EPP stores the new CA's certificate and uses the new CA's Public Verification Key as its new CA verification key. The diagram below shows how the Host and the ATM communicate to load the new CA.



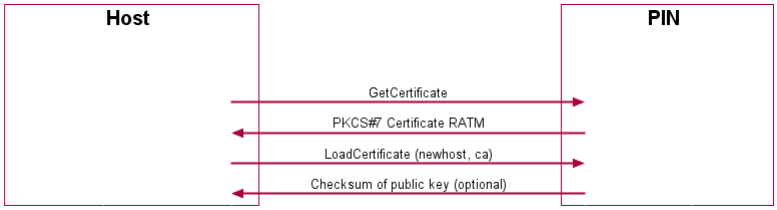
Primary and Secondary Certificate

Primary and Secondary Certificates for both the Public Verification Key and Public Encipherment Key are pre-loaded into the encryptor. Primary Certificates will be used until told otherwise by the host via the LoadCertificate or ReplaceCertificate commands. This change in state will be specified in the pkcs #7 message of the LoadCertificate or ReplaceCertificate commands. The reason why the host would want to change states is because the host thinks that the Primary Certificates have been compromised.

After the host tells the encryptor to shift to the secondary certificate state, only Secondary Certificates can be used. The encryptor will no longer be able to go back to the Primary State and any attempts from the host to get or load a Primary Certificate will return an error. When either Primary or Secondary certificates are compromised it is up to the vendor on how the encryptor should be handled with the manufacturer.

TR34 BIND To Host

This section defines the command to use when transferring a TR34 BIND token as defined in X9 TR34-2012 This step is a pre-requisite for all other TR34 operations. The PIN device must be bound to a host before any other TR34 operation will succeed. It is recommended that the encryption certificate retrieved during this process is stored for future use otherwise it will need to be requested prior to every operation.

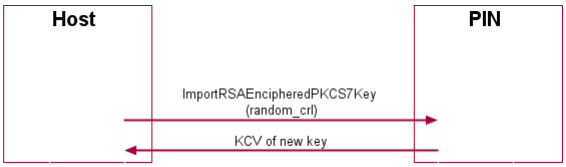


TR34 Key Transport

There are two mechanisms that can be used to transport symmetric keys under TR34; these are the One Pass and Two Pass protocols. The use of CEN commands for these two protocols are shown in the following sections. NOTE: Refer to CRKLLoadOptions in the Capabilities output structure for an indication of whether the PIN device supports one-pass and/or two-pass protocols.

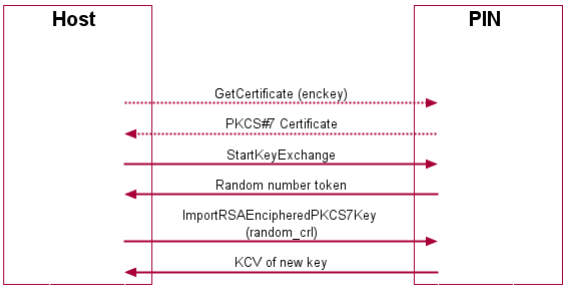
One Pass

This section defines the command to use when transferring a TR34 KEY token (1-pass) as defined in X9 TR342012. Pre-condition: A successful BIND command has completed such that the PIN device is bound to the host.



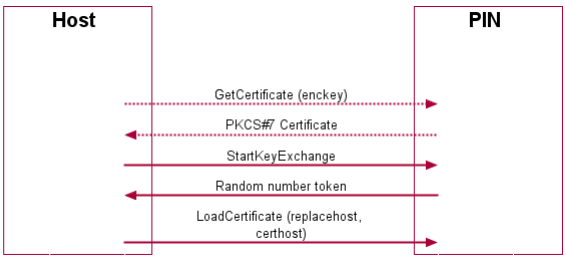
Two Pass

This section defines the command to use when transferring a TR34 KEY token (2-pass) as defined in reference. Pre-condition: A successful BIND command has completed such that the PIN device is bound to the host.



TR34 REBIND To New Host

This section defines the command to use when transferring a TR34 REBIND token as defined in X9 TR34-2012. Pre-condition: A successful BIND command has completed such that the PIN device is bound to the host.

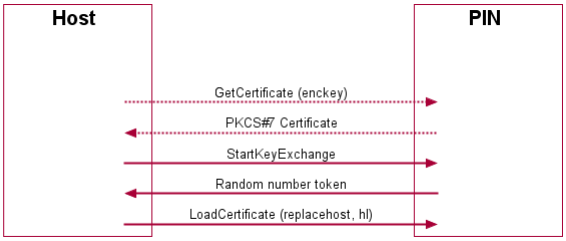


NB: Dotted lines represent commands that are only required if the PIN device encryption certificate has not been previously stored by the host.

TR34 Force REBIND To New Host

This section defines the command to use when transferring a TR34 Force REBIND token as defined in X9 TR342012. Pre-condition: A successful BIND command has

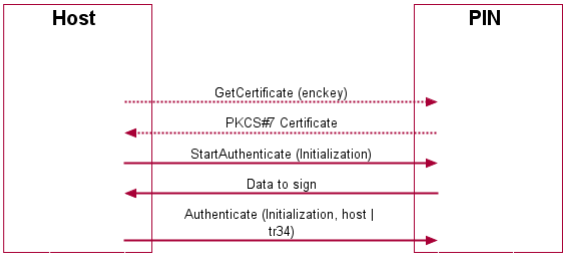
completed such that the PIN device is bound to the host.



NB: Dotted lines represent commands that are only required if the PIN device encryption certificate has not been previously stored by the host. Although the random number token is requested as part of this operation, it is discarded by the host and is not actually used in the Force Rebind token.

TR34 UNBIND From Host

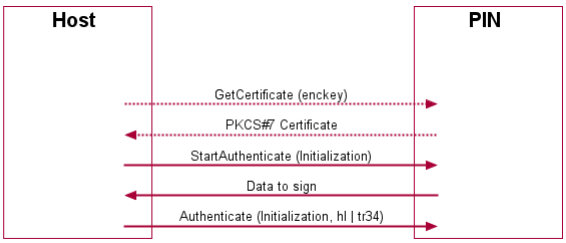
This section defines the command to use when transferring a TR34 UNBIND token as defined in X9 TR34-2012. Pre-condition: A successful BIND command has completed such that the PIN device is bound to the host.



NB: Dotted lines represent commands that are only required if the PIN device encryption certificate has not been previously stored by the host

TR34 Force UNBIND From Host

This section defines the command to use when transferring a TR34 Force UNBIND token as defined in X9 TR342012. Pre-condition: A successful BIND command has completed such that the PIN device is bound to the host.



NB: Dotted lines represent commands that are only required if the PIN device encryption certificate has not been previously stored by the host. Although the random number token is requested as part of this operation, it is discarded by the host and is not actually used in the Force Unbind token.

EMV Support

EMV support by this specification consists in the ability of importing Certification Authority and Chip Card Public Keys, creating the PIN blocks for offline PIN verification and verifying static and dynamic data. This section is used to further explain concepts and functionality that needs further clarification.

The PIN service is able to manage the EMV chip card regarding the card authentication and the RSA local PIN verification. Two steps are mandatory in order to reach these two functions: The loading of the keys which come from the Certification Authorities or from the card itself, and the EMV PIN block management.

The Service Provider is responsible for all key validation during the import process. The application is responsible for management of the key lifetime and expiry after the key is successfully imported

Keys loading

The final goal of an application is to retrieve the keys located on card to perform the operations of authentication or local PIN check (RSA encrypted). These keys are provided by the card using EMV certificates and can be retrieved using a Public Key provided by a Certification Authority. The application should first load the keys issued by the Certification Authority. At transaction time the application will use these keys to load the keys that the application has retrieved from the chip card.

Certification Authority keys

These keys are provided in the following formats:

```
•    Plain text.
•   Plain Text with EMV 2000 Verification Data (See [Ref. 4] under the reference section for this document).
•   EPI CA (or self signed) format as specified in the Europay International, EPI CA Module Technical – Interface specification Version
1.4 (See [Ref. 5] under the reference section for this document).
•   pkcsV15 encrypted (as used by GIECB in France) (See [Ref. 15] under the reference section for this document).
```

EPI CA format

The following table corresponds to table 4 of the Europay International, EPI CA Module Technical – Interface specification Version 1.4 (See [Ref. 5]) and identifies the Europay Public Key (self-certified) and the associated data:

Field Name	Length	Description	Format
ID of Certificate Subject	5	RID for Europay	Binar
Europay public key Index	1	Europay public key Index	Binary
Subject public key Algorithm Indicator	1	Algorithm to be used with the Europay public key Index, set to 0x01	Binary
Subject public key Length	1	Length of the Europay public key Modulus (equal to Nca)	Binary
Subject public key Exponent Length	1	Length of the Europay public key Exponent	Binary
Leftmost Digits of Subject public key	Nca-37	Nca-37 most significant bytes of the Europay public key Modulus	Binary
Subject public key Remainder	37	37 least significant bytes of the Europay public key Modulus	Binary
Subject public key Exponent	1	Exponent for Europay public key	Binary
Subject public key Certificate	Nca	Output of signature algorithm	Binary

Table 1

The following table corresponds to table 13 of the Europay International, EPI CA Module Technical – Interface specification Version 1.4 and identifies the Europay Public Key Hash code and associated data.

