XFS4IOT Sample Messaging for Encryptor 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
- German ZKA GeldKarte (Deutsche Kreditwirtschaft)
- EMV Support
- French Cartes Bancaires
- Secure Key Entry
- Command Usage
- restrictedKeyEncKey key usage
- ImportKey command Input-Output Parameters
- Appendix–D (TR-31 Key Use)
- Appendix-E (DUKPT)

Commands

- Pinpad.SetGuidanceLight
- Pinpad.PowerSaveControl
- Pinpad.SynchronizeCommand
- Pinpad.Status
- Pinpad.Capabilities
- Pinpad.FuncKeyDetail
- Pinpad.HSMTData
- Pinpad.KeyDetail
- Pinpad.SecureKeyDetail
- Pinpad.QueryLogicalHSMDetail
- Pinpad.QueryPCIPTSDeviceId
- Pinpad.GetLayout
- Pinpad.Crypt
- Pinpad.GetPin
- Pinpad.GetData

- Pinpad.Initialization
- Pinpad.LocalDES
- Pinpad.LocalEuroCheque
- Pinpad.LocalVisa
- Pinpad.CreateOffset
- Pinpad.DeriveKey
- Pinpad.PresentIDC
- Pinpad.LocalBanksys
- Pinpad.Banksyslo
- Pinpad.Reset
- Pinpad.HSMSetTData
- Pinpad.SecureMsgSend
- Pinpad.SecureMsgReceive
- Pinpad.GetJournal
- Pinpad.ImportKey
- Pinpad.Enclo
- Pinpad.HSMInit
- Pinpad.ImportRSAPublicKey
- Pinpad.ExportRSAIssuerSignedItem
- Pinpad.ImportRSASignedDESKey
- Pinpad.GenerateRSAKeyPair
- Pinpad.ExportRSAEPPSignedItem
- Pinpad.GetCertificate
- Pinpad.ReplaceCertificate
- Pinpad.StartKeyExchange
- Pinpad.EMVImportPublicKey
- Pinpad.Digest
- Pinpad.SecureKeyEntry
- Pinpad.GenerateKCV
- Pinpad.MaintainPin
- Pinpad.KeypressBeep
- Pinpad.SetPinblockData
- Pinpad.SetLogicalHSM
- Pinpad.ImportKeyBlock
- Pinpad.LoadCertificate
- Pinpad.ImportRSAEncipheredPKCS7Key
- Pinpad.DefineLayout
- Pinpad.StartAuthenticate
- Pinpad.Authenticate
- Pinpad.GetPinblock
- Pinpad.LuxLoadAppKey
- Pinpad.LuxGenerateMac
- Pinpad.LuxCheckMac
- Pinpad.LuxBuildPinBlock
- Pinpad.LuxDecryptTDES
- Pinpad.LuxEncryptTDES
- Pinpad.CHNDigest
- Pinpad.CHNSetSm2Param
- Pinpad.CHNImportSM2PublicKey
- Pinpad.CHNSign
- Pinpad.CHNVerify
- Pinpad.CHNExportSm2IssuerSignedItem

- Pinpad.CHNGenerateSm2KeyPair
- Pinpad.CHNExportSm2EPPSignedItem
- Pinpad.CHNImportSm2SignedSm4Key

Unsolicited Events

- Pinpad.DevicePositionEvent
- Pinpad.PowerSaveChangeEvent
- Pinpad.StatusChangeEvent
- Pinpad.InitializedEvent
- Pinpad.IllegalKeyAccessEvent
- Pinpad.OPTRequiredEvent
- Pinpad.HSMTDataChangedEvent
- Pinpad.CertificateChangeEvent
- Pinpad.HSMChangedEvent

Events

- Common.PowerSaveChangeEvent
- Pinpad.KeyEvent
- Pinpad.EnterDataEvent

XFS4IOT Sample Messaging for Encryptor Draft 0.0.4

This section describes the general interface for the following functions:

- Administration of encryption devices
- Loading of encryption keys
- Encryption / decryption
- Entering Personal Identification Numbers (PINs)
- PIN verification
- PIN block generation (encrypted PIN)
- Clear text data handling
- Function key handling
- PIN presentation to chipcard
- Read and write safety critical Terminal Data from/to HSM
- HSM and Chipcard Authentication
- EMV 4.0 PIN blocks, EMV 4.0 public key loading, static and dynamic data verification If the PIN pad device has local display capability, display handling should be handled using the Text Terminal Unit (TTU) interface. The adoption of this specification does not imply the adoption of a specific security standard. 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.
- Only numeric PIN pads are handled in this specification.

This specification also supports the Hardware Security Module (HSM), which is necessary for the German ZKA Electronic Purse transactions. Furthermore the HSM stores terminal specific data. This data will be compared against the message data fields (Sent and Received ISO8583 messages) prior to HSM-MAC generation/verification. HSM-MACs are generated/verified only if the message fields match the data stored. Keys used for cryptographic HSM functions are stored separate from other keys. This must be considered when importing keys. This version of PIN pad complies to the current ZKA specification 3.0. It supports loading and unloading against card account for both card types (Type 0 and Type 1) of the ZKA electronic purse. It also covers the necessary functionality for 'Loading against other legal tender'. 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:

- HSMTData
- SetTData
- SecureMsgSend
- SecureMsgReceive
- GetJournal
- OPTRequired
- HSMInit
- HSMTDataChanged

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 WFS_PIN_USEKEYENCKEY type of access. Applications can check impact of key deletion using KeyDetail or KeydetailEx.

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.

Definitions and Abbreviations	
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
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.

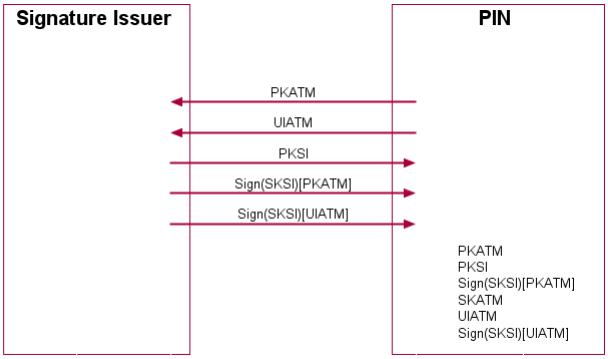
Item Name	Item Type	Signed by	Description
"sigIssuerVendor"	Public Key	N/A	The public key of the signature issuer, i.e. PKSI
"eppCryptKey"	Public/Private key-pair	The private key associated with sigIssuerVendor	The key-pair used to encrypt and encrypt the symmetric. key, i.e SK ATM and PK ATM . The public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

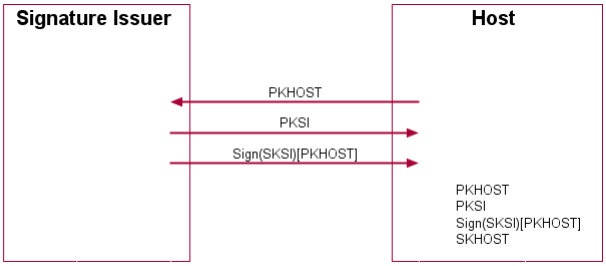
Initialization Phase – Signature Issuer and ATM PIN

This would typically occur in a secure manufacturing environment.



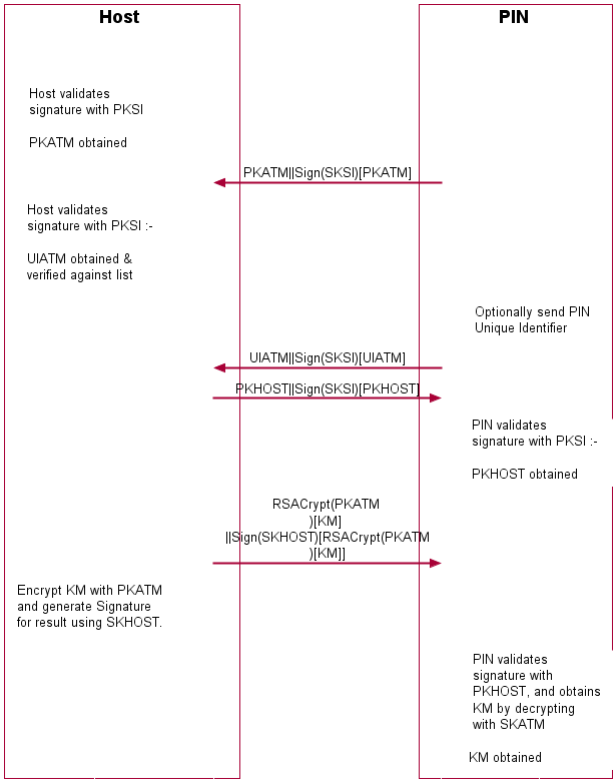
Initialization Phase – Signature Issuer and Host

This would typically occur in a secure offline environment.



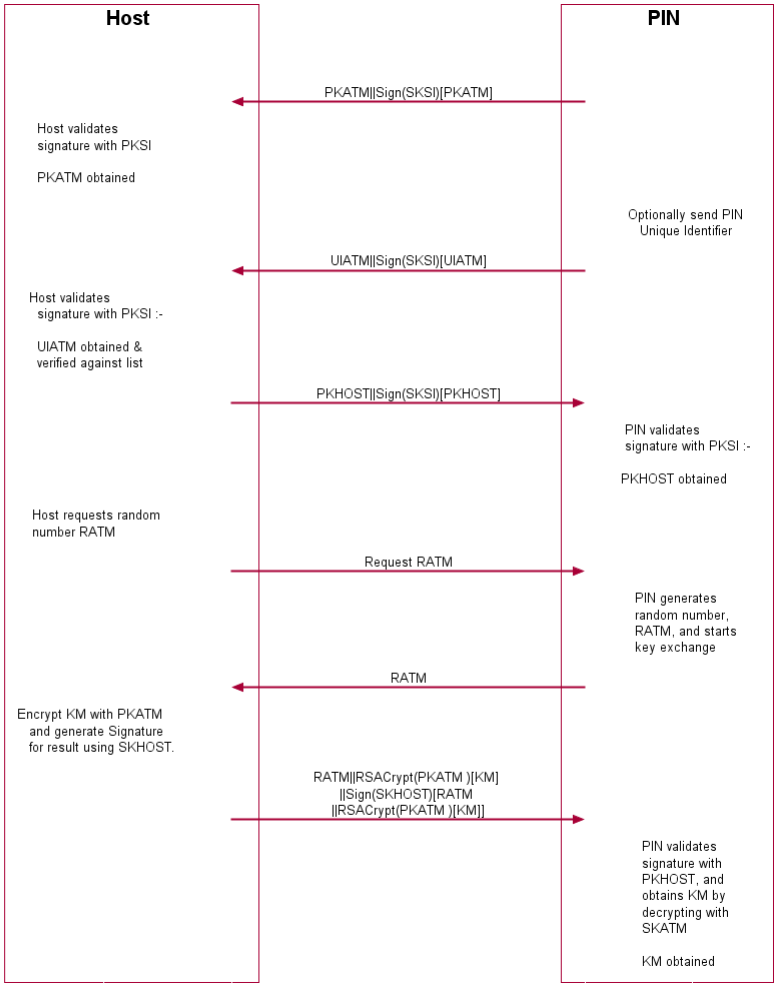
Key Exchange – Host and ATM PIN

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.



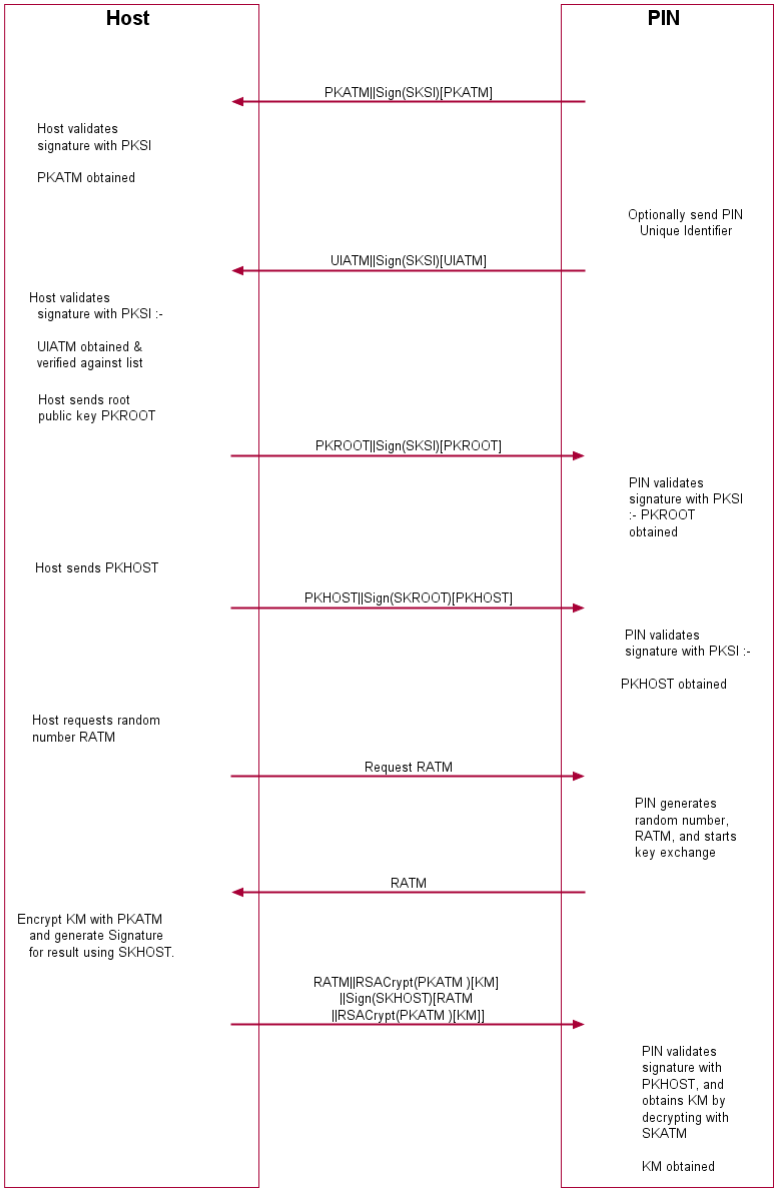
Key Exchange (with random number) – Host and ATM PIN

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

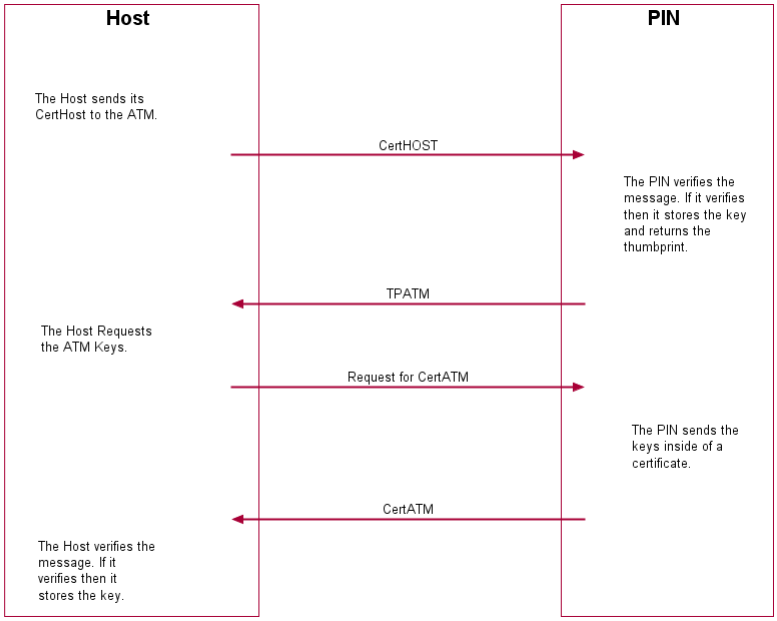
In summary, both end points, the ATM and the Host, inform each other of their Public Keys. This information is then used to securely send the PINS device Master Key to the ATM. A trusted third party, Certificate Authority (or a HOST if it becomes the new CA), is used to generate the certificates for 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).