Field Name	Length	Description	Format
ID of Certificate Subject	5	RID for Europay	Binary
Europay public key Index	1	Europay public key Index	Binary
Subject public key Algorithm Indicator	1	Algorithm to be used with the Europay public key Index, set to 0x01	Binary
Certification Authority public key Check Sum	20	Hash-code for Europay public key	Binary

Table 2

Table 2 corresponds to table 13 of the Europay International, EPI CA Module Technical – Interface specification Version 1.4 (See [Ref. 5]).

Chip card keys

These keys are provided as EMV certificates which come from the chip card in a multiple layer structure (issuer key first, then the ICC keys). Two kinds of algorithm are used with these certificates in order to retrieve the keys: One for the issuer key and the other for the ICC keys (ICC Public Key and ICC PIN encipherment key). The associated data with these algorithms – The PAN (Primary Account Number) and the SDA (Static Data to be Authenticated) - come also from the chip card.

PIN Block Management

The PIN block management is done through the command GetPinBlock. A new format formEmv has been added to indicate to the PIN service that the PIN block must follow the requirements of the EMVCo, Book2 – Security & Key management Version 4.0 document The parameter customerData is used in this case to transfer to the PIN service the challenge number coming from the chip card. The final encryption must be done using a RSA Public Key. Please note that the application is responsible to send the PIN block to the chip card inside the right APDU.

SHA-1 Digest

The SHA-1 Digest is a hash algorithm used by EMV in validating ICC static and dynamic data item. The SHA-1 Digest is supported through the digest command. The application will pass the data to be hashed to the Service Provider. Once the encryptor completes the SHA-1 hash code, the Service Provider will return the 20-byte hash value back to the application.

ImportKey command Input-Output Parameters

The tables in this section describe the input/output parameters for various scenarios in which the importKey command is used, compared to input/output parameters for older commands that it supercedes.

Importing a 3DES 16-byte terminal master key using signature-based remote key loading (SRKL):

For this example, the following input data is available:

```
Name of key to be imported = testKey
Name of the key used to decrypt the encrypted key value = eppCryptKey
Name of the key used to verify the signature = hostKey
Encrypted key value = <encrypted key value>
Signature = <signature generated by the host>
Usage of the key to be imported = key encrypting key
RSA Encipher Algorithm = rsa es oaep
RSA Signature Algorithm = rsa ssa pss
```

ImportRSASignedDESKey Input Data

Parameter Name	Example Value
key	testKey
decryptKey	eppCryptKey
rsaEnchiperAlgorithm	oaep
value	<encrypted key value>
use	keyEncKey

Parameter Name	Example Value
sigKey	hostKey
rsaSignatureAlgorithm	pss
signature	<signature generated by the host>

For this example, the following output data is expected:

```
Key Check Mode = kcv zero
Key Check Value = <key check value>
Key Length = double length key
```

ImportRSASignedDESKey Output Data

Parameter Name	Example Value
keyLength	double
keyCheckMode	zero
keyCheckValue	<key check value>

ImportKey Input Data

Parameter Name	Example Value
key	testKey
keyAttributes.keyUsage	'K0'
keyAttributes.algorithm	'T'
keyAttributes.modeOfUse	'D'
keyAttributes.cryptoMethod	0
value	<encrypted key value>
decryptKey	eppCryptKey
decryptMethod	rsaesOaep
verificationData	<signature generated by the host>
verifyKey	hostKey
verifyAttributes.keyUsage	'S0'
verifyAttributes.algorithm	'R'
verifyAttributes.modeOfUse	'V'
verifyAttributes.cryptoMethod	rsassaPss
vendorAttributes	

ImportKey Output Data

Parameter Name	Example Value
verifyAttributes.keyUsage	'00'
verifyAttributes.algorithm	'T'
verifyAttributes.modeOfUse	'V'
verifyAttributes.cryptoMethod	zero
verifyData	<key check value>
keyLength	128

Importing a 16-byte DES key for pin encryption with a key check value in the input

For this example, the following input data is available:

```
Name of key to be imported = testKey
Name of the key used to decrypt the encrypted key value = masterKey
Encrypted key value = <encrypted key value>
Usage of the key to be imported = pin encryption
Key Check Mode = kcv Zero
Key Check Value = <key check value>
```

ImportKey Input Data

Parameter Name	Example Value
key	testKey
keyAttributes.keyUsage	'P0' (Similar to use but a more precise key usage)
keyAttributes.algorithm	'T'
keyAttributes.modeOfUse	'E'
keyAttributes.cryptoMethod	0
value	<encrypted key value>
decryptKey	masterKey
decryptMethod	ecb
verificationData	<key check value>
verifyKey	
verifyAttributes.keyUsage	'00'
verifyAttributes.algorithm	'T'
verifyAttributes.modeOfUse	'V'
verifyAttributes.cryptoMethod	zero
vendorAttributes	

Likewise, the following output data is expected:

ImportKey Output Data

Parameter Name	Example Value
verifyAttributes	null
verifyData	
keyLength	128

Importing a 16-byte DES key for macing (MAC Algorithm 3)

For this example, the following input data is available:

```
Name of key to be imported = testKey
Name of the key used to decrypt the encrypted key value = masterKey
Encrypted key value = <encrypted key value>
Usage of the key to be imported = mac
```

ImportKey Input Data

Parameter Name	Example Value
key	testKey
keyAttributes.keyUsage	'M3' (Similar to fwUse but a more precise key usage)
keyAttributes.algorithm	'T'
kyAttributes.modeOfUse	'G'
keyAttributes.cryptoMethod	0
value	<encrypted key value>
decryptKey	masterKey
decryptMethod	ecb
verificationData	
verifyKey	
verifyAttributes	null
vendorAttributes	

ImportKey Output Data

Parameter Name	Example Value
verifyAttributes.keyUsage	'00'
verifyAttributes.algorithm	'T'
verifyAttributes.modeOfUse	'V'
verifyAttributes.cryptoMethod	zero
verifyData	<key check value>
keyLength	128

Importing a 2048-bit Host RSA public key

For this example, the following input data is available:

```
    Name of key to be imported = HostKey
    Name of the key used to verify the signature = sigIssuerVendor
    Key value = <key value>
    Signature = <signature generated by the vendor signature issuer>
    Usage of the key to be imported = RSA signature verification
    RSA Signature Algorithm = RSA SSA PSS
```

ImportRSAPublicKey Input Data

Parameter Name	Example Value
key	hostKey
value	<key value>
use	rsaPublicVerify
sigKey	sigIssuerVendor
rsaSignatureAlgorithm	rsassaPss
signature	<signature generated by the vendor signature issuer>

For this example, the following output data is expected:

```
    RSA Key Check Mode = sha256 digest
    Key Check Value = <sha256 digest>
    Key Length = 2048
```

ImportRSAPublicKey Output Data

Parameter Name	Example Value
rsaKeyCheckMode	sha256
keyCheckValue	<sha256 digest>

ImportKey Input Data

Parameter Name	Example Value
key	hostKey
keyAttributes.keyUsage	'S0'
keyAttributes.algorithm	'R'
keyAttributes.modeOfUse	'V'

Parameter Name	Example Value
keyAttributes.cryptoMethod	0
value	<key value>
decryptKey	
decryptMethod	0
verificationData	<signature generated by the vendor signature issuer>
verifyKey	sigIssuerVendor
verifyAttributes.KeyUsage	'S1'
verifyAttributes.Algorithm	'R'
verifyAttributes.ModeOfUse	'V'
verifyAttributes.CryptoMethod	rsassaPss
vendorAttributes	

ImportKey Output Data

Parameter Name	Example Value
verifyAttributes.algorithm	'R'
verifyAttributes.modeOfUse	'V'
verifyAttributes.CryptoMethod	sha256
verifyData	<sha256 digest>
keyLength	2048

Importing a 24-byte DES symmetric data encryption key via TR-31 keyblock

For this example, the following input data is available:

```
Name of key to be imported = testKey
Name of the key block protection key = masterKey
Key block = <key block>
```

ImportKeyBlock Input Data

Parameter Name	Example Value
Key	testKey
EncKey	masterKey
KeyBlock	<key block>

For this example, the following output data is expected:

Key Length = triple length (192 bits) DES key

ImportKeyBlock Output Data None

ImportKey Input Data

Parameter Name	Example Value
key	testKey
keyAttributes.keyUsage	'D0'
keyAttributes.algorithm	'T'
keyAttributes.modeOfUse	'E'
keyAttributes.cryptoMethod	0
value	<key block>
decryptKey	masterKey
decryptMethod	0
verificationData	
verifyKey	
verifyAttributes	null
vendorAttributes	

ImportKey Output Data

Parameter Name	Example Value
verifyAttributes	null
verifyData	
keyLength	192

Appendix–D (TR-31 Key Use)

This section contains a mapping of key usages as defined for TR-31 (see ANS X9 TR-31 2010 [Ref. 35]) to the use values defined in this document. The use values are those defined for the use input/output fields of a number of different PIN commands.

Keys imported within an ANS TR-31 key block have a usage encoded into the key block header (represented by BlockHeader in the KeyDetail commands), This usage specified in the key block header may be more specific than the Use values defined in this document. For consistency, the following table defines the corresponding Use value for each TR-31 key usage:

TR-31 Value	TR-31 Mode(s) of use	Definition	use
"B0"	"X"	BDK Base Derivation Key	keyDerKey
"B1"	"X"	DUKPT Initial Key (also known as IPEK)	keyDerKey** pinRemote function* crypt macing
"C0"	"C", "G", "V"	CVK Card Verification Key	NA

TR-31 Value	TR-31 Mode(s) of use	Definition	use
"D0"	"B", "D", "E"	Data Encryption using ecb, cbc, cfb, ofb, ccm or ctr	crypt
"E0"	"X"	EMV/chip Issuer Master Key: Application cryptograms	rsaPublicVerify
"E1"	"X"	EMV/chip Issuer Master Key: Secure Messaging for Confidentiality	rsaPublicVerify
"E2"	"X"	EMV/chip Issuer Master Key: Secure Messaging for Integrity	rsaPublicVerify
"E3"	"X"	EMV/chip Issuer Master Key: Data Authentication Code	rsaPublicVerify
"E4"	"X"	EMV/chip Issuer Master Key: Dynamic Numbers	rsaPublicVerify
"E5"	"X"	EMV/chip Issuer Master Key: Card Personalization	rsaPublicVerify
"E6"	"X"	EMV/chip Issuer Master Key: Other	rsaPublicVerify
"I0"	"N"	Initialization Vector (IV)	NA
"K0"	"B", "D", "E"	Key Encryption or wrapping	keyEncKey svEncKey
"K1"	"B", "D", "E"	TR-31 Key Block Protection Key	anstr31Master
"M0"	"C", "G", "V"	ISO 16609 MAC algorithm 1 (using TDEA)	macing
"M1"	"C", "G", "V"	ISO 9797-1 MAC Algorithm 1	macing
"M2"	"C", "G", "V"	ISO 9797-1 MAC Algorithm 2	macing
"M3"	"C", "G", "V"	ISO 9797-1 MAC Algorithm 3	macing
"M4"	"C", "G", "V"	ISO 9797-1 MAC Algorithm 4	macing
"M5"	"C", "G", "V"	ISO 9797-1 MAC Algorithm 5	macing
"P0"	"B", "D", "E"	PIN Encryption	pinRemote function*
"V0"	"C", "G", "V"	PIN verification, KPV, other algorithm	pinLocal function*
"V1"	"C", "G", "V"	PIN verification, IBM 3624	pinLocal function*
"V2"	"C", "G", "V"	PIN Verification, VISA PVV	pinLocal function*

*Note that function is listed here for backward compatibility, but pinLocal/pinRemote is the more accurate single-use value.

** The Base Derivation Key is used to derive the IPEK. When a dukpt IPEK is loaded, derived future keys are stored and the IPEK deleted. Therefore, while the IPEK is no longer loaded, future keys directly related to it are. pinRemote and optionally function are included as the primary use of an IPEK future key is to create a variant for PIN encryption. If the optional variant data encryption and MAC keys are supported, crypt and macing must be included. To use the optional data or MAC keys in a crypt command, key must be the name of the IPEK and Algorithm must be cryptTriDesCbc cryptTriDesMac. If the optional data encryption key is being used, Mode must be modeEncCrypt. The optional variant response data encryption and MAC keys are not supported.

Commands

KeyManagement.PowerSaveControl

Description

This command activates or deactivates the power-saving mode. If the Service Provider receives another execute command while in power saving mode, the Service Provider automatically exits the power saving mode, and executes the requested command. If the Service Provider receives an information command while in power saving mode, the Service Provider will not exit the power saving mode.

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
maxPowerSaveRecoveryTime	integer		Specifies the maximum number of seconds in which the device must be able to return to its normal operating state when exiting power save mode. The device will be set to the highest possible power save mode within this constraint. If usMaxPowerSaveRecoveryTime is set to zero then the device will exit the power saving mode.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "maxPowerSaveRecoveryTime": 0
  }
}
```

Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string"
  }
}
```


Event Messages

- Common.PowerSaveChangeEvent

KeyManagement.SynchronizeCommand

Description

This command is used to reduce response time of a command (e.g. for synchronization with display) as well as to synchronize actions of the different device classes. This command is intended to be used only on hardware which is capable of synchronizing functionality within a single device class or with other device classes.

The list of execute commands which this command supports for synchronization is retrieved in the *lpdwSynchronizableCommands* parameter of the WFS_INF_CDM_CAPABILITIES.

This command is optional, i.e. any other command can be called without having to call it in advance. Any preparation that occurs by calling this command will not affect any other subsequent command. However, any subsequent execute command other than the one that was specified in the *dwCommand* input parameter will execute normally and may invalidate the pending synchronization. In this case the application should call the WFS_CMD_CDM_SYNCHRONIZE_COMMAND again in order to start a synchronization.

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
command	string		The command name to be synchronized and executed next.
cmdData	object		A payload that represents the parameter that is normally associated with the command.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "command": "string",
    "cmdData": {}
  }
}
```


Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string"
  }
}
```


Event Messages

KeyManagement.GetStatus

Description

This command returns several kinds of status information

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000"
  }
}
```

Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
device	string		Specifies the state of the device.
extra	array		Specifies a list of vendor-specific, or any other extended, information. The information is returned as a series of "key=value" strings so that it is easily extendable by Service Providers.
guidLights	array		Specifies the state of the guidance light indicators. A number of guidance light types are defined below. Vendor specific guidance lights are defined starting from the end of the array.
guidLights.flashRate	string		Indicates the current flash rate of the guidelight.
guidLights.color	string		Indicates the current color of the guidelight.
guidLights.direction	string		Indicates the current direction of the guidelght.
devicePosition	string		Position of the device.
powerSaveRecoveryTime	integer		Specifies the actual number of seconds required by the device to resume its normal operational state from the current power saving mode. This value is zero if either the power saving mode has not been activated or no power save control is supported
antiFraudModule	string		Specifies the state of the anti-fraud module
encryptionState (Required)	string		Specifies the state of the encryption module
certificateState (Required)	string		Specifies the state of the public verification or encryption key in the PIN certificate modules

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "device": "online",
    "extra": [
      "string"
    ],
    "guidLights": [
      {}
    ],
    "devicePosition": "inposition",
    "powerSaveRecoveryTime": 0,
    "antiFraudModule": "notSupp",
    "encryptionState": "ready",
    "certificateState": "unknown"
  }
}
```

Event Messages

KeyManagement.GetCapabilities

Description

This command is used to retrieve the capabilities of the device.