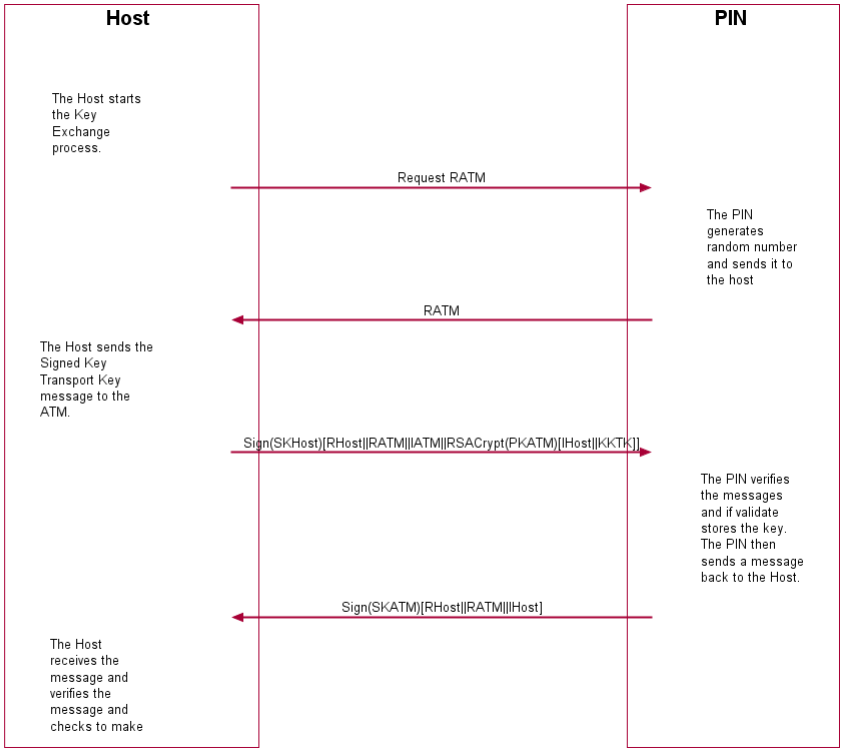
The following diagram shows how the Host and ATM Load and Get each other's information to make Remote Key Loading possible:



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.

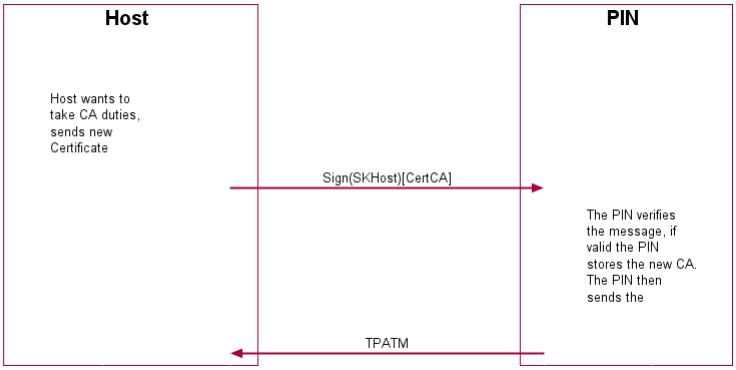
The following diagram below shows how the Host and ATM transmit the Key Transport Key.



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.



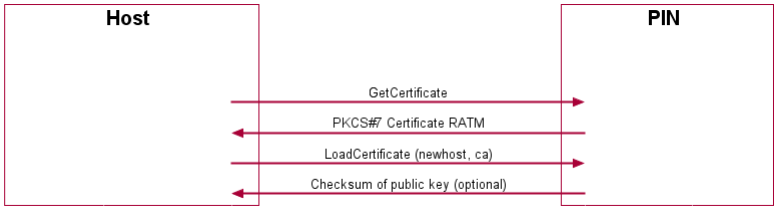
Primary and Secondary Certificate

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.

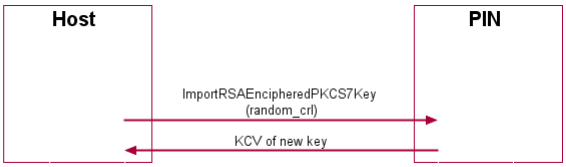


TR34 Key Transport

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.

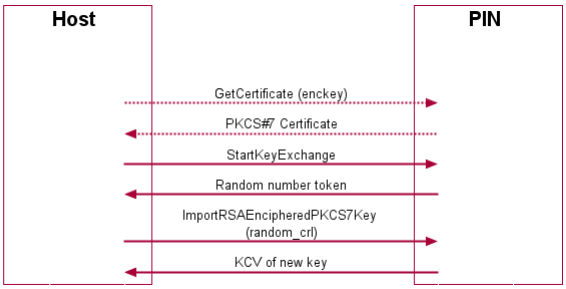
One Pass

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.



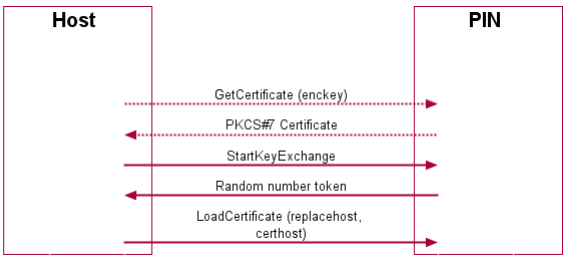
Two Pass

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.

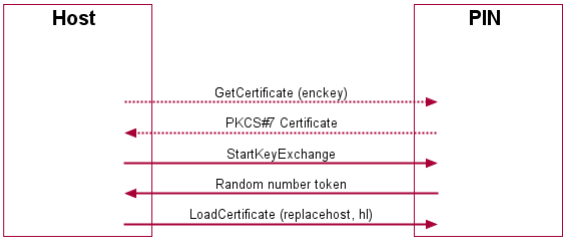


NB: Dotted lines represent commands that are only required if the PIN device encryption certificate has not been previously stored by the host.

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

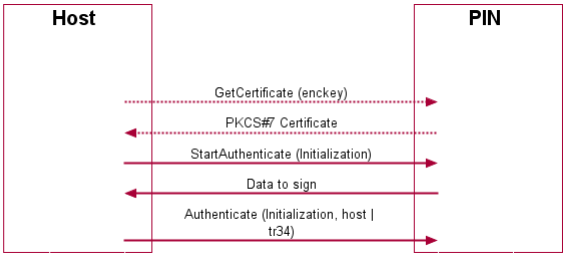
completed such that the PIN device is bound to the host.



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

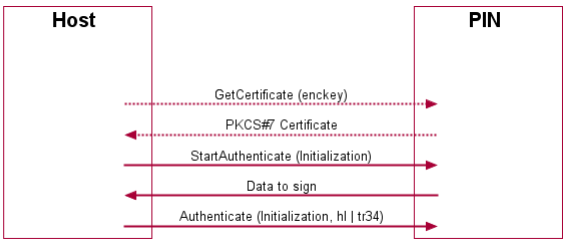
This section defines the command to use when transferring a TR34 UNBIND token as defined in X9 TR34-2012. Pre-condition: A successful BIND command has completed such that the PIN device is bound to the host.



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.

German ZKA GeldKarte (Deutsche Kreditwirtschaft)

The pin service is able to handle the German "Geldkarte", which is an electronic purse specified by the DK (Deutsche Kreditwirtschaft) formerly known as the ZKA (Zentraler Kreditausschuß) protocol. For anyone attempting to write an application that handles this type of chip card, it is essential to read and understand the ZKA specifications see [Ref 17], [Ref 6] and [Ref 7].

How to use the SecureMsg commands

This is to describe how an application should use the SecureMsgSend and SecureMsgReceive commands for transactions involving chipcards with a German ZKA GeldKarte chip.

```
•   Applications must call SecureMsgSend for every command they send to the chip or to a host system, including those commands that do
not actually require secure messaging. This enables the Service Provider to remember security-relevant data that may be needed or check
ed later in the transaction.

•   Applications must pass a complete message as input to SecureMsgSend, with all fields - including those that will be filled by the S
ervice Provider - being present in the correct length. All fields that are not filled by the Service Provider must be filled with the u
ltimate values in order to enable MACing by the Service Provider.

•   Every command SecureMsgSend that an application issues must be followed by exactly one command SecureMsgReceive that informs the Se
```

rvice Provider about the response from the chip or host. If no response is received (timeout or communication failure) the application must issue a SecureMsgReceive command with Msg = NULL to inform the Service Provider about this fact.

- If a system is restarted after a SecureMsgSend was issued to the Service Provider but before the SecureMsgReceive was issued, the restart has the same effect as a SecureMsgReceive command with msg = NULL.

- Between a SecureMsgSend and the corresponding SecureMsgReceive no SecureMsgSend with the same Protocol must be issued. Other pin... commands – including SecureMsgSend / receive with different Protocol – may be used.

Protocol isoAs

This protocol handles ISO8583 messages between an ATM and an authorization system (AS).

Only messages in the new ISO format, with new PAC/MAC-format using session keys and Triple-DES are supported.

Authorization messages may be used to dispense the amount authorized in cash or to load the amount into an electronic purse (GeldKarte).

For loading a GeldKarte the only type of authorization supported is a transaction originating from track 3 of a German ec-card (message types 0200/0210 for authorization and 0400/0410 for reversal).

For dispensing cash, transactions originating from international cards (message types 0100/0110 and 0400/0410) are supported as well.

The following bitmap positions are filled by the Service Provider:

- BMP11 – Trace-Nummer
- BMP52 – PAC
- BMP57 – Verschlüsselungsparameter (only the challenge values RNDMES and RNDPAC)
- BMP64 – MAC

These bitmaps have to be present and the corresponding flag has to be set in the primary bitmap when the ISO message is passed to the HSM.

The following bitmap positions are checked by the Service Provider and have to be filled by the application:

- Nachrichtentyp
- BMP3 – Abwicklungskennzeichen (only for GeldKarte, not for cash)
- BMP4 – Transaktionsbetrag (only for GeldKarte, not for cash)
- BMP41 – Terminal-ID
- BMP42 – Betreiber-BLZ

For additional documentation of authorization messages see [Ref. 27] – [Ref. 30].

Protocol isoLz

This protocol handles ISO8583 messages between a „Ladeterminal" and a „Ladezentrale" (LZ).

Only messages in the new ISO format, with new MAC-format using session keys and Triple-DES are supported.

Both types of GeldKarte chip (type 0 = DEM, type 1 = EUR) are supported.

The following bitmap positions are filled by the Service Provider:

- BMP11: Trace-Nummer
- BMP57: Verschlüsselungsparameter (only the challenge value RNDMES)
- BMP64: MAC

These bitmaps have to be present and the corresponding flag has to be set in the primary bitmap when the ISO message is passed to the HSM.

The following bitmap positions are checked by the Service Provider and have to be filled by the application:

- Nachrichtentyp
- BMP3: Abwicklungskennzeichen
- BMP4: Transaktionsbetrag
- BMP12: Uhrzeit
- BMP13: Datum
- BMP25: Konditionscode
- BMP41: Terminal-ID
- BMP42: Betreiber-BLZ (caution: "Ladeentgelt" also in BMP42 is not set by the EPP)
- BMP61: Online-Zeitpunkt
- BMP62: Chipdaten

The following bitmap positions are only checked if they are available:

- BMP43: Standort
- BMP60: Kontodaten Ladeterminal

For a documentation of the Ladezentrale interface see [Ref. 31].

Protocol isoPs

This protocol handles ISO8583 messages between a terminal and a "Personalisierungsstelle" (PS). These messages are about OPT.

The Service Provider creates the whole message with SecureMsgSend, including message type and bitmap.

For a documentation of the Personalisierungsstelle interface see [Ref. 7].

Protocol chipZka

This protocol is intended to handle messages between the application and a GeldKarte.

Both types of GeldKarte are supported.

Both types of load transactions ("Laden vom Kartenkonto" and "Laden gegen andere Zahlungsmittel") are supported.

See the chapter "Command Sequence" below for the actions that Service Providers take for the various chip card commands.

Only the command APDUs to and the response APDUs from the chip must be passed to the Service Provider, the ATR (answer to reset) data from the chip is not passed to the Service Provider.

For a documentation of the chip commands used to load a GeldKarte see [Ref. 31].