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000"
  }
}
```

Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error
class	string		Specifies the logical service class
compound	boolean		Specifies whether the logical device is part of a compound physical device
extra	array		Specifies a list of vendor-specific, or any other extended, information. The information is returned as a series of "key=value" strings so that it is easily extendable by Service Providers
guidLights	array		Specifies which guidance lights are available
guidLights.flashRate	object		Indicates which flash rates are supported by the guidelight.
guidLights.flashRate.slow	boolean		The light can blink slowly.
guidLights.flashRate.medium	boolean		The light can blink medium frequency.
guidLights.flashRate.quick	boolean		The light can blink quickly.
guidLights.flashRate.continuous	boolean		The light can be continuous (steady).

Name	Type	Default	Description
guidLights.color	object		Indicates which colors are supported by the guidelight.
guidLights.color.red	boolean		The light can be red.
guidLights.color.green	boolean		The light can be green.
guidLights.color.yellow	boolean		The light can be yellow.
guidLights.color.blue	boolean		The light can be blue.
guidLights.color.cyan	boolean		The light can be cyan.
guidLights.color.magenta	boolean		The light can be magenta.
guidLights.color.white	boolean		The light can be white.
guidLights.direction	object		Indicates which directions are supported by the guidelight.
guidLights.direction.entry	boolean		The light can indicate entry.
guidLights.direction.exit	boolean		The light can indicate exit.
powerSaveControl	boolean		Specifies whether power saving control is available
antiFraudModule	boolean		Specifies whether the anti-fraud module is available
synchronizableCommands	array		list of commands support synchronization.
keyNum (Required)	integer		Number of the keys which can be stored in the encryption/decryption module
idKey (Required)	object		Specifies if key owner identification (in commands referenced as lpxIdent), which authorizes access to the encryption module, is required. A zero value is returned if the encryption module does not support this capability
idKey.initialization	boolean		ID key is returned by the Initialization command
idKey.import	boolean		ID key is required as input for the ImportKey and DeriveKey command
keyCheckModes (Required)	object		Specifies the key check modes that are supported to check the correctness of an imported key value
keyCheckModes.self	boolean		The key check value is created by an encryption of the key with itself. For a double-length or triple-length key the kcv is generated using 3DES encryption using the first 8 bytes of the key as the source data for the encryption
keyCheckModes.zero	boolean		The key check value is created by encrypting a zero value with the key.
hsmVendor	string		Identifies the hsm Vendor. hsmVendor is an empty string or this field is not set when the hsm Vendor is unknown or the HSM is not supported
rsaAuthenticationScheme (Required)	boolean		Specifies which type of Remote Key Loading/Authentication
rsaAuthenticationScheme.2partySig	boolean		Two-party Signature based authentication.
rsaAuthenticationScheme.3partyCert	boolean		Three-party Certificate based authentication.
rsaAuthenticationScheme.3partyCertTr34	boolean		Three-party Certificate based authentication described by X9 TR34-2012.
rsaSignatureAlgorithm (Required)	object		Specifies which type of rsa Signature Algorithm
rsaSignatureAlgorithm.pkcs1V15	boolean		pkcs1V15 Signatures supported.
rsaSignatureAlgorithm.pss	boolean		pss Signatures supported.
rsaCryptAlgorithm (Required)	object		Specifies which type of rsa Encipherment Algorith
rsaCryptAlgorithm.pkcs1V15	boolean		pkcs1V15 algorithm supported.
rsaCryptAlgorithm.oaep	boolean		oaep algorithm supported.

Name	Type	Default	Description
rsaKeyCheckMode (Required)	object		Specifies which algorithm/method used to generate the public key check value/thumb print
rsaKeyCheckMode.sha1	boolean		sha1 is supported as defined in Ref. 3.
rsaKeyCheckMode.sha256	boolean		sha256 is supported as defined in ISO/IEC 10118-3:2004 and FIPS 180-2.
signatureScheme (Required)	object		Specifies which capabilities are supported by the Signature scheme
signatureScheme.genRsaKeyPair	boolean		Specifies if the Service Provider supports the rsa Signature Scheme GenerateRSAKeyPair and ExportRSAEPPSignedItem commands.
signatureScheme.randomNumber	boolean		Specifies if the Service Provider returns a random number from the StartKeyExchange GE command within the rsa Signature Scheme.
signatureScheme.exportEppId	boolean		Specifies if the Service Provider supports exporting the EPP Security Item within the rsa Signature Scheme.
signatureScheme.enhancedRkl	boolean		Specifies that the Service Provider supports the Enhanced Signature Remote Key Scheme. This scheme allows the customer to manage their own public keys independently of the Signature Issuer. When this mode is supported then the key loaded signed with the Signature Issuer key is the host root public key PKROOT, rather than PKHOST.
emvImportSchemes (Required)	object		Identifies the supported emv Import Scheme(s)
emvImportSchemes.plainCA	boolean		A plain text CA public key is imported with no verification.
emvImportSchemes.chksumCA	boolean		A plain text CA public key is imported using the EMV 2000 verification algorithm.
emvImportSchemes.epiCA	boolean		A CA public key is imported using the selfsign scheme defined in the Europay International, epi CA Module Technical - Interface specification.
emvImportSchemes.issuer	boolean		An Issuer public key is imported as defined in EMV 2000 Book II
emvImportSchemes.icc	boolean		An ICC public key is imported as defined in EMV 2000 Book II
emvImportSchemes.iccPin	boolean		An ICC PIN public key is imported as defined in EMV 2000 Book II
emvImportSchemes.pkcsv15CA	boolean		A CA public key is imported and verified using a signature generated with a private key for which the public key is already loaded.
emvHashAlgorithm (Required)	object		Specifies which hash algorithm is supported for the calculation of the HASH.
emvHashAlgorithm.sha1Digest	boolean		The SHA 1 digest algorithm is supported by the Digest command.
emvHashAlgorithm.sha256Digest	boolean		The SHA 256 digest algorithm, as defined in ISO/IEC 10118-3:2004 and FIPS 180-2, is supported by the Digest command.
keyBlockImportFormats (Required)	object		Supported key block formats
keyBlockImportFormats.ansTr31KeyBlock	boolean		Supports ANS TR-31A Keyblock format key import.
keyBlockImportFormats.ansTr31KeyBlockB	boolean		Supports ANS TR-31B Keyblock format key import.
keyBlockImportFormats.ansTr31KeyBlockC	boolean		Supports ANS TR-31C Keyblock format key import.
keyImportThroughParts (Required)	boolean		Specifies whether the device is capable of importing keys in multiple parts. TRUE means the device supports the key import in multiple parts
desKeyLength (Required)	object		Specifies which length of DES keys are supported.
desKeyLength.single	boolean		8 byte DES keys are supported.
desKeyLength.double	boolean		16 byte DES keys are supported.
desKeyLength.triple	boolean		24 byte DES keys are supported.
certificateTypes (Required)	object		Specifies supported certificate types
certificateTypes.encKey	boolean		Supports the EPP public encryption certificate.
certificateTypes.verificationKey	boolean		Supports the EPP public verification certificate.

Name	Type	Default	Description
certificateTypes.hostKey	boolean		Supports the Host public certificate.
loadCertOptions (Required)	array		Specifying the options supported by the LoadCertificate command
loadCertOptions.signer	string		Specifies the signers supported by the LoadCertificate command.
loadCertOptions.option	object		Specifies the load options supported by the LoadCertificate command
loadCertOptions.option.newHost	boolean		Load a new Host certificate, where one has not already been loaded.
loadCertOptions.option.replaceHost	boolean		Replace the epp to a new Host certificate, where the new Host certificate is signed by signer.
crklLoadOptions (Required)	object		Supported options to load the Key Transport Key using the Certificate Remote Key Loading protocol.
crklLoadOptions.noRandom	boolean		Import a Key Transport Key without generating and using a random number.
crklLoadOptions.noRandomCrl	boolean		Import a Key Transport Key with a Certificate Revocation List appended to the input message. A random number is not generated nor used.
crklLoadOptions.random	boolean		Import a Key Transport Key by generating and using a random number.
crklLoadOptions.randomCrl	boolean		Import a Key Transport Key with a Certificate Revocation List appended to the input parameter. A random number is generated and used.
restrictedKeyEncKeySupport (Required)	array		A array of object specifying the loading methods that support the RestrictedKeyEncKey usage flag and the allowable usage flag combinations.
restrictedKeyEncKeySupport.loadingMethod	string		Specifies the loading methods supported.
restrictedKeyEncKeySupport.uses	object		Specifies one or more usage flags that can be used in combination with the RestrictedKeyEncKey
restrictedKeyEncKeySupport.uses.crypt	boolean		Key isused for encryption and decryption.
restrictedKeyEncKeySupport.uses.function	boolean		Key is used for Pin block creation.
restrictedKeyEncKeySupport.uses.macing	boolean		Key is using for macing.
restrictedKeyEncKeySupport.uses.pinlocal	boolean		Key is used only for local PIN check.
restrictedKeyEncKeySupport.uses.svenckey	boolean		Key is used as cbc start Value encryption key.
restrictedKeyEncKeySupport.uses.pinremote	boolean		Key is used only for PIN block creation.
symmetricKeyManagementMethods (Required)	object		Specifies the Symmentric Key Management modes
symmetricKeyManagementMethods.fixedKey	boolean		This method of key management uses fixed keys for transaction processing.
symmetricKeyManagementMethods.masterKey	boolean		This method uses a hierarchy of Key Encrypting Keys and Transaction Keys. The highest level of Key Encrypting Key is known as a Master Key. Transaction Keys are distributed and replaced encrypted under a Key Encrypting Key.
symmetricKeyManagementMethods.tdesDukpt	boolean		This method uses TDES Derived Unique Key Per Transaction (see reference 45).
keyAttributes (Required)	array		Array of attributes supported by ImportKey command for the key to be loaded.
keyAttributes.keyUsage (Required)	string		Specifies the Key usages supported by Import Key command.
keyAttributes.propkeyUsage	integer	0	if keyUsage is set to numeric, proprietary value is set. otherwise this field is not required.
keyAttributes.algorithm (Required)	string		Specifies the encryption algorithms supported by the import command.
keyAttributes.propAlgorithm	integer	0	if algorithm is set to numeric, proprietary value is set. otherwise this field is not required.
keyAttributes.modeOfUse (Required)	string		Specifies the encryption modes supported by import key.
keyAttributes.propmodeOfUse	integer	0	if modeOfUse is set to numeric, proprietary value is set. otherwise this field is not required.
keyAttributes.cryptoMethod (Required)	string		Specifies the cryptographic methods supported by the import command. For attributes, this parameter is 0, because the key being imported is not being used yet to perform a cryptographic method

Name	Type	Default	Description
decryptAttributes (Required)	array		Array of attributes supported by the Import command for the key used to decrypt or unwrap the key being imported.
decryptAttributes.algorithm (Required)	string		Specifies the encryption algorithms supported by the import command.
decryptAttributes.propAlgorithm	integer	0	if algorithm is set to numeric, proprietary value is set. otherwise this field is not required.
decryptAttributes.cryptoMethod (Required)	string		This parameter specifies the cryptographic method that will be used with the encryption algorithm specified by bAlgorithm.
verifyAttributes (Required)	array		Array of attributes supported by Import command for the key used for verification before importing the key.
verifyAttributes.keyUsage (Required)	string		Specifies the key usages supported by the import command.
verifyAttributes.propkeyUsage	integer	0	if keyUsage is set to numeric, proprietary value is set. otherwise this field is not required.
verifyAttributes.algorithm (Required)	string		Specifies the encryption algorithms supported by the import command.
verifyAttributes.propAlgorithm	integer	0	if algorithm is set to numeric, proprietary value is set. otherwise this field is not required.
verifyAttributes.modeOfUse (Required)	string		Specifies the encryption modes supported by the import command.
verifyAttributes.propmodeOfUse	integer	0	if modeOfUse is set to numeric, proprietary value is set. otherwise this field is not required.
verifyAttributes.cryptoMethod (Required)	string		This parameter specifies the cryptographic method that will be used with encryption algorithm.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "class": "IDC",
    "compound": true,
    "extra": [
      "string"
    ],
    "guidLights": [
      {
        "flashRate": {
          "slow": true,
          "medium": true,
          "quick": true,
          "continuous": true
        },
        "color": {
          "red": true,
          "green": true,
          "yellow": true,
          "blue": true,
          "cyan": true,
          "magenta": true,
          "white": true
        },
        "direction": {
          "entry": true,
          "exit": true
        }
      }
    ],
    "powerSaveControl": true,
    "antiFraudModule": true,
    "synchronizableCommands": [
      "string"
    ],
    "keyNum": 0,
    "idKey": {
      "initialization": true,
      "import": true
    },
    "keyCheckModes": {
      "self": true,
      "zero": true
    },
    "hsmVendor": "string",
```

```
    "rsaAuthenticationScheme": true,
    "rsaSignatureAlgorithm": {
      "pkcs1V15": true,
      "pss": true
    },
    "rsaCryptAlgorithm": {
      "pkcs1V15": true,
      "oaep": true
    },
    "rsaKeyCheckMode": {
      "sha1": true,
      "sha256": true
    },
    "signatureScheme": {
      "genRsaKeyPair": true,
      "randomNumber": true,
      "exportEppId": true,
      "enhancedRkl": true
    },
    "emvImportSchemes": {
      "plainCA": true,
      "chksumCA": true,
      "epiCA": true,
      "issuer": true,
      "icc": true,
      "iccPin": true,
      "pkcsv15CA": true
    },
    "emvHashAlgorithm": {
      "sha1Digest": true,
      "sha256Digest": true
    },
    "keyBlockImportFormats": {
      "ansTr31KeyBlock": true,
      "ansTr31KeyBlockB": true,
      "ansTr31KeyBlockC": true
    },
    "keyImportThroughParts": true,
    "desKeyLength": {
      "single": true,
      "double": true,
      "triple": true
    },
    "certificateTypes": {
      "encKey": true,
      "verificationKey": true,
      "hostKey": true
    },
    "loadCertOptions": [
      {
        "signer": "certHost",
        "option": {
          "newHost": true,
          "replaceHost": true
        }
      }
    ],
    "crklLoadOptions": {
      "noRandom": true,
      "noRandomCrl": true,
      "random": true,
      "randomCrl": true
    },
    "restrictedKeyEncKeySupport": [
      {
        "loadingMethod": "rsaAuth2partySig",
        "uses": {
          "crypt": true,
          "function": true,
          "macing": true,
          "pinlocal": true,
          "svenckey": true,
          "pinremote": true
        }
      }
    ],
    "symmetricKeyManagementMethods": {
      "fixedKey": true,
      "masterKey": true,
      "tdesDukpt": true
    },
    "keyAttributes": [],
    "decryptAttributes": [],
    "verifyAttributes": []
  }
}
```


Event Messages

KeyManagement.GetKeyDetail

Description

This command returns extended detailed information about the keys in the encryption module, including des,dukpt,aes,rsa private and public keys. This command will also return information on all keys loaded during manufacture that can be used by applications. Details relating to the keys loaded using OPT (via the ZKA ProtIsoPs protocol) are retrieved using the ZKA hsmLdi protocol. These keys are not reported by this command.

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
keyName	string		Name of the key for which detailed information is requested.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "keyName": "string"
  }
}
```

Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error
keyDetail	array		
keyDetail.keyName	string		Specifies the name of the key.
keyDetail.generation	integer		Specifies the generation of the key as bcd value. Different generations might correspond to different environments (e.g. test or production environment). The content is vendor specific. This value will be 0xFF if no such information is available for the key.
keyDetail.version	integer		Specifies the version of the key (the year in which the key is valid, e.g. 01 for 2001) as bcd value. This value will be 0xFF if no such information is available for the key.
keyDetail.activatingDate	array		Specifies the date when the key is activated as bcd value in the format YYYYMMDD. This value will be 0xFFFFFFFF if no such information is available for the key.

Name	Type	Default	Description
keyDetail.expiryDate	array		Specifies the date when the key expires as bcd value in the format YYYYMMDD. This value will be 0xFFFFFFFF if no such information is available for the key.
keyDetail.loaded (Required)	object		Specifies whether the key has been loaded (imported from Application or locally from Operator).
keyDetail.loaded.no	boolean		The key is not loaded or not ready to be used in cryptographic operations.
keyDetail.loaded.yes	boolean		The key is loaded and ready to be used in cryptographic operations.
keyDetail.loaded.unknown	boolean		The State of the key is unknown.
keyDetail.loaded.construct	boolean		The key is under construction, meaning that at least one key part has been loaded but the key is not activated and ready to be used in other cryptographic operations. This flag can only be returned in combination with loaded_no.
keyDetail.keyBlockInfo	object		Contains the key block header of keys imported within an ANS TR-31 key block. This data is encoded in the same format that it was imported in, and contains all mandatory and optional header fields. keyBlockHeader is NULL if the key was not imported within a key block or has not been loaded yet. The use field provides an accurate summary of the key use, but the use defined within the key block header is more precise
keyDetail.keyBlockInfo.keyUsage (Required)	string		Specifies the intended function of the key. See [Reference 35. ANS X9 TR-31 2018] for all possible values.
keyDetail.keyBlockInfo.algorithm (Required)	string		Specifies the algorithm for which the key may be used. See [Reference 35. ANS X9 TR-31 2018] for all possible values.
keyDetail.keyBlockInfo.modeOfUse (Required)	string		Specifies the operation that the key may perform. See [Reference 35. ANS X9 TR-31 2018] for all possible values.
keyDetail.keyBlockInfo.keyVersionNumber	string		Specifies a two-digit ASCII character version number, which is optionally used to indicate that contents of the key block are a component, or to prevent re-injection of old keys. See [Reference 35. ANS X9 TR-31 2018] for all possible values.
keyDetail.keyBlockInfo.exportability	string		Specifies whether the key may be transferred outside of the cryptographic domain in which the key is found. See [Reference 35. ANS X9 TR-31 2018] for all possible values.
keyDetail.keyBlockInfo.optionalBlockHeader	string		Contains any optional header blocks, as defined in [Reference 35. ANS X9 TR-31 2018]. This value will be NULL if there are no optional block headers.
keyDetail.keyBlockInfo.keyLength (Required)	integer		Specifies the length, in bits, of the key. 0 if the key length is unknown.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "keyDetail": [
      {
        "keyName": "string",
        "generation": 0,
        "version": 0,
        "activatingDate": [
          0
        ],
        "expiryDate": [
          0
        ],
        "loaded": {
          "no": true,
          "yes": true,
          "unknown": true,
          "construct": true
        },
        "keyBlockInfo": {
          "keyUsage": "string",
          "algorithm": "string",
          "modeOfUse": "string",
          "keyVersionNumber": "string",
          "exportability": "string",
          "optionalBlockHeader": "string",
          "keyLength": 0
        }
      }
    ]
  }
}
```