Protocol rawData

This protocol is intended for vendor-specific purposes. Generally the use of this protocol is not recommended and should be restricted to issues that are impossible to handle otherwise.

For example a HSM that requires vendor-specific, cryptographically secured data formats for importing keys or terminal data may use this protocol.

Application programmers should be aware that the use of this command may prevent their applications from running on different hardware.

Protocol pbm

This protocol handles host messages between a terminal and a host system, as specified by PBM protocol.

For documentation of this protocol see [Ref. 8] – [Ref. 13].

Some additions are defined to the PBM protocol in order to satisfy the German ZKA 3.0 PAC/MAC standard. See [Ref. 14].

The commands SecureMsgSend and SecureMsgReceive handle the PAC and MAC in the VARDATA 'K' or 'Q' subfield of transactions records and responses. The MAC in the traditional MACODE field is not affected.

In order to enable the Service Provider to understand the messages, the application must provide the messages according to the following rules:

```
•    All alphanumeric fields must be coded in EBCDIC.

•   Pre-Edit (padding and blank compression) must not be done by the application. The Service Provider will check the macMode field and
will perform the pre-edit according to what the macMode field intends.

•   In order to enable the Service Provider to find the vardata subfield 'K' or 'Q', it must be included in the message by the applicat
ion, with the indicator 'K' or 'Q' and its length set.

•   Because CARDDATA (track 2) and T3DATA (track 3) fields always take part in the MAC computation for a transaction record, these fiel
ds must be included in the message, even if they already have been sent to the host in a previous transaction record and the CI-Option
shortRec prevents them from being sent again.
```

Protocol hsmLdi

With this protocol an application can request information about the personalized OPT groups.

The information returned consists of personalization record like in BMP62 of an OPT response but without MAC.

Data format:

```
        XX XX VV - group ID and version number (BCD format)

        XX - number of LDIs within the group (BCD format)
        …
        first LDI of the group
        …
        last LDI of the group

        XX XX VV - group ID and version number (BCD format)
        ...
        etc. for several groups
```

Each LDI consists of:

```
       NN           Number of the LDI

       00           Alg. Code

       LL           Length of the following data

       XX...XX      data of the LDI
```

For each group ID the Service Provider must always return the standard LDI. LDI 01 must also be returned for groups AF XX VV. Further LDIs can be returned optionally.

Protocol genas

This protocol provides the capability to create a PAC (encrypted PIN block) and to create and verify a MAC for a proprietary message. As the Service Provider does not know the message format, it cannot complete the message by adding security relevant fields like random values, PAC and MAC, like it does for the protocol isoAs. Only the application is able to place these fields into the proper locations. Using this protocol, an application can generate the PAC and the random values in separate steps, adds them to the proprietary send-message, and finally lets the Service Provider generate the MAC. The generated MAC can then be added to the send-message as well.

For a received message, the application extracts the MAC and the associated random value and passes them along with the entire message data to the Service Provider for MAC verification.

PAC generation supports PIN block ISO-Format 0 and 1 for 3DES and ISO-Format 4 for AES.

Command description:

The first byte of field msg of secMsg contains a subcommand, which is used to qualify the type of operation. The remaining bytes of the command data are depending on the value of the subcommand.

The following sub-commands are defined:

- generatePAC 3DES (Code 0x01)
  Returns the encrypted PIN block together with generation and version values of the Master Key and the PAC random value.

- getMacRandom 3DES (Code 0x02)
  Returns the generation and version values of the Master Key and the MAC random value.
- GenerateMAC 3DES (Code 0x03)
  Returns the generated MAC for the message data passed in. Note that the MAC is generated for exactly the data that is presented (contents and sequence). Data that should not go into MAC calculation must not be passed in.
- verifyMac 3DES (Code 0x04)
  Generates a MAC for the data passed in and compares it with the provided MAC value. MAC random value, key generation and key version must be passed in separately.
- Generate PAC AES (Code 0x05)
  Returns the encrypted PIN block wrapped in the BMP110.2 (Dataset 01).
- Get MAC Random AES (Code 0x06)
  Returns the MAC random value wrapped in the BMP110.3 (Dataset 02).
- Generate MAC AES (Code 0x07)
  Returns the generated MAC for the message data passed in. Note that the MAC is generated for exactly the data that is presented (contents and sequence). Data that should not go into MAC calculation must not be passed in.
  Used algorithm is CMAC.
- Verify MAC AES (Code 0x08)
  Generates a MAC for the data passed in and compares it with the provided MAC value. The MAC data must be passed in as BMP110.3 (Dataset 02) in the format:
  08 (sub-command) + BMP110.3 + MAC + message to be verified.

Command/Message sequence:

Command	msg in SecureMsgCommand	msg in SecureMsgCompletion	Service Provider´s actions
SecureMsgSend	Byte 0: 0x01 (Generate PAC) Byte 1: format (0 or 1) Byte 2-9: ANF (Primary Account Number, if length is less than 12 digits, value must be left padded with binary 0, only applicable for format 0)	Byte 0: key generation Byte 1: key version Byte 2-17: PAC random Byte 18-25: PAC value (all values are binary values)	Generates a session key for PAC generation and finally the PAC itself. Determine generation and version values of Master- Key and return them along with the random value
SecureMsgSend	Byte 0: 0x02 (Get MAC Random)	Byte 0: key generation Byte 1: key version Byte 2-17: MAC random (all values are binary values	Generates a session key for MAC generation (see next step below) Determine generation and version values of Master- Key and return them along with the random value
SecureMsgSend	Byte 0: 0x03 (Generate MAC) Byte 1-n: Message to be mac'ed (all values are binary values)	Byte 0-7: generated MAC (binary value)	Generates MAC over bytes 1-n of the inbound message using the session key created in the previous step.
SecureMsgReceive	Byte 0: 0x04 (Verify MAC) Byte 1: key generation Byte 2: key version Byte 3-18: MAC random Byte 19-26: MAC Byte 27-n: Message to be verified (all values are binary values) NOTE: If no message has been received, this function must be called<by> by omitting Bytes 1-n	N/A	Generates a session key using the Master key identified by key generation and version by using the random value passed in. Generates a MAC for the message data passed in and compare the resulting MAC with the MAC passed in.
SecureMsgSend	Byte 0: 0x05 (Generate PAC AES) Byte 1: format (4)	Byte 0: 01 Identification for Dataset 01 Byte 1-2: length of data Byte 3-n: data	Generates a session key for PAC generation and finally the PAC itself. Returned values are in the format of dataset 01 of BMP110
SecureMsgSend	Byte 0: 0x06 (Get MAC Random AES)	Byte 0: 02 Identification for Dataset 02 Byte 1-2: length of data Byte 3-n: data	Generates a session key for MAC generation (see next step below) Returned values are in the format of dataset 02 of BMP110
SecureMsgSend	Byte 0: 0x07 (Generate MAC AES) Byte 1-n: Message to be mac'ed (all values are binary values)	Byte 0-7: generated MAC (binary value)	Generates MAC over bytes 1-n of the inbound message using the session key created in the previous step.
SecureMsgReceive	Byte 0: 0x08 (Verify MAC AES) Byte 1-37: BMP110 Dataset 02 Byte 38-45: MAC Byte 46-n: Message to be verified (all values are binary values)	N/a	Generates a session key using the Master key identified by key generation and version by using the random value passed in. Generates a MAC for the message data passed in and compare the resulting MAC with the MAC passed in.

Returns:

The error code formatInvalid is returned when:

- The subcommand in Byte 0 of msg for Execute Command SecureMsgSend with protocol genas is not 01, 02, 03, 05, 06 or 07.
- The subcommand in Byte 0 of msg for Execute Command SecureMsgReceive with protocol genas is not 04 or 08.
- The subcommand in Byte 0 of msg for Execute Command SecureMsgSend with protocol genas is 01 and Byte 1 is not 00 and not 01 (PIN block format is not ISO-0 and ISO-1).
- The subcommand in Byte 0 of msg for Execute Command SecureMsgSend with protocol genas is 05 and Byte 1 is not 04 (PIN block format is not ISO-4)
- The individual command data length for a subcommand is less than specified.

The error code hsmStateInvalid is returned when:

- The subcommand in Byte 0 of msg for Execute Command SecureMsgSend with protocol genas is 03 (Generate MAC) without a preceding getMacRandom (secureMsgSend with subcommand 02).
- The subcommand in Byte 0 of msg for Execute Command SecureMsgSend with protocol genas is 07 (Generate MAC) without a preceding getMacRandom (secureMsgSend with subcommand 06).

The error code macInvalid is returned when:

- The subcommand in Byte 0 of msg for Execute Command SecureMsgReceive with protocol genas is 04 (Verify MAC) and the MACs did not match.

The error code keyNotFound is returned when:

- The subcommand in Byte 0 of msg for Execute Command SecureMsgSend with protocol genas is 01 or 05 (Generate PAC) and the Service Provider does not find a master key.
- The subcommand in Byte 0 of msg for Execute Command SecureMsgSend with protocol genas is 02 or 06 (Get MAC Random) and the Service Provider does not find a master key.
- The subcommand in Byte 0 of msg for Execute Command SecureMsgReceive with protocol genas is 04 or 08 (Verify MAC) and the Service Provider does not find a key for the provided key generation and key version values.

The error code noPin is returned when:

- The subcommand in Byte 0 of msg for Execute Command SecureMsgSend with protocol genas is 01 or 05 (Generate PAC) and no PIN or insufficient PIN-digits have been entered.

Protocol chipinchg

This protocol is intended to handle messages exchanged between the PIN pad and a GeldKarte, which are all related to the PIN change transaction.

Only Type-1-GeldKarte is supported, because the former Type-0-GeldKarte will no longer be used as it was a dedicated Deutsche Mark electronic purse only. The Type-1-GeldKarte is used for Euro currency.

The transaction types supported are:

• PIN-Activation („PIN-Aktivierung“)

• PIN-Activation after Failure („PIN-Aktivierung nach Fehlerfall“)

• PIN-Change ("PIN-Änderung")

See the command sequence section below for the actions that Service Providers take for the various chip card commands.

Only the command APDUs to and the response APDUs from the chip must be passed to the Service Provider, the ATR (answer to reset) data from the chip is not passed to the Service Provider.

For the complete documentation of the chip commands used for PIN-Change see [Ref. 34].

Protocol pinCmp

This simple protocol is used to perform a comparison of two PINs entered into the PIN Pad. In order to be able to compare the PINs, the first value must be temporary stored while the second value is entered. The user will be prompted to enter the PIN twice. After the PIN has been entered for the first time, the PIN pad needs to store the PIN value into a temporary location. After the user has entered the PIN for the second time, the PIN pad has to compare both values.

This protocol consists of two subcommands. The first subcommand requests the PIN pad to save the PIN value entered by the getPin command for subsequent comparison. The second subcommand forces the PIN pad to compare the PIN stored with the second value entered by the getPin command. The status of the PIN comparison is returned in the output data.

See the command sequence section below for the actions that Service Providers take for this protocol.

Use of pinCmp with non-GeldKarte ZKA PIN Management

For use with the non-GeldKarte ZKA PIN compare function (see [Ref 37]) there are two more subcommands “start PIN compare” and “end PIN compare”. These have to be called before entry of the first PIN and after querying of the PAC to signal the end of the PIN comparison, respectively.

This is the command sequence for the non-GeldKarte transaction:

Flow	Command pin	protocol	msg in SecureMsgCommand	msg in SecureMsgCompletion	Service Provider’s actions
PIN Compare					
Start PIN comparision	SecureMsgSend	pinCmp	Byte 0: 0x00 (Start PIN compare)		Prepare EPP for PIN comparison. Output data buffer length is zero.
Let the user enter the new PIN for the first time.	GetPin	n/a	n/a	n/a	PIN entry.
	SecureMsgSend	pinCmp	Byte 0: 0x01 (Save PIN)		Save the PIN value entered for subsequent compare. Output data buffer length is zero.
Let the user enter the new PIN for the second time	GetPin	n/a	n/a	n/a	PIN entry.
	SecureMsgSend	pinCmp	Byte 0: 0x02 (Compare PINs)	Byte 0: 0x00 when PIN does not match, and 0x01 when PIN does match.	Compare PIN values.
Get the PAC of the new PIN via genas or isoAs (as usual).					
End PIN comparison.	SecureMsgSend	pinCmp	Byte 0: 0xFF (End PIN compare)		All PIN buffers are cleared. Output data buffer length is zero.

Please note that no other PIN commands apart from GetPin and SecureMsgSend as specified above are allowed inside a start / end PIN compare flow, with the exception of creating the PAC for the old PIN. While the old PIN always has to be entered (using GetPin) before the “Start PIN Compare”, the PAC for the old PIN may be created (using SecureMsgSend with protocol=genas) after the “Start PIN Compare” if (enforced by the host protocol) the same session key SKPAC has to be used for encrypting both the old and the new PIN.

Protocol isopinchg

This protocol handles ISO8583 messages between an ATM and an authorization system (AS) related to the transactions:

- PIN-Activation („PIN-Aktivierung“)
- PIN-Activation after Failure („PIN-Aktivierung nach Fehlerfall“)
- PIN-Change ("PIN-Änderung")

The message types supported are:

- 0640 (PIN Change / PIN Activation Request)
- 0642 (Confirmation / Reversal Request for PIN Change / PIN Activation)
- 0643 (Confirmation Repeat Request for PIN Change / PIN Activation)
- 0650 (PIN Change / PIN Activation Response)
- 0652 (Confirmation / Reversal Response)

The following bitmap positions are filled by the Service Provider:

- BMP52 PAC
- BMP57 Verschlüsselungsparameter (KTerminal Generation, KTerminal Version, RNDMES and RNDPAC)
- BMP62 (EFID, EFINFO, Record number of PIN, Key Version of KCard, EFFBZ, PAC, Random value returned by getChallenge)
- BMP64 MAC

These bitmaps have to be present and the corresponding flag has to be set in the primary bitmap when the ISO message is passed to the HSM.

See the command sequence section below for the actions that Service Providers take for the various messages.

For the complete documentation of the messages used for PIN-Change see [Ref. 34].