Event Messages

KeyManagement.Initialization

Description

The encryption module must be initialized before any encryption function can be used. Every call to Initialization destroys all application keys that have been loaded or imported; it does not affect those keys loaded during manufacturing. Usually this command is called by an operator task and not by the application program. Public keys imported under the rsa Signature based remote key loading scheme when public key deletion authentication is required will not be affected. However, if this command is requested in authenticated mode, public keys that require authentication for deletion will be deleted. This includes public keys imported under either the rsa Signature based remote key loading scheme or the TR34 RSA Certificate based remote key loading scheme. Initialization also involves loading â€œinitial' application keys and local vendor dependent keys. These can be supplied, for example, by an operator through a keyboard, a local configuration file, remote RSA key management or possibly by means of some secure hardware that can be attached to the device. The application 'initial' keys would normally get updated by the application during a ImportKeyEx command as soon as possible. Local vendor dependent static keys (e.g. storage, firmware and offset keys) would normally be transparent to the application and by definition cannot be dynamically changed. Where initial keys are not available immediately when this command is issued (i.e. when operator intervention is required), the Service Provider returns accessDenied and the application must await the InitializedEvent. During initialization an optional encrypted ID key can be stored in the HW module. The ID key and the corresponding encryption key can be passed as parameters; if not, they are generated automatically by the encryption module. The encrypted ID is returned to the application and serves as authorization for the key import function. The Capabilities command indicates whether or not the device will support this feature. This function also resets the hsm terminal data, except session key index and trace number. This function resets all certificate data and authentication public/private keys back to their initial states at the time of production (except for those public keys imported under the rsa Signature based remote key loading scheme when public key deletion authentication is required). Key-pairs created with GenerateRSAKeyPair are deleted. Any keys installed during production, which have been permanently replaced, will not be reset. Any Verification certificates that may have been loaded must be reloaded. The Certificate state will remain the same, but the LoadCertificate or ReplaceCertificate commands must be called again. When multiple ZKA HSMs are present, this command deletes all keys loaded within all ZKA logical HSMs.

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.

Name	Type	Default	Description
ident	string		The value of the ID key. this field is not required if an indent is not required.
key	string		The value of the encryption key formatted in base64. this field is not required if no specific key name required.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "ident": "string",
    "key": "string"
  }
}
```


Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error
identification	string		The value of the ID key encrypted by the encryption key formatted in base64. This value can be used as authorization for the ImportKey command, this field is not set if no authorization required.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "identification": "string"
  }
}
```


Event Messages

KeyManagement.DeriveKey

Description

The encryption key in the secure key buffer or passed by the application is loaded in the encryption module. The key can be passed in clear text mode or encrypted with an accompanying 'key encryption key'. A key can be loaded in multiple unencrypted parts by combining the construct or secureConstruct value with the final usage flags within the use field. If the construct flag is used then the application must provide the key data through the value parameter, If secureConstruct is used then the encryption key part in the secure key buffer previously populated with the SecureKeyEntry command is used and value is ignored. Key parts loaded with the secureConstruct flag can only be stored once as the encryption key in the secure key buffer is no longer available after this command has been executed. The construct and secureConstruct construction flags cannot be used in combination.

Command Message

Message Header

Name	Type	Default	Description

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
derivationAlgorithm (Required)	integer		Specifies the algorithm that is used for derivation.
key (Required)	string		Specifies the name where the derived key will be stored.
keyGenKey (Required)	string		Specifies the name of the key generating key that is used for the derivation.
startValueKey	string		Specifies the name of the stored key used to decrypt the startValue to obtain the Initialization Vector. If this field is not set, startValue is used as the Initialization Vector.
startValue (Required)	string		des initialization vector for the encryption step within the derivation.
padding (Required)	integer		Specifies the padding character for the encryption step within the derivation. The valid range is 0x00 to 0xFF
inputData (Required)	string		Data to be used for key derivation.
ident	string		Specifies the key owner identification. It is a handle to the encryption module and is returned to the application in the intialization command. See idKey in Capabilities for whether this value is required. If not required, this field should not be set. The use of this parameter is vendor dependent.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "derivationAlgorithm": 0,
    "key": "string",
    "keyGenKey": "string",
    "startValueKey": "string",
    "startValue": "string",
    "padding": 0,
    "inputData": "string",
    "ident": "string"
  }
}
```


Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string"
  }
}
```

Event Messages

KeyManagement.Reset

Description

Sends a service reset to the Service Provider.

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000"
  }
}
```

Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string"
  }
}
```

Event Messages

KeyManagement.ImportKey

Description

The encryption key passed by the application is loaded in the encryption module. For secret keys, the key must be passed encrypted with an accompanying "key encrypting key" or "key block protection key". For public keys, they key is not required to be encrypted but is required to have verification data in order to be loaded. This command can also be used to delete a key without authentication. Where an authenticated delete is required, the StartAuthenticate and Authenticate commands should be used.

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
key (Required)	string		Specifies the name of key being loaded or deleted.
keyAttributes	object		This parameter specifies the encryption algorithm, cryptographic method, and mode to be used for the key imported by this command. For a list of valid values see the keyAttributes capability field. The values specified must be compatible with the key identified by key. This field must not set if the key specified by Key is to be deleted.
keyAttributes.keyUsage (Required)	string		Specifies the Key usages supported by Import Key command.
keyAttributes.propkeyUsage	integer	0	if keyUsage is set to numeric, proprietary value is set. otherwise this field is not required.
keyAttributes.algorithm (Required)	string		Specifies the encryption algorithms supported by the import command.
keyAttributes.propAlgorithm	integer	0	if algorithm is set to numeric, proprietary value is set. otherwise this field is not required.
keyAttributes.modeOfUse (Required)	string		Specifies the encryption modes supported by import key.
keyAttributes.propmodeOfUse	integer	0	if modeOfUse is set to numeric, proprietary value is set. otherwise this field is not required.
keyAttributes.cryptoMethod (Required)	string		Specifies the cryptographic methods supported by the import command. For attributes, this parameter is 0, because the key being imported is not being used yet to perform a cryptographic method
value (Required)	string		Specifies the value of key to be loaded formatted in base64. If it is an rsa key the first 4 bytes contain the exponent and the following 128 the modulus.
decryptKey	string		Specifies the name of the key used to decrypt the key being loaded. If value contains a TR-31 key block, then decryptKey is the name of the key block protection key that is used to verify and decrypt the key block. This field is not required if the data in Value is not encrypted.Must be an empty if the key specified by key is to be deleted.
decryptMethod	string		Specifies the cryptographic method that shall be used with the key specified by decryptKey. The device shall use this method to decrypt the encrypted value in the value parameter. For a list of valid values see the cryptoMethod field in the decryptAttributes capability field. This field must not set if decryptKey is an empty string or the key specified by key is to be deleted. Must be an empty string if a keyblock is being imported, as the decrypt method is contained within the keyblock.

Name	Type	Default	Description
verificationData	string		Contains the data to be verified before importing. if this field is not set when no verification is needed before importing or deleting the key. Where an authenticated delete is required, the StartAuthenticate and Authenticate commands should be used.
verifyKey	string		Specifies the name of the previously loaded key which will be used to verify the verificationData. This field is not required when no verification is needed before importing or deleting the key.
verifyAttributes	object		This parameter specifies the encryption algorithm, cryptographic method, and mode to be used to verify this command or to generate verification output data. Verifying input data will result in no verification output data. For a list of valid values see the verifyAttributes capability fields. This field must not be set if verificationData is not required.
verifyAttributes.keyUsage (Required)	string		Specifies the key usages supported by the import command.
verifyAttributes.propkeyUsage	integer	0	if keyUsage is set to numeric, proprietary value is set. otherwise this field is not required.
verifyAttributes.algorithm (Required)	string		Specifies the encryption algorithms supported by the import command.
verifyAttributes.propAlgorithm	integer	0	if algorithm is set to numeric, proprietary value is set. otherwise this field is not required.
verifyAttributes.modeOfUse (Required)	string		Specifies the encryption modes supported by the import command.
verifyAttributes.propmodeOfUse	integer	0	if modeOfUse is set to numeric, proprietary value is set. otherwise this field is not required.
verifyAttributes.cryptoMethod (Required)	string		This parameter specifies the cryptographic method that will be used with encryption algorithm.
vendorAttributes	string		Specifies the vendor attributes of the key to be imported. Refer to vendor documentation for details. If no vendor attributes are used, then this field must not be set.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "key": "string",
    "keyAttributes": {
      "keyUsage": "b0",
      "propkeyUsage": 0,
      "algorithm": "a",
      "propAlgorithm": 0,
      "modeOfUse": "b",
      "propmodeOfUse": 0,
      "cryptoMethod": "string"
    },
    "value": "string",
    "decryptKey": "string",
    "decryptMethod": "string",
    "verificationData": "string",
    "verifyKey": "string",
    "verifyAttributes": {
      "keyUsage": "m0",
      "propkeyUsage": 0,
      "algorithm": "a",
      "propAlgorithm": 0,
      "modeOfUse": "v",
      "propmodeOfUse": 0,
      "cryptoMethod": "none"
    },
    "vendorAttributes": "string"
  }
}
```


Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
verificationData	string		The verification data. This field is not set if there is no verification data.
verifyAttributes	object		This parameter specifies the encryption algorithm, cryptographic method, and mode used to verify this command For a list of valid values see the verifyAttributes capability fields.This field is not set if there is no verification data.
verifyAttributes.keyUsage (Required)	string		Specifies the key usages supported by the import command.
verifyAttributes.propkeyUsage	integer	0	if keyUsage is set to numeric, proprietary value is set. otherwise this field is not required.
verifyAttributes.algorithm (Required)	string		Specifies the encryption algorithms supported by the import command.
verifyAttributes.propAlgorithm	integer	0	if algorithm is set to numeric, proprietary value is set. otherwise this field is not required.
verifyAttributes.modeOfUse (Required)	string		Specifies the encryption modes supported by the import command.
verifyAttributes.propmodeOfUse	integer	0	if modeOfUse is set to numeric, proprietary value is set. otherwise this field is not required.
verifyAttributes.cryptoMethod (Required)	string		This parameter specifies the cryptographic method that will be used with encryption algorithm.
keyLength (Required)	integer		Specifies the length, in bits, of the key. 0 is the key length is unknown.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "verificationData": "string",
    "verifyAttributes": {
      "keyUsage": "m0",
      "propkeyUsage": 0,
      "algorithm": "a",
      "propAlgorithm": 0,
      "modeOfUse": "v",
      "propmodeOfUse": 0,
      "cryptoMethod": "none"
    },
    "keyLength": 0
  }
}
```


Event Messages

KeyManagement.ExportRSAIssuerSignedItem

Description

This command is used to export data elements from the device, which have been signed by an offline Signature Issuer. This command is used when the default keys and Signature Issuer signatures, installed during manufacture, are to be used for remote key loading. This command allows the following data items are to be exported:

- The Security Item which uniquely identifies the device. This value may be used to uniquely identify a device and therefore confer trust upon any key or data obtained from this device.
- The rsa public key component of a public/private key pair that exists within the device. These public/private key pairs are installed during manufacture. Typically, an exported public key is used by the host to encipher the symmetric key.

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
exportItemType	string		Defines the type of data item to be exported from the device.
name	string		Specifies the name of the public key to be exported. The private/public key pair was installed during manufacture; see section 8.1.8 (Default Keys and Security Item loaded during manufacture) for a definition of these default keys. If name is an empty string, then the default EPP public key that is used for symmetric key encryption is exported.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "exportItemType": "eppId",
    "name": "string"
  }
}
```


Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error
value	string		If a public key was requested then value contains the PKCS #1 formatted rsa public key represented in DER encoded ASN.1 format. If the security item was requested then value contains the PIN's Security Item, which may be vendor specific.
rsaSignatureAlgorithm	string		Specifies the algorithm used to generate the Signature returned in signature
signature	string		Specifies the RSA signature of the data item exported formatted in base64. An empty sting can be returned when key signature are not supported.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "value": "string",
    "rsaSignatureAlgorithm": "na",
    "signature": "string"
  }
}
```


Event Messages

KeyManagement.GenerateRSAKeyPair

Description

This command will generate a new rsa key pair. The public key generated as a result of this command can subsequently be obtained by calling ExportRSAEPPSignedItem.

The newly generated key pair can only be used for the use defined in the dwUse flag. This flag defines the use of the private key; its public key can only be used for the inverse function.

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
key	string		Specifies the name of the new key-pair to be generated. Details of the generated key-pair can be obtained through the KeyDetail command.
use	string		Specifies what the private key component of the key pair can be used for. The public key part can only be used for the inverse function. For example, if the rsaPrivateSign use is specified, then the private key can only be used for signature generation and the partner public key can only be used for verification.
modulusLength	integer		Specifies the number of bits for the modulus of the rsa key pair to be generated. When zero is specified then the device will be responsible for defining the length.
exponentValue	string		Specifies the value of the exponent of the rsa key pair to be generated

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "key": "string",
    "use": "rsaPrivate",
    "modulusLength": 0,
    "exponentValue": "default"
  }
}
```


Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string"
  }
}
```

Event Messages

KeyManagement.ExportRSAEPPSignedItem

Description

This command is used to export data elements from the device that have been signed by a private key within the epp. This command is used in place of the ExportRSAIssuerSignedItem command, when a private key generated within the device is to be used to generate the signature for the data item. This command allows an application to define which of the following data items are to be exported

- The Security Item which uniquely identifies the device. This value may be used to uniquely identify a device and therefore confer trust upon any key or data obtained from this device.
- The rsa public key component of a public/private key pair that exists within the device.

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
exportItemType	string		Defines the type of data item to be exported from the device
name	string		Specifies the name of the public key to be exported. This can either be the name of a key-pair generated through GenerateRsaKeyPair or the name of one of the default key-pairs installed during manufacture.
sigKey	string		Specifies the name of the private key to use to sign the exported item.
signatureAlgorithm	string		Specifies the algorithm to use to generate the Signature returned in both the selfSignature and signature fields.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "exportItemType": "eppId",
    "name": "string",
    "sigKey": "string",
    "signatureAlgorithm": "na"
  }
}
```

Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.

Name	Type	Default	Description
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error
value	string		If a public key was requested then value contains the pkcs #1 formatted rsa Public Key represented in DER encoded ASN.1 format. If the security item was requested then value contains the PIN's Security Item, which may be vendor specific
selfSignature	string		If a public key was requested then selfSignature contains the rsa signature of the public key exported, generated with the key-pair's private component. An empty string can be returned when key selfSignatures are not supported/required.
signature	string		Specifies the rsa signature of the data item exported. An empty string can be returned when signatures are not supported/required

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "value": "string",
    "selfSignature": "string",
    "signature": "string"
  }
}
```


Event Messages

KeyManagement.GetCertificate

Description

This command is used to read out the encryptorâ€™s certificate, which has been signed by the trusted Certificate Authority and is sent to the host. This command only needs to be called once if no new Certificate Authority has taken over. The output of this command will specify in the pkcs #7 message the resulting Primary or Secondary certificate

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
getCertificate (Required)	string		Specifies which public key certificate is requested. If the Status command indicates Primary Certificates are accepted, then the Primary Public Encryption Key or the Primary Public Verification Key will be read out. If the Status command indicates Secondary Certificates are accepted, then the Secondary Public Encryption Key or the Secondary Public Verification Key will be read out.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "getCertificate": "enckey"
  }
}
```

Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error
certificate (Required)	string		Contains the certificate that is to be loaded represented in DER encoded ASN.1 notation. This data should be in a binary encoded pkcs #7 using the degenerate certificate only case of the signed-data content type in which the inner content's data file is omitted and there are no signers

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "certificate": "string"
  }
}
```