Command Sequence

The following list shows the sequence of actions an application has to take for the various GeldKarte Transactions. Please note that this is a summary and is just intended to clarify the purpose of the chipcard-related ... commands. In no way it can replace the ZKA specifications mentioned above.

Command	protocol	msg	Service Provider's actions
Preparation for Load/Unload			
SecureMsgSend	chipZka	Command APDU SELECT FILE DFBÖRSE	
SecureMsgReceive	chipZka	Response APDU	recognize type of chip
SecureMsgSend	chipZka	Command APDU READ RECORD EFID	
SecureMsgReceive	chipZka	record EFID	store EFID
SecureMsgSend	chipZka	Command APDU READ RECORD EFLLOG	
SecureMsgReceive	chipZka	record EFLLOG	
SecureMsgSend	chipZka	Command APDU READRECORD EFBÖRSE	
SecureMsgReceive	chipZka	record EFBÖRSE	
SecureMsgSend	chipZka	Command APDU READRECORD EFBETRAG	
SecureMsgReceive	chipZka	record EFBETRAG	
Load against other ec-Card			
SecureMsgSend	chipZka	**for type 0 chips only** Command APDU READ RECORD EFKEYD	
SecureMsgReceive	chipZka	record EFKEYD	
SecureMsgSend	chipZka	**for type 1 chips only** Command APDU GET KEYINFO	
SecureMsgReceive	chipZka	Response APDU	
SecureMsgSend	chipZka	Command APDU GET CHALLENGE	
SecureMsgReceive	chipZka	Random number RND1 from Chip	store RND1
SecureMsgSend	chipZka	Command APDU LADEN EINLEITEN with Secure msg.	fill: -Terminal ID -Traceno. -RND2 -MAC
SecureMsgReceive	chipZka	Response APDU	store response APDU for later check of isoLz message, BMP 62
SecureMsgSend	isoAz	ISO8583 message 0200 Authorization Request	Fill: - Traceno. (BMP 11) - PAC (BMP 52) - RNDMES + RNDPAC (BMP 57) - MAC (BMP 64) check other security relevant fields
SecureMsgReceive	isoAz	ISO8583 message 0210 Authorization Response	check MAC and other security relevant fields
SecureMsgSend	isoLz	ISO8583 message 0200 Ladeanfrage	Fill: - Traceno. (BMP 11) - RNDMES (BMP 57) - MAC (BMP 64) check other security relevant fields.
SecureMsgReceive	isoLz	ISO8583 message 0210 Ladeantwort	check MAC and other security relevant fields, store BMP62 for later use in LADEN command.
SecureMsgSend	chipZka	Command APDU GET CHALLENGE	
SecureMsgReceive	chipZka	Random number RND3 from chip	store RND3
SecureMsgSend	chipZka	Command APDU LADEN with Secure msg.	provide complete command from BMP62 of isoLz response , compute command MAC
SecureMsgReceive	chipZka	Response APDU	check response MAC
GetJournal	isoLz	Vendor specific	
GetJournal	IsoAz	Vendor specific	
Reversal of a Load against other ec-Card			
SecureMsgSend	chipZka	Command APDU SELECT FILE DFBÖRSE	
SecureMsgReceive	chipZka	Response APDU	
SecureMsgSend	chipZka	Command APDU GET CHALLENGE	
SecureMsgReceive	chipZka	Random number RND5 from chip	store RND5
SecureMsgSend	chipZka	Command APDU LADEN EINLEITEN with Secure msg.	Fill: -Terminal ID -Traceno. -RND6 -Keyno. KGKLT -MAC

Command	protocol	msg	Service Provider's actions
SecureMsgReceive	chipZka	Response APDU	store response APDU for later check of isoLz message, BMP 62
SecureMsgSend	IsoAz	ISO8583 message 0400 Storno	Fill: - Traceno. (BMP 11) - PAC (BMP 52) - RNDMES + RNDPAC (BMP 57) - MAC (BMP 64) check other security relevant fields
SecureMsgReceive	IsoAz	ISO8583 message 0410 Storno Response	check MAC and other security relevant fields.
SecureMsgSend	isoLz	ISO8583 message 0400 Storno	Fill: - Traceno. (BMP 11) - RNDMES (BMP 57) - MAC (BMP 64) check other security relevant fields.
SecureMsgReceive	isoLz	ISO8583 message 0410 Storno Response	check MAC and other security relevant fields, store BMP62 for later use in LADEN command.
SecureMsgSend	chipZka	Command APDU GET CHALLENGE	
SecureMsgReceive	chipZka	Random number RND7 from chip	store RND7
SecureMsgSend	chipZka	Command APDU LADEN with Secure msg.	provide complete command from BMP62 of isoLz response , compute command MAC
SecureMsgReceive	chipZka	Response APDU	check response MAC
GetJournal	isoLz	Vendor specific	
GetJournal	IsoAz	Vendor specific	
PIN Verification Type 0			
SecureMsgSend	chipZka	Command APDU GET CHALLENGE	
SecureMsgReceive	chipZka	Random number RND0 from chip	store RND0
SecureMsgSend	chipZka	Command APDU EXTERNAL AUTHENTICATE	fill -Keyno. KINFO -ENCRND
SecureMsgReceive	chipZka	Response APDU	
SecureMsgSend	chipZka	Command APDU PUT DATA	fill RND1
SecureMsgReceive	chipZka	Response APDU	
SecureMsgSend	chipZka	Command APDU READ RECORD EFINFO with Secure Messaging	
SecureMsgReceive	chipZka	record EFINFO	check MAC
SecureMsgSend	chipZka	Command APDU GET CHALLENGE	
SecureMsgReceive	chipZka	Random number RND2 from chip	store RND2
SecureMsgSend	chipZka	Command APDU VERIFY	provide complete command APDU
SecureMsgReceive	chipZka	Response APDU	
PIN Verification Type !			
SecureMsgSend	chipZka	Command APDU GET KEYINFO	
SecureMsgReceive	chipZka	Response APDU	
SecureMsgSend	chipZka	Command APDU GET CHALLENGE	
SecureMsgReceive	chipZka	Random number RND0 from chip	store RND0
SecureMsgSend	chipZka	Command APDU MUTUAL AUTHENTICATE	fill ENC0
SecureMsgReceive	chipZka	Response APDU	check ENC1
SecureMsgSend	chipZka	Command APDU VERIFY	provide complete command APDU
SecureMsgReceive	chipZka	Response APDU	check MAC
„Laden vom Kartenkonto" (both types)			
SecureMsgSend	chipZka	Command APDU LADEN EINLEITEN	fill -Terminal ID -Trace No.
SecureMsgReceive	chipZka	Response APDU	
SecureMsgSend	isoLz	ISO8583 message 0200 Ladeanfrage	fill - Traceno. (BMP 11) - RNDMES (BMP 57) - MAC (BMP 64) check other security relevant fields.
SecureMsgReceive	isoLz	ISO8583 message 0210 Ladeantwort	check MAC and other security relevant fields.
SecureMsgSend	chipZka	Command APDU LADEN	
SecureMsgReceive	chipZka	Response APDU	
GetJournal	isoLz	Vendor specific	
Reversal of a „Laden vom Kartenkonto"			
SecureMsgSend	chipZka	Command APDU SELECT FILE DFBÖRSE	
SecureMsgReceive	chipZka	Response APDU	
SecureMsgSend	chipZka	Command APDU LADEN EINLEITEN	fill -Terminal ID -Traceno.
SecureMsgReceive	chipZka	Response APDU	
SecureMsgSend	isoLz	ISO8583 message 0400 Storno	fill - Traceno. (BMP 11) - RNDMES (BMP 57) - MAC (BMP 64) check other security relevant fields.
SecureMsgReceive	isoLz	ISO8583 message 0410 Storno Response	check MAC and other security relevant fields
SecureMsgSend	chipZka	Command APDU LADEN	
SecureMsgReceive	chipZka	Response APDU	
GetJournal	isoLz	Vendor specific	
unload			
SecureMsgSend	chipZka	ENTLADEN EINLEITEN	fill -Terminal ID -Trace No.
SecureMsgReceive	chipZka	Response APDU	
SecureMsgSend	isoLz	ISO8583 message Entladeanfrage 0200	fill - Traceno. (BMP 11) - RNDMES (BMP 57) - MAC (BMP 64) check other security relevant fields.
SecureMsgReceive	isoLz	ISO8583 message Entladeantwort 0210	check MAC and other security relevant fields
SecureMsgSend	chipZka	ENTLADEN	
SecureMsgReceive	chipZka	Response APDU	
SecureMsgSend	chipZka	ENTLADEN EINLEITEN	fill -Terminal ID -Trace No.
SecureMsgReceive	chipZka	Response APDU	
SecureMsgSend	isoLz	ISO8583 message Entladequittung 0202	fill - Traceno. (BMP 11) - RNDMES (BMP 57) - MAC (BMP 64) check other security relevant fields.
SecureMsgReceive	isoLz	ISO8583 message Entladebestätigung 0212	check MAC and other security relevant fields
SecureMsgSend	chipZka	Command APDU ENTLADEN	
secureMsgReceive	chipZka	Response APDU	

Command	protocol	msg	Service Provider's actions
GetJournal	isoLz	Vendor specific	
Repeated Messages (Stornowiederholung / Entladequittungswiederholung)			
SecureMsgSend	isoLz	ISO8583 message Stornowiederholung 0401 or Entladequittungswiederholung 0203	fill - Traceno. (BMP 11) - RNDMES (BMP 57) - MAC (BMP 64) check other security relevant fields.
SecureMsgReceive	isoLz	ISO8583 message Stornoantwort 410 or Entladebestätigung 0212	check MAC and other security relevant fields
GetJournal	isoLz	Vendor specific	

Command	protocol	msg	Service Provider's actions
Preparation for PIN Change			
SecureMsgSend	chippinchg	Command APDU READ RECORD EFID	
SecureMsgReceive	chippinchg	Response APDU Record EFID Store EFID Will be inserted into BMP62 of a PIN Change request	
SecureMsgSend	chippinchg	Command APDU GET CHALLENGE	
SecureMsgReceive	chippinchg	Random number RND0 from Chip	Store RND0
SecureMsgSend	chippinchg	Command APDU READ RECORD EFINFO	Fill RND1
SecureMsgReceive	chippinchg	Response APDU Record EFINFO	Check MAC, Store EFINO Will be inserted into BMP62 of a PIN Change request
SecureMsgSend	chippinchg	Command APDU GET KEYINFO	
SecureMsgReceive	chippinchg	Response APDU Version of KCard	Store version byte Will be inserted into BMP62 of a PIN Change request
SecureMsgSend	chippinchg	Command APDU SEARCH RECORD '01' of EFPWDD	
SecureMsgReceive	chippinchg	Response APDU	Store record number Will be inserted into BMP62 of a PIN Change request
SecureMsgSend	chippinchg	Command APDU READ RECORD EFFBZ	
SecureMsgReceive	chippinchg	Response APDU Initial value FBZ Actual value FBZ	
PIN Verification			
SecureMsgSend	chippinchg	Command APDU GET KEYINFO	
SecureMsgReceive	chippinchg	Response APDU	
SecureMsgSend	chippinchg	Command APDU GET CHALLENGE	
SecureMsgReceive	chippinchg	Random number RND0 from chip	Store RND0
SecureMsgSend	chippinchg	Command APDU MUTUAL AUTHENTICATE	Fill ENC0
SecureMsgReceive	chippinchg	Response APDU	Check ENC1
SecureMsgSend	chippinchg	Command APDU VERIFY	Provide complete command APDU
SecureMsgReceive	chippinchg	Response APDU	Check MAC Create PAC for old PIN
PIN Change			
Let the user enter the PIN for the first time, by invoking the command GetPin			
SecureMsgSend	HSMPinCmp	Byte 0: 0x01 (Save PIN)	Save the PIN value entered for subsequent compare. Output data buffer length is zero.
Let the user enter the PIN for the second time, by invoking the command GetPin			
SecureMsgSend	HSMPinCmp	Byte 0: 0x02 (Compare PINs)	Compare PIN values. Returns Byte 0: as 0x00 when PIN does not match, and 0x01 when PIN does match. Create PAC for new PIN if values match
SecureMsgSend	chippinchg	Command APDU MANAGE SECURITY ENVIRONMENT	
SecureMsgReceive	chippinchg	Response APDU	
SecureMsgSend	chippinchg	Command APDU GET CHALLENGE	
SecureMsgReceive	chippinchg	Random number RND0 from Chip	Store RND0 Will be inserted into BMP62 of a PIN Change request
SecureMsgSend	isopinchg	ISO8583 Message 0640	Fill - PAC old PIN (BMP52) - KTerminal generation + KTerminal version + RNDMES + RNDPAC (BMP57) - Chip Data (BMP62) with PAC of new PIN - MAC (BMP64)
SecureMsgReceive	isopinchg	ISO8583 message 0650	Check MAC
SecureMsgSend	chippinchg	Command APDU from BMP62	
SecureMsgReceive	chippinchg	Response APDU	
PIN Change Confirmation/ Repeated Confirmation			
SecureMsgSend	isopinchg	ISO8583 message 0642 or 0643 BMP25 = 00	Fill - KTerminal generation + KTerminal version + RNDMES (BMP57) - Chip Data (BMP62) with PAC of new PIN - MAC (BMP64)
SecureMsgReceive	isopinchg	ISO8583 message 0652	Check MAC
PIN Change Reversal/ Repeated Reversal			
SecureMsgSend	isopinchg	ISO8583 message 0642 or 0643 BMP25 ≠ 00	Fill - KTerminal generation + KTerminal version + RNDMES (BMP57) - Chip Data (BMP62) with PAC of old PIN - MAC (BMP64)
SecureMsgReceive	isopinchg	ISO8583 message 0652	Check MAC
PIN Activation after failure			
SecureMsgSend	isopinchg	ISO8583 message 0640	Fill - PAC entered PIN (BMP52) - KTerminal generation + KTerminal version + RNDMES + RNDPAC (BMP57) - Chip Data (BMP62) with PAC of entered PIN - MAC (BMP64)
SecureMsgReceive	isopinchg	ISO8583 message 0650	Check MAC
PIN Activation			

Command	protocol	msg	Service Provider's actions
SecureMsgSend	chippinchg	Command APDU MANAGE SECURITY ENVIRONMENT	
SecureMsgReceive	chippinchg	Response APDU	
SecureMsgSend	chippinchg	Command APDU GET CHALLENGE	
SecureMsgReceive	chippinchg	Random number RND0 from Chip	Store RND0 Will be inserted into BMP62 of a PIN Activation request
SecureMsgSend	isopinchg	ISO8583 Message 0640	Fill - PAC entered PIN (BMP52) - KTerminal generation + KTerminal version + RNDMES + RNDPAC (BMP57) - Chip Data (BMP62) with PAC of entered PIN - MAC (BMP64)
SecureMsgReceive	isopinchg	ISO8583 message 0650	Check MAC
SecureMsgSend	chippinchg	Command APDU from BMP62	
SecureMsgReceive	chippinchg	Response APDU	
PIN Activation Confirmation/ Repeated Confirmation			
SecureMsgSend	chippinchg	Command APDU MANAGE SECURITY ENVIRONMENT	
SecureMsgReceive	chippinchg	Response APDU	
SecureMsgSend	chippinchg	Command APDU GET CHALLENGE	
SecureMsgReceive	chippinchg	Random number RND0 from Chip	Store RND0 Will be inserted into BMP62 of a PIN Activation confirmation
SecureMsgSend	isopinchg	ISO8583 message 0642 or 0643 BMP25 = 00	Fill - KTerminal generation + KTerminal version + RNDMES (BMP57) - Chip Data (BMP62) with PAC of entered PIN - MAC (BMP64)
SecureMsgReceive	isopinchg	ISO8583 message 0652	Check MAC
SecureMsgSend	chippinchg	Command APDU from BMP62	
SecureMsgReceive	chippinchg	Response APDU	

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.

French Cartes Bancaires

"Groupement des Cartes Bancaires" from France has specified a cryptographic architecture for ATM networks. See the document [Ref. 15] for details.

The command Enclo with the protocol giecb is used for:

- ATM initialization
- Renewal of ATM master key
- Renewal of HOST master key
- Generation and loading of key transport key

Keys loaded or generated with Enclo get names like any other keys in a PIN service. keyDetail shows the key with this name and the name may be used with ImportKey to delete a key.

Data Structure for Enclo

Data will be transferred as tag-length-value (TLV) structure, encoded according to the distinguished encoding rules (DER) defined in [Ref. 16].

The following is a list of top level tags defined for the use with giecb. All these tags have the application class, therefore the Identifier Octets are (binary):

- 0 1 0 n n n n n - for the primitive types
- 0 1 1 n n n n n - for the constructed types

Tag Number	Primitive/Constructed	Identifier Octet	Contents
0	P	0x40	**Protocol Version** The integer value zero for this version of the protocol
1	P	0x41	**Interchange Code** An ascii string holding one of the interchange codes defined in [Ref. 15], e.g. "HRN-H1"
2	C	0x62	**Interchange Data** The data items as defined by [Ref.15], see table below for details
3	P	0x43	**Key Name** An ascii string holding the name for the key being loaded or generated.

The Interchange Data (Tag 2) is constructed from data items where tag numbers of the sub-tags from 1 to 23 correspond to the data item numbers ("No donnée") as defined in section 3.1 of [Ref. 15]. Some of the data items consist of data elements, for these the constructed encoding will be used. For data items with no data elements the primitive encoding will be used.

All Tags have the context class, therefore the Identifier Octets are (binary):

- 1 0 0 n n n n n - for the primitive types
- 1 0 1 n n n n n - for the constructed types

Tag (=Data Item No)	Primitive/Constructed	Identifier Octet	Data Item Label
1	C	0xA1	IdKG
2	C	0xA2	KTK-encrypted
3	C	0xA3	KGp
4	C	0xA4	KDp
5	C	0xA5	SnSCD
6	P	0x86	Rand
7	P	0x87	HOST authentication
8	P	0x88	KDp signature
9	P	0x89	KGp signature
10	P	0x8A	KTK signature
11	P	0x8B	KT-encrypted
12	P	0x8C	Ksc-encrypted
13	P	0x8D	PIN cryptogram
14	P	0x8E	Seal
15	P	0x8F	Thumbprint of KDp
16	P	0x90	Thumbprint of KGp
17	C	0xB1	IdKD
18	C	0xB2	IdKTK
19	C	0xB3	IdKT
20	C	0xB4	IdKSC
21	P	0x95	Manufacturer

Tag (=Data Item No)	Primitive/Constructed	Identifier Octet	Data Item Label
22	C	0xB6	SCD type
23	C	0xB7	Firmware version

Inside the constructed data items, primitive encoding is used for the data elements, all tags having context class with tag numbers corresponding to the data element numbers (“No d’élément de donnée”) as defined in section 3.1 of [Ref. 15].

Example:

The example shows the der encoding of the input for a Enclo command, for the interchange “GIN-H5”. All data except the 128 byte content of data item 7 is shown in hexadecimal (0x omitted for the sake of readability).

40 01 00 (tag / length / value for Protocol Version 0)

41 06 47 49 4E 2D 48 35 (tag / length / value for Interchange Code “GIN-H5”)

62 81 B5 (tag / length for Interchange Data)

A1 14 (tag / length for data item 1)

```
81 01 00        (data element 1)

82 0C 00 00 00 00 00 00 00 00 00 00 00 00  (data element 2)

83 01 00                      (data element 3)
```

A5 10 (tag / length for data item 5)

```
81 03 00 00 00                  (data element 1)
82 09 00 00 00 00 00 00 00 00 00       (data element 2)
```

86 08 00 00 00 00 00 00 00 00 (tag / length / value for data item 6)

87 81 80 <128 bytes> (tag / length / value for data item 7)

43 05 4D 59 4B 45 59 (tag / length / value for Key Name “MYKEY”)

Command Sequence

The following list shows the sequence of actions an application has to take for the various Cartes Bancaires interchanges.

• GIN (ATM initialization)

Action	Interchange Code	Key Name	Input Data Items	Output Data Items
Thumbprint supplied by host via external channel (GIN-H1)				
Enclo	GIN-G2			21,22,23
Host Communication (GIN-G2 / GIN-H3)				
Enclo	GIN-H3	Key Name for KG 3		16
Enclo	GIN-G4			5,6,1
Host Communication (GIN-G4 / GIN-H5)				
Enclo	GIN-H5	Key Name for KD 5,6,1,7		
Enclo	GIN-G6			5,4,8
Host Communication (GIN-G6)				
Enclo	GIN-G7			15
Send thumbprint to host via external channel (GIN-G7)				

• GRN (Renewal of ATM Master Key)

Action	Interchange Code	Key Name	Input Data Items	Output Data Items
Enclo	GRN-G1			5,6,1
Host Communication (GRN-G1 / GRN-H2)				
Enclo	GRN-H2	Key Name for KD 5,6,1,7		
Enclo	GRN-G3			5,4,8,17
Host Communication (GRN-G3)				
Enclo	GRN-C or GRN-R		17	

The Interchange codes “GRN-C” to commit the transaction resp. “GRN-R” to roll back the transactions are an addition to those defined in [Ref. 15].

• HRN (Renewal of HOST Master Key)

Action	Interchange Code	Key Name	Input Data Items	Output Data Items
Host Communication (HRN-H1)				
Enclo	HRN-H1	Key Name for KG 3,9,1		

• DKT (Generation and Loading of KTK)

Action	Interchange Code	Key Name	Input Data Items	Output Data Items
Enclo	DKT-G1			5,6
Host Communication (DKT-G1 / DKT-H2)				
Enclo	DKT-H2	Key Name for KTK 5,6,2,10,1,17		

Secure Key Entry

This section provides additional information to describe how encryption keys are entered securely through the PIN pad keyboard and also provides examples of possible

keyboard layouts.

Keyboard Layout

The following sections describe what is returned within the SecureKeyDetail output parameters to describe the physical keyboard layout. These descriptions are purely examples to help understand the usage of the parameters they do not indicate a specific layout per Key Entry Mode.

In the following section all references to parameters relate to the output fields of the SecureKeyDetail command.

When keyEntryMode represents a regular shaped PIN pad (regUnique or regShift) then hexKeys must contain one entry for each physical key on the PIN pad (i.e. the product of Rows by Columns). On a regular shaped PIN pad the application can choose to ignore the position and size data and just use the rows and columns parameters to define the layout. However, a Service Provider must return the position and size data for each key.

keyEntryMode == regUnique

When keyEntryMode is regUnique then the values in the array report which physical keys are associated with the function keys 0-9, A-F and any other function keys that can be enabled as defined in the funcKeyDetail parameter. Any positions on the PIN pad that are not used must be defined as a fkUnused in the fk and shiftFk field of the hexKeys structure.

```
1      2   3    Clear  (A)
4      5   6    Cancel (B)
7      8   9    Enter  (C)
(D)    0  (E)  (F)
```

In the above example, where all keys are the same size and the hex digits are located as shown the hexKeys will contain the entries in the array as defined in the following table.

Index	xPos	yPos	xSize	ySize	fk	shiftfk
0	0	0	250	250	fk1	fkUnused
1	250	0	250	250	fk2	fkUnused
2	500	0	250	250	fk3	fkUnused
3	750	0	250	250	fkA	fkUnused
4	0	250	250	250	fk4	fkUnused
5	250	250	250	250	fk5	fkUnused
6	500	250	250	250	fk6	fkUnused
7	750	250	250	250	fkB	fkUnused
8	0	500	250	250	fk7	fkUnused
9	250	500	250	250	fk8	fkUnused
10	500	500	250	250	fk9	fkUnused
11	750	500	250	250	fkC	fkUnused
12	0	750	250	250	fkD	fkUnused
13	250	750	250	250	fk0	fkUnused
14	500	750	250	250	fkE	fkUnused
15	750	750	250	250	fkF	fkUnused

keyEntryMode == regShift

When keyEntryMode is regShift then the values in the array report which physical keys are associated with the function keys 0-9, A-F, and the shift key as defined in the funcKeyDetail parameter. Other function keys as defined by the funcKeyDetail parameter that can be enabled must also be reported. Any positions on the PIN pad that are not used must be defined as a fkUnused in the fk and shiftFk field of the hexKeys structure. Digits 0 to 9 are accessed through the numeric keys as usual. Digits A to F are accessed by using the shift key in combination with another function key, e.g. shift-0 (zero) is hex digit A.

```
1 (B)    2 (C)   3 (D)   Clear
4 (E)    5 (F)   6       Cancel
7        8       9       Enter
SHIFT    0 (A)
```

In the above example, where all keys are the same size and the hex digits 'A' to 'F' are accessed through shift '0' to '5', then the hexKeys will contain the entries in the array as defined in the following table.

Index	xPos	yPos	xSize	ySize	fk	shiftfk
0	0	0	250	250	fk1	fkB
1	250	0	250	250	fk2	fkC
2	500	0	250	250	fk3	fkD
3	750	0	250	250	fkClear	fkUnused
4	0	250	250	250	fk4	fkE
5	250	250	250	250	fk5	fkF
6	500	250	250	250	fk6	fkUnused
7	750	250	250	250	fkCancel	fkUnused
8	0	500	250	250	fk7	fkUnused
9	250	500	250	250	fk8	fkUnused
10	500	500	250	250	fk9	fkUnused
11	750	500	250	250	fkEnter	fkUnused
12	0	750	250	250	fkShift	fkUnused
13	250	750	250	250	fk0	fkA
14	500	750	250	250	fkUnused	fkUnused
15	750	750	250	250	fkUnused	fkUnused

keyEntryMode == irregShift

When keyEntryMode represents an irregular shaped PIN pad the rows and columns parameters define the ratio of the width to height, i.e. square if the parameters are the

same or rectangular if Columns is larger than rows, etc. A Service Provider must return the position and size data for each key reported.

When keyEntryMode is irregShift then the values in the array must be the function keys codes for 0-9 and the shift key as defined in the funcKeyDetail parameter. Other function keys as defined by the funcKeyDetail parameter that can be enabled must also be reported. Any positions on the PIN pad that are not used must be defined as a fkUnused in the fk and shiftfk field of the hexKeys structure. Digits 0 to 9 are accessed through the numeric keys as usual. Digits A - F are accessed by using the shift key in combination with another function key, e.g. shift-0(zero) is hex digit A.

```
   1 (B)    2 (C)   3 (D)   Clear
   4 (E)   5 (F)   6   Cancel
   7   8   9   Enter
       0 (A)
                 SHIFT
```

In the above example, where the hex digits 'A' to 'F' are accessed through shift '0' to '5', columns will be 4, rows will be 5 and the hexKeys will contain the entries in the array as defined in the following table.

Index	xPos	yPos	xSize	ySize	fk	shiftfk
0	0	0	250	200	fk1	fkB
1	250	0	250	200	fk2	fkC
2	500	0	250	200	fk3	fkD
3	750	0	250	200	fkClear	fkUnused
4	0	200	250	200	fk4	fkE
5	250	200	250	200	fk5	fkF
6	500	200	250	200	fk6	fkUnused
7	750	200	250	200	fkCancel	fkUnused
8	0	400	250	200	fk7	fkUnused
9	250	400	250	200	fk8	fkUnused
10	500	400	250	200	fk9	fkUnused
11	750	400	250	200	fkEnter	fkUnused
12	0	600	250	200	fkUnused	fkUnused
13	250	600	250	200	fk0	fkA
14	500	600	250	200	fkUnused	fkUnused
15	750	600	250	200	fkUnused	fkUnused
16	0	800	1000	200	fkShift	fkUnused

keyEntryMode == irregUnique

When keyEntryMode is irregUnique then the values in the array report which physical keys are associated with the function keys 0-9, A-F and any other function keys that can be enabled as defined in the FuncKeyDetail parameter. The rows and columns parameters define the ratio of the width to height, i.e. square if the parameters are the same or rectangular if columns is larger than rows, etc. A Service Provider must return the position and size data for each key.