Event Messages

KeyManagement.ReplaceCertificate

Description

This command is used to replace the existing primary or secondary Certificate Authority certificate already loaded into the KeyManagement. This operation must be done by an Initial Certificate Authority or by a Sub-Certificate Authority. These operations will replace either the primary or secondary Certificate Authority public verification key inside of the KeyManagement. After this command is complete, the application should send the LoadCertificate and GetCertificate commands to ensure that the new HOST and the encryptor have all the information required to perform the remote key loading process

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
replaceCertificate (Required)	string		The pkcs # 7 message that will replace the current Certificate Authority formatted in base64. The outer content uses the signedData content type, the inner content is a degenerate certificate only content containing the new ca certificate and Inner Signed Data type The certificate should be in a format represented in DER encoded ASN.1 notation.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "replaceCertificate": "string"
  }
}
```


Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error
newCertificateData (Required)	string		A pkcs #7 using a Digested-data content type formatted in base64. The digest parameter should contain the thumb print value.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "newCertificateData": "string"
  }
}
```


Event Messages

KeyManagement.StartKeyExchange

Description

This command is used to start communication with the host, including transferring the hostâ€™s Key Transport Key, replacing the Host certificate, and requesting initialization remotely. This output value is returned to the host and is used in the ImportRSASignedDESKey, LoadCertificate, and ImportRSAEncipheredPKCS7Key commands to verify that the encryptor is talking to the proper host. The ImportRSAEncipheredRKCS7Key command end the key exchange process

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.

Name	Type	Default	Description
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000"
  }
}
```


Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error
randomItem	string		A randomly generated number created by the encryptor formatted in base 64. If the device does not support random number generation and verification, a zero length random number is returned and an empty string is returned.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "randomItem": "string"
  }
}
```


Event Messages

KeyManagement.GenerateKCV

Description

This command returns the Key Check Value (kcv) for the specified key.

Command Message

Message Header

Name	Type	Default	Description

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
key (Required)	string		Specifies the name of key that should be used to generate the kcv.
keyCheckMode (Required)	string		Specifies the mode that is used to create the key check value.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "key": "string",
    "keyCheckMode": "self"
  }
}
```


Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error
kcv (Required)	string		Contains the key check value data that can be used for verification of the key formatted in base64.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "kcv": "string"
  }
}
```


Event Messages

KeyManagement.LoadCertificate

Description

This command is used to load a host certificate to make remote key loading possible. This command can be used to load a host certificate when there is not already one present in the encryptor as well as replace the existing host certificate with a new host certificate. The type of certificate (Primary or Secondary) to be loaded will be embedded within the actual certificate structure.

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
loadOption (Required)	string		Specifies the method to use to load the certificate
signer (Required)	string		Specifies the signer of the certificate to be loaded
certificateData (Required)	string		The structure that contains the certificate that is to be loaded represented in DER encoded ASN.1 notation. For loadNewHost, this data should be in a binary encoded PKCS #7 using the 'degenerate certificate only' case of the signed-data content type in which the inner content's data file is omitted and there are no signers. For replaceHost, the message has an outer signedData content type with the signerInfo encryptedDigest field containing the signature of signer. The inner content is binary encoded pkcs#7 using the degenerate certificate. The optional crl field may or may not be included in the pkcs#7 signedData structure

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "loadOption": "newHost",
    "signer": "certHost",
    "certificateData": "string"
  }
}
```


Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error
rsaKeyCheckMode (Required)	object		Defines algorithm/method used to generate the public key check value/thumb print. The check value can be used to verify that the public key has been imported correctly
rsaData	string		A pkcs #7 structure using a Digested-data content type formatted in base64. The digest parameter should contain the thumb print value calculated by the algorithm specified by rsaKeyCheckMode. If rsaKeyCheckMode is none, then this field is not be set or an empty string.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "rsaKeyCheckMode": "none",
    "rsaData": "string"
  }
}
```


Event Messages

KeyManagement.StartAuthenticate

Description

This command is used to retrieve the data that needs to be signed and hence provided to the Authenticate command in order to perform an authenticated action on the device. If this command returns data to be signed then the Authenticate command must be used to call the command referenced by startAuthenticate. Any attempt to call the referenced command without using the Authenticate command, if authentication is required, shall result in AuthRequired.

Command Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
timeout	integer	0	Timeout in milliseconds for the command to complete. If set to zero, the command will not timeout but can be cancelled.
commandId (Required)	string		The command name to which authentication is being applied
inputData	object		A payload to the input data of the command referred to by the execution command.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "commandId": "string",
    "inputData": {}
  }
}
```


Completion Message

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description

Name	Type	Default	Description
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error
internalCmdResult (Required)	string		Result from the command referenced by execution command. If the data within payload is invalid or cannot be used for some reason, then hInternalCmdResult will return an error but the result of this command will be ok.
dataToSign	string		The data that must be signed by one of the authorities indicated by signers before the command referenced by execution command can be executed. If the command specified by execution command does not require authentication, then this field is not set string and the command result is success.
signers (Required)	object		Specifies the allowed signers of the data as a combination
signers.none	boolean		Authentication is not required
signers.certhost	boolean		The data is signed by the current Host, using the RSA certificate-based scheme.
signers.sighost	boolean		The data is signed by the current Host, using the RSA signature-based scheme.
signers.ca	boolean		The data is signed by the Certificate Authority (CA).
signers.hl	boolean		The data is signed by the Higher Level (HL) Authority.
signers.tr34	boolean		The format of the data that was signed complies with the data defined in X9 TR342012 [Ref. 42]. This value can only be used in combination with the CERTHOST, CA or HL flags.
signers.cbcmac	boolean		A MAC is calculated over the data using lpsKey and the CBC MAC algorithm.
signers.cmac	boolean		A MAC is calculated over the data using lpsKey and the CMAC algorithm.
signers.reserved_1	boolean		Reserved for a vendor-defined signing method.
signers.reserved_2	boolean		Reserved for a vendor-defined signing method.
signers.reserved_3	boolean		Reserved for a vendor-defined signing method.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "internalCmdResult": "string",
    "dataToSign": "string",
    "signers": {
      "none": true,
      "certhost": true,
      "sighost": true,
      "ca": true,
      "hl": true,
      "tr34": true,
      "cbcmac": true,
      "cmac": true,
      "reserved_1": true,
      "reserved_2": true,
      "reserved_3": true
    }
  }
}
```


Event Messages

Unsolicited Events

KeyManagement.DevicePositionEvent

Description

This service event reports that the device has changed its position status.

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
Position	string		Position of the device

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "Position": "inposition"
  }
}
```

KeyManagement.PowerSaveChangeEvent

Description

This service event specifies that the power save recovery time has changed.

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
powerSaveRecoveryTime	integer		Specifies the actual number of seconds required by the device to resume its normal operational state. This value is zero if the device exited the power saving mode

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "powerSaveRecoveryTime": 0
  }
}
```

KeyManagement.InitializedEvent

Description

This event specifies that, as a result of a Initialization command, the encryption module is now initialized and the master key (where required) and any other initial keys are loaded; ready to import other keys

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
ident	string		The value of the ID key formatted in base 64. if not required, this field can not be set or an empty string
key	string		The value of the encryption key formatted in base 64. if not required, this field can not be set or an empty string

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "ident": "string",
    "key": "string"
  }
}
```


KeyManagement.IllegalKeyAccessEvent

Description

This event specifies that an error occurred accessing an encryption key. Possible situations for generating this event are listed in the description of IErrorCode.

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
keyName (Required)	string		Specifies the name of the key that caused the error.
errorCode (Required)	string		Specifies the type of illegal key access that occurred

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "keyName": "string",
    "errorCode": "keynotfound"
  }
}
```


KeyManagement.CertificateChangeEvent

Description

This event indicates that the certificate module state has changed from Primary to Secondary.

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
certificateChange (Required)	string		Specifies change of the certificate state inside of the KeyManagement.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "certificateChange": "secondary"
  }
}
```


Events

Common.PowerSaveChangeEvent

Description

This service event specifies that the power save recovery time has changed.

Message Header

Name	Type	Default	Description
requestId (Required)	string		Unique request identifier supplied by the client used to correlate the command with responses, events and completions. For Unsolicited Events the field will be empty.
type (Required)	string		The message type, either command, response, event or completion.
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
powerSaveRecoveryTime	integer		Specifies the actual number of seconds required by the device to resume its normal operational state. This value is zero if the device exited the power saving mode

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "powerSaveRecoveryTime": 0
  }
}
```