In the above example, where an alphanumeric keyboard supports secure key entry and the hex digits are located as shown, the hexKeys will contain the entries in the array as defined in the following table. All the hex digits and function keys that can be enabled must be included in the array; in addition any keys that would help an application display an image of the keyboard can be included. In this example only the PIN pad digits (the keys on the right) and the unique hex digits are reported. Note that the position data in this example may not be 100% accurate as the diagram is not to scale.

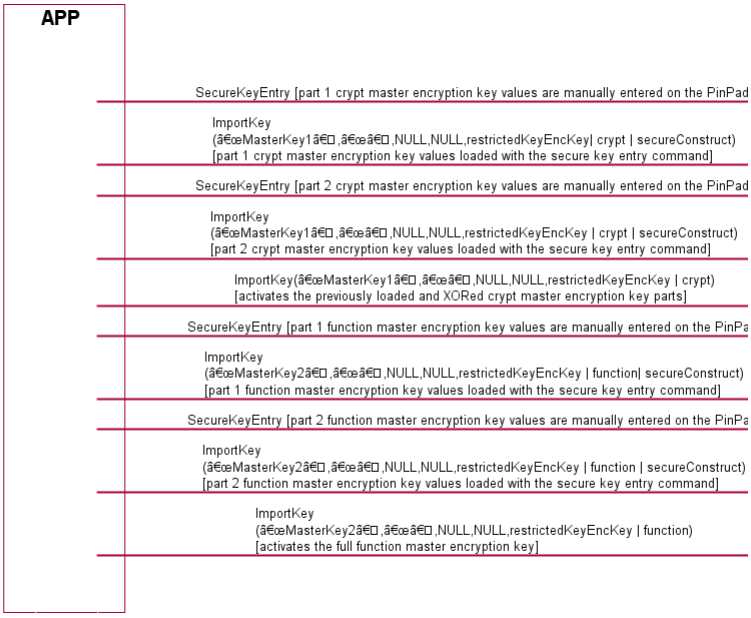
Index	xPos	yPos	xSize	ySize	fk	shiftfk
0	780	18	40	180	fk1	fkUnused
1	830	18	40	180	fk2	fkUnused
2	880	18	40	180	fk3	fkUnused
3	930	18	60	180	fkCancel	fkUnused
4	780	216	40	180	fk4	fkUnused
5	830	216	40	180	fk5	fkUnused
6	880	216	40	180	fk6	fkUnused
7	930	216	60	180	fkEnter	fkUnused
8	780	414	40	180	fk7	fkUnused
9	830	414	40	180	fk8	fkUnused
10	880	414	40	180	fk9	fkUnused
11	930	414	60	180	fkClear	fkUnused
12	780	612	40	180	fkUnused	fkUnused
13	830	612	40	180	fk0	fkUnused
14	880	612	40	180	fkUnused	fkUnused
15	930	612	60	180	fkUnused	fkUnused
16	680	810	40	180	fkA	fkUnused
17	730	810	40	180	fkB	fkUnused
18	780	810	40	180	fkC	fkUnused
19	830	810	40	180	fkD	fkUnused
20	880	810	40	180	fkE	fkUnused
21	930	810	60	180	fkF	fkUnused

Command Usage

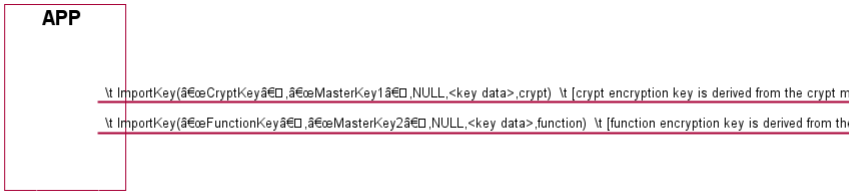
This section provides an example of the sequence of commands required to enter an encryption key securely. In the following sequence, the application retrieves the keyboard secure key entry mode and associated keyboard layout and displays an image of the keyboard for the user. It then gets the first key part, verifies the KCV for the key part and stores it. The sequence is repeated for the second key part and then finally the key part is activated.

restrictedKeyEncKey key usage

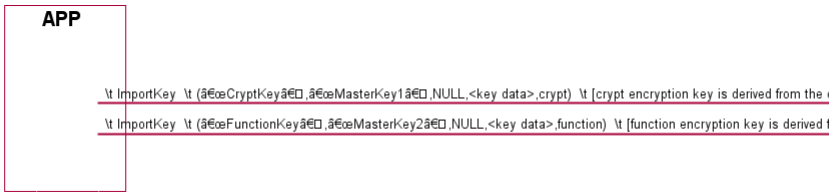
This sample command flow sequence shows how encryption keys can be derived/not derived if the master key has a restricted use. NOTE: In this example the master encryption key is loaded using the secure key entry command instead of using RKL commands. The loading with RKL works in the same way. Secure key entry based restricted master encryption key loading with RestrictedKeyEncKey flag:



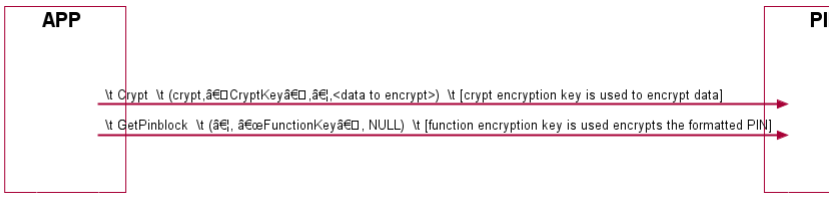
New master keys loaded with restrictedKeyEncKey flag, encrypted with themselves



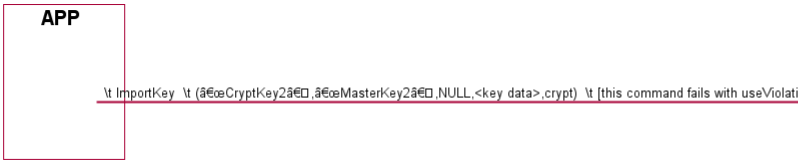
Loading derived keys:



Usage sample for derived keys



Master key restriction disallows loading of derived keys with different usage:



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
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'
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
"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

TR-31 Value	TR-31 Mode(s) of use	Definition	use
“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.

Appendix-E (DUKPT)

Definitions and Abbreviations

DUKPT	Derived Unique Key Per Transaction
BDK	Base Derivation Key
IPEK	Initial PIN Encryption Key
KSN	Key Serial Number.
TRSM	Tamper Resistant Security Module.

For additional information see reference 45.

2.1 Default Key Name

The dukpt IPEK key is given a fixed name so multi-vendor applications can be developed without the need for vendor specific configuration tools.

If dukpt is supported, this key must be included in the KeyDetail output.

Item Name Description

“dukptIpek” This key represents the IPEK, the derived future keys stored during import of the IPEK and the variant per transaction keys (PIN and optionally data and MAC).

Commands

Pinpad.SetGuidanceLight

Description

This command is used to set the status of the devices guidance lights. This includes defining the flash rate, the color and the direction. When an application tries to use a color or direction that is not supported then the Service Provider will return the generic error WFS_ERR_UNSUPP_DATA.

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.
guidLight	integer		Specifies the index of the guidance light to set as one of the values defined within the capabilities section:
command	object		
command.flashRate	string		Indicates which flash rates are supported by the guidelight.
command.color	string		Indicates which colors are supported by the guidelight.
command.direction	string		Indicates which directions are supported by the guidelight. and it'an optional field

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "guidLight": 0,
    "command": {
      "flashRate": "off",
      "color": "default",
      "direction": "entry"
    }
  }
}
```

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

Pinpad.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

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

Pinpad.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

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

Pinpad.Status

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	string		Specifies the state of the encryption module
autoBeepMode	string		Specifies whether automatic beep tone on key press is active or not. Active and in-active key beeping is reported independently. autoBeepMode can take a combination of the following values, if the flag is not set auto beeping is not activated (or not supported) for that key type (i.e. active or in-active keys)
certificateState	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",
    "autoBeepMode": "active",
    "certificateState": "unknown"
  }
}
```


Event Messages

Pinpad.Capabilities

Description

This command is used to retrieve the capabilities of the PIN pad.

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

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).
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.
type	object		Specifies the type of the PIN pad security module.
type.epp	boolean		Electronic PIN pad (keyboard data entry device).
type.edm	boolean		Encryption/decryption module.
type.hsm	boolean		Hardware security module (electronic PIN pad and encryption module within the same physical unit).
type.ets	boolean		Encrypting Touch Screen (touch screen data entry device).
keyNum	integer		Number of the keys which can be stored in the encryption/decryption module
algorithms	object		Supported encryption modes
algorithms.ecb	boolean		Electronic Code Book.
algorithms.cbc	boolean		Cipher Block Chaining.

Name	Type	Default	Description
algorithms.cfb	boolean		Cipher Feed Back.
algorithms.rsa	boolean		RSA Encryption.
algorithms.cma	boolean		ECMA Encryption.
algorithms.desMac	boolean		MAC calculation using CBC.
algorithms.triDesEcb	boolean		Triple DES with Electronic Code Book.
algorithms.triDesCbc	boolean		Triple DES with Cipher Block Chaining.
algorithms.triDesCfb	boolean		Triple DES with Cipher Feed Back.
algorithms.triDesMac	boolean		Last Block Triple DES MAC as defined in ISO/IEC 9797-1:1999 [Ref. 32], using: block length n=64, padding Method 1 (when padding=0), MAC Algorithm 3, MAC length m where 32<=m<=64.
algorithms.maaMac	boolean		MAC calculation using the Message authenticator algorithm as defined in ISO 8731-2.
algorithms.triDesMac2805	boolean		Triple DES MAC calculation as defined in ISO 16609:2004 and and Australian Standard 2805.4.
algorithms.sm4	boolean		SM4 block cipher algorithm as defined in Password industry standard of the People's Republic of China GM/T 0002-2012.
algorithms.sm4Mac	boolean		EMAC calculation using the Message authenticator algorithm as defined in as defined in Password industry standard of the People's Republic of China GM/T 0002-2012 and and in PBOC3.0 JR/T 0025.17-2013.
pinFormats	object		Supported PIN format
pinFormats.3624	boolean		PIN left justified, filled with padding characters, PIN length 4-16 digits. The padding character is a hexadecimal digit in the range 0x00 to 0x0F.
pinFormats.ansi	boolean		PIN is preceded by 0x00 and the length of the PIN (0x04 to 0x0C), filled with padding character 0x0F to the right, PIN length 4-12 digits, XORed with PAN (Primary Account Number, minimum 12 digits without check number)
pinFormats.iso0	boolean		PIN is preceded by 0x00 and the length of the PIN (0x04 to 0x0C), filled with padding character 0x0F to the right, PIN length 4-12 digits, XORed with PAN (Primary Account Number without check number, no minimum length specified, missing digits are filled with 0x00).
pinFormats.iso1	boolean		PIN is preceded by 0x01 and the length of the PIN (0x04 to 0x0C), padding characters are taken from a transaction field (10 digits).
pinFormats.eci2	boolean		PIN left justified, filled with padding characters, PIN only 4 digits.
pinFormats.eci3	boolean		PIN is preceded by the length (digit), PIN length 4-6 digits, the padding character can range from 0x0 through 0xF
pinFormats.visa	boolean		PIN is preceded by the length (digit), PIN length 4-6 digits. If the PIN length is less than six digits the PIN is filled with 0x0 to the length of six, the padding character can range from 0x0 through 0x9 (This format is also referred to as VISA2).
pinFormats.diebold	boolean		PIN is padded with the padding character and may be not encrypted, single encrypted or double encrypted.
pinFormats.dieboldCo	boolean		PIN with the length of 4 to 12 digits, each one with a value of 0x0 to 0x9, is preceded by the one-digit coordination number with a value from 0x0 to 0xF, padded with the padding character with a value from 0x0 to 0xF and may be not encrypted, single encrypted or double encrypted.
pinFormats.visa3	boolean		PIN with the length of 4 to 12 digits, each one with a value of 0x0 to 0x9, is followed by a delimiter with the value of 0xF and then padded by the padding character with a value between 0x0 to 0xF.
pinFormats.banksys	boolean		PIN is encrypted and formatted according to the Banksys PIN block specifications.
pinFormats.emv	boolean		The PIN block is constructed as follows: PIN is preceded by 0x02 and the length of the PIN (0x04 to 0x0C), filled with padding character 0x0F to the right, formatted up to 248 bytes of other data as defined within the EMV 4.0 specifications and finally encrypted with an RSA key.
pinFormats.iso3	boolean		PIN is preceded by 0x03 and the length of the PIN (0x04 to 0x0C), padding characters sequentially or randomly chosen, XORed with digits from PAN.
pinFormats.ap	boolean		PIN is formatted according to the Italian Bancomat specifications. It is known as the Authentication Parameter PIN block and is created with a 5 digit PIN, an 18 digit PAN, and the 8 digit CCS from the track data.

Name	Type	Default	Description
derivationAlgorithms	object		Supported derivation algorithms
derivationAlgorithms.chipZka	boolean		Algorithm for the derivation of a chip card individual key as described by the German ZKA.
presentationAlgorithms	object		Supported presentation algorithms
presentationAlgorithms.presentClear	boolean		Algorithm for the presentation of a clear text PIN to a chipcard. Each digit of the clear text PIN is inserted as one nibble (=halfbyte) into ChipData
display	object		Specifies the type of the display used in the PIN pad module
display.none	boolean		No display unit
display.ledThrough	boolean		Lights next to text guide user
display.display	boolean		A real display is available (this doesnâ€™t apply for self-service).
idConnect	boolean		Specifies whether the PIN pad is directly physically connected to the ID card unit. If the value is TRUE, the PIN will be transported securely during the command WFS_CMD_PIN_PRESENT_IDC
idKey	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
validationAlgorithms	object		Specifies the algorithms for PIN validation supported by the service
validationAlgorithms.des	boolean		DES algorithm
validationAlgorithms.euroCheque	boolean		euroCheque algorithm
validationAlgorithms.visa	boolean		visa algorithm
validationAlgorithms.desOffset	boolean		DES offset generation algorithm
validationAlgorithms.banksys	boolean		Banksys algorithm
keyCheckModes	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.
pinCanPersistAfterUse	boolean		Specifies whether the device can retain the PIN after a PIN processing command
autoBeep	object		Specifies whether the PIN device will emit a key beep tone on key presses of active keys or inactive keys, and if so, which mode it supports
autoBeep.activeAvailable	boolean		Automatic beep tone on active key key-press is supported. If this flag is not set then automatic beeping for active keys is not supported.
autoBeep.activeSelectable	boolean		Automatic beeping for active keys can be controlled turned on and off by the application. If this flag is not set then automatic beeping for active keys cannot be controlled by an application
autoBeep.inactiveAvailable	boolean		Automatic beep tone on in-active key keypress is supported. If this flag is not set then automatic beeping for in-active keys is not supported
autoBeep.inactiveSelectable	boolean		Automatic beeping for in-active keys can be controlled turned on and off by the application. If this flag is not set then automatic beeping for in-active keys cannot be controlled by an application.
hsmVendor	string		Identifies the hsm Vendor. hsmVendor is an empty string when the hsm Vendor is unknown or the HSM is not supported
hsmJournaling	boolean		Specifies whether the hsm supports journaling by the GetJournal command. The value of this parameter is either TRUE or FALSE. TRUE means the hsm supports journaling by GetJournal

Name	Type	Default	Description
rsaAuthenticationScheme	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	object		Specifies which type of rsa Signature Algorithm
rsaSignatureAlgorithm.pkcs1V15	boolean		pkcs1V15 Signatures supported.
rsaSignatureAlgorithm.pss	boolean		pss Signatures supported.
rsaCryptAlgorithm	object		Specifies which type of rsa Encipherment Algorith
rsaCryptAlgorithm.pkcs1V15	boolean		pkcs1V15 algorithm supported.
rsaCryptAlgorithm.oaep	boolean		oaep algorithm supported.
rsaKeyCheckMode	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	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	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	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.
keyImportThroughParts	boolean		Specifies whether the device is capable of importing keys in multiple parts. TRUE means the device supports the key import in multiple parts
encloProtocols	object		Specifies the Enclo protocols supported to communicate with the encryption module

Name	Type	Default	Description
encIoProtocols.ch	boolean		For Swiss specific protocols. The document specification for Swiss specific protocols is â€œCMD_ENC_IO - CH Protocol.docâ€? .This document is available at the following address: EUROPAY SA Terminal Management Hertistrasse 27 CH-8304 Wallisellen
encIoProtocols.giecb	boolean		Protocol for â€œGroupement des Cartes Bancairesâ€? (France).
encIoProtocols.lux	boolean		Protocol for Luxemburg commands. The reference for this specific protocol is the Authorization Center in Luxemburg (CETREL.) Cryptography Management Postal address: CETREL SociÃ©tÃ© CoopÃ©rative Centre de Transferts Electroniques L-2956 Luxembourg.
encIoProtocols.chn	boolean		Protocol for China commands. The reference for this specific protocol are the Financial industry standard of the Peopleâ€™s Republic of China PBOC3.0 JR/T 0025 and the Password industry standard of the People's Republic of China GM/T 0003, GM/T 004.
typeCombined	boolean		Specifies whether the keypad used in the secure PIN pad module is integrated within a generic Win32 keyboard. TRUE means the secure PIN keypad is integrated within a generic Win32 keyboard and standard Win32 key events will be generated for any key when there is no "active GetData or GetPin command. Note that XFS continues to support defined PIN keys only, and is not extended to support new alphanumeric keys.
setPinblockDataRequired	boolean		Specifies whether the command SetPinblockData must be called before the PIN is entered via GetPin and retrieved via GetPinblock.
keyBlockImportFormats	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.
desKeyLength	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	object		Specifies supported certificate types
certificateTypes.encKey	boolean		Supports the EPP public encryption certificate.
certificateTypes.verificationKey	boolean		Supports the EPP public verification certificate.
certificateTypes.hostKey	boolean		Supports the Host public certificate.
loadCertOptions	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	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.
etsCaps	array		Specifies the capabilities of the ets device.

Name	Type	Default	Description
etsCaps.xPos	integer		Specifies the position of the left edge of the ets in Windows virtual screen coordinates. This value may be negative because the of the monitor position on the virtual desktop.
etsCaps.yPos	integer		Specifies the position of the right edge of the ets in Windows virtual screen coordinates. This value may be negative because the of the monitor position on the virtual desktop
etsCaps.xSize	integer		Specifies the width of the ets in Windows virtual screen coordinates.
etsCaps.ySize	integer		Specifies the height of the ets in Windows virtual screen coordinates.
etsCaps.maximumTouchFrames	integer		Specifies the maximum number of Touch-Frames that the device can support in a touch keyboard definition.
etsCaps.maximumTouchKeys	integer		Specifies the maximum number of Touch-Keys that the device can support within any a touchframe.
etsCaps.floatFlags	object		Specifies if the device can float the touch keyboards. FloatNone if the PIN device cannot randomly shift the layout.
etsCaps.floatFlags.x	boolean		Specifies that the PIN device will randomly shift the layout in a horizontal direction
etsCaps.floatFlags.y	boolean		Specifies that the PIN device will randomly shift the layout in a vertical direction.
synchronizableCommands	array		command names that can be synchronized. If no execute command can be synchronized then this parameter will be an empty string.
restrictedKeyEncKeySupport	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	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).
cryptAttributes	array		Array of attributes supported by the CRYPT command.
cryptAttributes.keyUsage	string		Specifies the key usage supported by the crypt command
cryptAttributes.algorithm	string		Specifies the encryption algorithms supported by CRYPT command
cryptAttributes.modeOfUse	string		Specifies the encryption mode supported by CRYPT command.
cryptAttributes.cryptoMethod	string		Specifies the cryptographic method supported by the CRYPT command.
pinBlockAttributes	array		Array of attributes supported by the PinBlock command.
pinBlockAttributes.keyUsage	string		Specifies the key usages supported by the PINBLOCK command.
pinBlockAttributes.algorithm	string		Specifies the encryption algorithms supported by the PINBLOCK command as one of the following values
pinBlockAttributes.modeOfUse	string		Specifies the encryption modes supported by the PINBLOCK command as one of the following values

Name	Type	Default	Description
pinBlockAttributes.cryptoMethod	string		This parameter specifies the cryptographic method that will be used with the encryption algorithm specified by Algorithm.
keyAttributes	array		Array of attributes supported by ImportKey command for the key to be loaded.
keyAttributes.keyUsage	string		Specifies the Key usages supported by Import Key command.
keyAttributes.propkeyUsage	integer	0	if keyUsage is set to Numeric, proprietary value is set.
keyAttributes.algorithm	string		Specifies the encryption algorithms supported by the import command.
keyAttributes.propAlgorithm	integer	0	if algorithm is set to Numeric, proprietary value is set.
keyAttributes.modeOfUse	string		Specifies the encryption modes supported by import key.
keyAttributes.propmodeOfUse	integer	0	if modeOfUse is set to Numeric, proprietary value is set.
keyAttributes.cryptoMethod	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
decryptAttributes	array		Array of attributes supported by the Import command for the key used to decrypt or unwrap the key being imported.
decryptAttributes.algorithm	string		Specifies the encryption algorithms supported by the import command.
decryptAttributes.propAlgorithm	integer	0	if algorithm is set to Numeric, proprietary value is set.
decryptAttributes.cryptoMethod	string		This parameter specifies the cryptographic method that will be used with the encryption algorithm specified by bAlgorithm.
verifyAttributes	array		Array of attributes supported by Import command for the key used for verification before importing the key.
verifyAttributes.keyUsage	string		Specifies the key usages supported by the import command.
verifyAttributes.propkeyUsage	integer	0	if keyUsage is set to Numeric, proprietary value is set.
verifyAttributes.algorithm	string		Specifies the encryption algorithms supported by the import command.
verifyAttributes.propAlgorithm	integer	0	if algorithm is set to Numeric, proprietary value is set.
verifyAttributes.modeOfUse	string		Specifies the encryption modes supported by the import command.
verifyAttributes.propmodeOfUse	integer	0	if modeOfUse is set to Numeric, proprietary value is set.
verifyAttributes.cryptoMethod	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"
  ],
  "type": {
    "epp": true,
    "edm": true,
    "hsm": true,
    "ets": true
  },
  "keyNum": 0,
  "algorithms": {
    "ecb": true,
    "cbc": true,
    "cfb": true,
    "rsa": true,
    "cma": true,
    "desMac": true,
    "triDesEcb": true,
    "triDesCbc": true,
    "triDesCfb": true,
    "triDesMac": true,
    "maaMac": true,
    "triDesMac2805": true,
    "sm4": true,
    "sm4Mac": true
  },
  "pinFormats": {
    "3624": true,
    "ansi": true,
    "iso0": true,
    "iso1": true,
    "eci2": true,
    "eci3": true,
    "visa": true,
    "diebold": true,
    "dieboldCo": true,
    "visa3": true,
    "banksys": true,
    "emv": true,
    "iso3": true,
    "ap": true
  },
  "derivationAlgorithms": {
    "chipZka": true
  },
  "presentationAlgorithms": {
    "presentClear": true
  },
  "display": {
    "none": true,
    "ledThrough": true,
    "display": true
  },
  "idConnect": true,
  "idKey": {
    "initialization": true,
    "import": true
  },
  "validationAlgorithms": {
    "des": true,
    "euroCheque": true,
    "visa": true,
    "desOffset": true,
    "banksys": true
  },
  "keyCheckModes": {
    "self": true,
    "zero": true
  },
  "pinCanPersistAfterUse": true,
  "autoBeep": {
    "activeAvailable": true,
    "activeSelectable": true,
    "inactiveAvailable": true,
    "inactiveSelectable": true
  },
  "hsmVendor": "string",
```

```
  "hsmJournaling": true,
  "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
  },
  "keyImportThroughParts": true,
  "encIoProtocols": {
    "ch": true,
    "giecb": true,
    "lux": true,
    "chn": true
  },
  "typeCombined": true,
  "setPinblockDataRequired": true,
  "keyBlockImportFormats": {
    "ansTr31KeyBlock": true,
    "ansTr31KeyBlockB": true,
    "ansTr31KeyBlockC": 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
  },
  "etsCaps": [
    {
      "xPos": 0,
      "yPos": 0,
      "xSize": 0,
      "ySize": 0,
      "maximumTouchFrames": 0,
      "maximumTouchKeys": 0,
      "floatFlags": {
        "x": true,
        "y": true
      }
    }
  ],
  "restrictedKeyEncKeySupport": [
    {
      "loadingMethod": "rsaAuth2partySig",
```

```
        "uses": {
          "crypt": true,
          "function": true,
          "macing": true,
          "pinlocal": true,
          "svenckey": true,
          "pinremote": true
        }
      }
    ],
    "symmetricKeyManagementMethods": {
      "fixedKey": true,
      "masterKey": true,
      "tdesDukpt": true
    },
    "cryptAttributes": [],
    "pinBlockAttributes": [],
    "keyAttributes": [],
    "decryptAttributes": [],
    "verifyAttributes": []
  }
}
```


Event Messages

Pinpad.FuncKeyDetail

Description

This command returns information about the names of the Function Keys supported by the device. Location information is also returned for the supported FDKs (Function Descriptor Keys). This includes screen overlay FDKs. This command should be issued before the first call to GetPin or GetData to determine which Function Keys (FKs) and Function Descriptor Keys (FDKs) are available and where the FDKs are located. Then, in these two commands, they can then be specified as Active and Terminate keys and options on the customer screen can be aligned with the active FDKs

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.
fdkMask	integer		Mask for the fdks for which additional information is requested. If 0x00000000, only information about function keys is returned. If 0xFFFFFFFF, information about all the supported fdks is returned

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "fdkMask": 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	Type	Default	Description
name (Required)	string		The original message name, for example "CardReader.Status"

Message Payload

Name	Type	Default	Description
funcMask	object		Specifies the function keys available for this physical device.
funcMask.fk0	boolean	false	
funcMask.fk1	boolean	false	
funcMask.fk2	boolean	false	
funcMask.fk3	boolean	false	
funcMask.fk4	boolean	false	
funcMask.fk5	boolean	false	
funcMask.fk6	boolean	false	
funcMask.fk7	boolean	false	
funcMask.fk8	boolean	false	
funcMask.fk9	boolean	false	
funcMask.fkA	boolean	false	
funcMask.fkB	boolean	false	
funcMask.fkC	boolean	false	
funcMask.fkD	boolean	false	
funcMask.fkE	boolean	false	
funcMask.fkF	boolean	false	
funcMask.fkEnter	boolean	false	
funcMask.fkCancel	boolean	false	
funcMask.fkClear	boolean	false	
funcMask.fkBackspace	boolean	false	
funcMask.fkHelp	boolean	false	
funcMask.fkDecPoint	boolean	false	
funcMask.fk00	boolean	false	
funcMask.fk000	boolean	false	
funcMask.fkShift	boolean	false	
funcMask.fkRES01	boolean	false	
funcMask.fkRES02	boolean	false	
funcMask.fkRES03	boolean	false	
funcMask.fkRES04	boolean	false	
funcMask.fkRES05	boolean	false	
funcMask.fkRES06	boolean	false	
funcMask.fkRES07	boolean	false	
funcMask.fkRES08	boolean	false	
funcMask.fkOEM01	boolean	false	
funcMask.fkOEM02	boolean	false	
funcMask.fkOEM03	boolean	false	
funcMask.fkOEM04	boolean	false	
funcMask.fkOEM05	boolean	false	
funcMask.fkOEM06	boolean	false	
fdks	array		It is the responsibility of the application to identify the mapping between the FDK code and the physical location of the FDK. An empty array if no FDKs are requested or supported.
fdks.fdk	string		Specifies the code returned by this FDK, defined as one of the following values:
fdks.xPosition	integer		For FDKs, specifies the screen position the FDK relates to. This position is relative to the top of the screen expressed as a percentage of the height of the screen. For FDKs above or below the screen this will be 0 (above) or 100 (below).
fdks.yPosition	integer		For FDKs, specifies the screen position the FDK relates to. This position is relative to the Left Hand side of the screen expressed as a percentage of the width of the screen. For FDKs along the side of the screen this will be 0 (left side) or 100 (right side, user's view).

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "funcMask": {
      "fk0": false,
      "fk1": false,
      "fk2": false,
      "fk3": false,
      "fk4": false,
      "fk5": false,
      "fk6": false,
      "fk7": false,
      "fk8": false,
      "fk9": false,
      "fkA": false,
      "fkB": false,
      "fkC": false,
      "fkD": false,
      "fkE": false,
      "fkF": false,
      "fkEnter": false,
      "fkCancel": false,
      "fkClear": false,
      "fkBackspace": false,
      "fkHelp": false,
      "fkDecPoint": false,
      "fk00": false,
      "fk000": false,
      "fkShift": false,
      "fkRES01": false,
      "fkRES02": false,
      "fkRES03": false,
      "fkRES04": false,
      "fkRES05": false,
      "fkRES06": false,
      "fkRES07": false,
      "fkRES08": false,
      "fkOEM01": false,
      "fkOEM02": false,
      "fkOEM03": false,
      "fkOEM04": false,
      "fkOEM05": false,
      "fkOEM06": false
    },
    "fdks": [
      {
        "fdk": "fk_fdk01",
        "xPosition": 0,
        "yPosition": 0
      }
    ]
  }
}
```

Event Messages

Pinpad.HSMTData

Description

This function returns the current hsm terminal data. The data is returned as a series of â€œtag/length/valueâ€? items.

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.

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
tData	string		Contains the parameter settings as a series of 'tag/length/value' items with no separators. See command hsmSetTData for the tags supported.

Example Message (generated)


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


Event Messages

Pinpad.KeyDetail

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. If NULL, detailed information about all the keys in the encryption module is returned.

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.
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	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	string		Specifies the intended function of the key. See [Reference 35. ANS X9 TR-31 2018] for all possible values.

Name	Type	Default	Description
keyDetail.keyBlockInfo.algorithm	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	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	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

Pinpad.SecureKeyDetail

Description

This command reports the secure key entry method used by the device. This allows an application to enable the relevant keys and inform the user how to enter the hex digits 'A' to 'F', e.g. by displaying an image indicating which key pad locations correspond to the 16 hex digits and/or shift key. It reports the following information:

- The secure key entry mode (uses a shift key to access the hex digit 'A' to 'F' or each hex digit has a specific key assigned to it).
- The function keys and FDKs available during secure key entry.
- The FDKs that are configured as function keys (Enter, Cancel, Clear and Backspace).
- The physical keyboard layout. The keys that are active during the secure key entry command are vendor specific but must be sufficient to enter a secure encryption key. On some systems a unique key is assigned to each encryption key digit. On some systems encryption key digits are entered by pressing a shift key and then a numeric digit, e.g. to enter 'A' the shift key (fkShift) is pressed followed by the zero key (fk0). On these systems fkShift is not returned to the application in a keyEvent. The exact behavior of the shift key is vendor dependent, some devices will require the shift to be used before every key and some may require the shift key to enter and exit shift mode. There are many different styles of PIN pads in operation. Most have a regular shape with all keys having the same size and are laid out in a regular matrix. However, some devices have a layout with keys of different sizes and different numbers of keys on some rows and columns. This command returns information that allows an application to provide user instructions and an image of the keyboard layout to assist with key entry.

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
keyEntryMode	string		Specifies the method to be used to enter the encryption key digits (including 'A' to 'F') during secure key entry.
funcKeyDetail	object		Contains information about the function keys and FDKs supported by the device while in secure key entry mode. This is the same as the output of the FuncKeyDetail command with information always returned for every FDK valid during secure key entry. It describes the function keys that represent the hex digits and shift key, but also reports any other keys that can be enabled while in secure key entry mode.
funcKeyDetail.funcMask	object		Specifies the function keys available for this physical device.
funcKeyDetail.funcMask.fk0	boolean	false	
funcKeyDetail.funcMask.fk1	boolean	false	
funcKeyDetail.funcMask.fk2	boolean	false	
funcKeyDetail.funcMask.fk3	boolean	false	
funcKeyDetail.funcMask.fk4	boolean	false	
funcKeyDetail.funcMask.fk5	boolean	false	
funcKeyDetail.funcMask.fk6	boolean	false	
funcKeyDetail.funcMask.fk7	boolean	false	
funcKeyDetail.funcMask.fk8	boolean	false	
funcKeyDetail.funcMask.fk9	boolean	false	
funcKeyDetail.funcMask.fkA	boolean	false	
funcKeyDetail.funcMask.fkB	boolean	false	
funcKeyDetail.funcMask.fkC	boolean	false	
funcKeyDetail.funcMask.fkD	boolean	false	
funcKeyDetail.funcMask.fkE	boolean	false	
funcKeyDetail.funcMask.fkF	boolean	false	
funcKeyDetail.funcMask.fkEnter	boolean	false	
funcKeyDetail.funcMask.fkCancel	boolean	false	

Name	Type	Default	Description
funcKeyDetail.funcMask.fkClear	boolean	false	
funcKeyDetail.funcMask.fkBackspace	boolean	false	
funcKeyDetail.funcMask.fkHelp	boolean	false	
funcKeyDetail.funcMask.fkDecPoint	boolean	false	
funcKeyDetail.funcMask.fk00	boolean	false	
funcKeyDetail.funcMask.fk000	boolean	false	
funcKeyDetail.funcMask.fkShift	boolean	false	
funcKeyDetail.funcMask.fkRES01	boolean	false	
funcKeyDetail.funcMask.fkRES02	boolean	false	
funcKeyDetail.funcMask.fkRES03	boolean	false	
funcKeyDetail.funcMask.fkRES04	boolean	false	
funcKeyDetail.funcMask.fkRES05	boolean	false	
funcKeyDetail.funcMask.fkRES06	boolean	false	
funcKeyDetail.funcMask.fkRES07	boolean	false	
funcKeyDetail.funcMask.fkRES08	boolean	false	
funcKeyDetail.funcMask.fkOEM01	boolean	false	
funcKeyDetail.funcMask.fkOEM02	boolean	false	
funcKeyDetail.funcMask.fkOEM03	boolean	false	
funcKeyDetail.funcMask.fkOEM04	boolean	false	
funcKeyDetail.funcMask.fkOEM05	boolean	false	
funcKeyDetail.funcMask.fkOEM06	boolean	false	
funcKeyDetail.fdks	array		It is the responsibility of the application to identify the mapping between the FDK code and the physical location of the FDK. An empty array if no FDKs are requested or supported.
funcKeyDetail.fdks.fdk	string		Specifies the code returned by this FDK, defined as one of the following values:
funcKeyDetail.fdks.xPosition	integer		For FDKs, specifies the screen position the FDK relates to. This position is relative to the top of the screen expressed as a percentage of the height of the screen. For FDKs above or below the screen this will be 0 (above) or 100 (below).
funcKeyDetail.fdks.yPosition	integer		For FDKs, specifies the screen position the FDK relates to. This position is relative to the Left Hand side of the screen expressed as a percentage of the width of the screen. For FDKs along the side of the screen this will be 0 (left side) or 100 (right side, user's view).
clearFDK	object		The FDK code mask reporting any FDKs associated with Clear. If this field is zero then Clear through an FDK is not supported, otherwise the bit mask reports which FDKs are associated with Clear.
clearFDK.fdk01	boolean	false	
clearFDK.fdk02	boolean	false	
clearFDK.fdk03	boolean	false	
clearFDK.fdk04	boolean	false	
clearFDK.fdk05	boolean	false	
clearFDK.fdk06	boolean	false	
clearFDK.fdk07	boolean	false	
clearFDK.fdk08	boolean	false	
clearFDK.fdk09	boolean	false	
clearFDK.fdk10	boolean	false	
clearFDK.fdk11	boolean	false	
clearFDK.fdk12	boolean	false	
clearFDK.fdk13	boolean	false	
clearFDK.fdk14	boolean	false	
clearFDK.fdk15	boolean	false	
clearFDK.fdk16	boolean	false	
clearFDK.fdk17	boolean	false	
clearFDK.fdk18	boolean	false	
clearFDK.fdk19	boolean	false	
clearFDK.fdk20	boolean	false	
clearFDK.fdk21	boolean	false	
clearFDK.fdk22	boolean	false	
clearFDK.fdk23	boolean	false	
clearFDK.fdk24	boolean	false	
clearFDK.fdk25	boolean	false	
clearFDK.fdk26	boolean	false	
clearFDK.fdk27	boolean	false	
clearFDK.fdk28	boolean	false	
clearFDK.fdk29	boolean	false	
clearFDK.fdk30	boolean	false	
clearFDK.fdk31	boolean	false	
clearFDK.fdk32	boolean	false	
cancelFDK	object		The FDK code mask reporting any FDKs associated with Cancel. If this field is zero then Cancel through an FDK is not supported, otherwise the bit mask reports which FDKs are associated with Cancel.
cancelFDK.fdk01	boolean	false	
cancelFDK.fdk02	boolean	false	
cancelFDK.fdk03	boolean	false	
cancelFDK.fdk04	boolean	false	
cancelFDK.fdk05	boolean	false	
cancelFDK.fdk06	boolean	false	
cancelFDK.fdk07	boolean	false	

Name	Type	Default	Description
cancelFDK.fdk08	boolean	false	
cancelFDK.fdk09	boolean	false	
cancelFDK.fdk10	boolean	false	
cancelFDK.fdk11	boolean	false	
cancelFDK.fdk12	boolean	false	
cancelFDK.fdk13	boolean	false	
cancelFDK.fdk14	boolean	false	
cancelFDK.fdk15	boolean	false	
cancelFDK.fdk16	boolean	false	
cancelFDK.fdk17	boolean	false	
cancelFDK.fdk18	boolean	false	
cancelFDK.fdk19	boolean	false	
cancelFDK.fdk20	boolean	false	
cancelFDK.fdk21	boolean	false	
cancelFDK.fdk22	boolean	false	
cancelFDK.fdk23	boolean	false	
cancelFDK.fdk24	boolean	false	
cancelFDK.fdk25	boolean	false	
cancelFDK.fdk26	boolean	false	
cancelFDK.fdk27	boolean	false	
cancelFDK.fdk28	boolean	false	
cancelFDK.fdk29	boolean	false	
cancelFDK.fdk30	boolean	false	
cancelFDK.fdk31	boolean	false	
cancelFDK.fdk32	boolean	false	
backspaceFDK	object		The FDK code mask reporting any FDKs associated with Backspace. If this field is zero then Backspace through an FDK is not supported, otherwise the bit mask reports which FDKs are associated with Backspace
backspaceFDK.fdk01	boolean	false	
backspaceFDK.fdk02	boolean	false	
backspaceFDK.fdk03	boolean	false	
backspaceFDK.fdk04	boolean	false	
backspaceFDK.fdk05	boolean	false	
backspaceFDK.fdk06	boolean	false	
backspaceFDK.fdk07	boolean	false	
backspaceFDK.fdk08	boolean	false	
backspaceFDK.fdk09	boolean	false	
backspaceFDK.fdk10	boolean	false	
backspaceFDK.fdk11	boolean	false	
backspaceFDK.fdk12	boolean	false	
backspaceFDK.fdk13	boolean	false	
backspaceFDK.fdk14	boolean	false	
backspaceFDK.fdk15	boolean	false	
backspaceFDK.fdk16	boolean	false	
backspaceFDK.fdk17	boolean	false	
backspaceFDK.fdk18	boolean	false	
backspaceFDK.fdk19	boolean	false	
backspaceFDK.fdk20	boolean	false	
backspaceFDK.fdk21	boolean	false	
backspaceFDK.fdk22	boolean	false	
backspaceFDK.fdk23	boolean	false	
backspaceFDK.fdk24	boolean	false	
backspaceFDK.fdk25	boolean	false	
backspaceFDK.fdk26	boolean	false	
backspaceFDK.fdk27	boolean	false	
backspaceFDK.fdk28	boolean	false	
backspaceFDK.fdk29	boolean	false	
backspaceFDK.fdk30	boolean	false	
backspaceFDK.fdk31	boolean	false	
backspaceFDK.fdk32	boolean	false	
enterFDK	object		The FDK code mask reporting any FDKs associated with Enter. If this field is zero then Enter through an FDK is not supported, otherwise the bit mask reports which FDKs are associated with Enter.
enterFDK.fdk01	boolean	false	
enterFDK.fdk02	boolean	false	
enterFDK.fdk03	boolean	false	
enterFDK.fdk04	boolean	false	
enterFDK.fdk05	boolean	false	
enterFDK.fdk06	boolean	false	
enterFDK.fdk07	boolean	false	
enterFDK.fdk08	boolean	false	
enterFDK.fdk09	boolean	false	
enterFDK.fdk10	boolean	false	
enterFDK.fdk11	boolean	false	
enterFDK.fdk12	boolean	false	
enterFDK.fdk13	boolean	false	
enterFDK.fdk14	boolean	false	
enterFDK.fdk15	boolean	false	

Name	Type	Default	Description
enterFDK.fdk16	boolean	false	
enterFDK.fdk17	boolean	false	
enterFDK.fdk18	boolean	false	
enterFDK.fdk19	boolean	false	
enterFDK.fdk20	boolean	false	
enterFDK.fdk21	boolean	false	
enterFDK.fdk22	boolean	false	
enterFDK.fdk23	boolean	false	
enterFDK.fdk24	boolean	false	
enterFDK.fdk25	boolean	false	
enterFDK.fdk26	boolean	false	
enterFDK.fdk27	boolean	false	
enterFDK.fdk28	boolean	false	
enterFDK.fdk29	boolean	false	
enterFDK.fdk30	boolean	false	
enterFDK.fdk31	boolean	false	
enterFDK.fdk32	boolean	false	
columns	integer		Specifies the maximum number of columns on the PIN pad (the columns are defined by the x coordinate values within the hexKeys below). When the keyEntryMode parameter represents an irregular shaped keyboard the rows and columns parameters define the ratio of the width to height, i.e. square if the parameters are the same or rectangular if columns is larger than rows, etc.
rows	integer		Specifies the maximum number of rows on the PIN pad (the rows are defined by the y co-ordinate values within the hexKeys below). When the keyEntryMode parameter represents an irregular shaped keyboard the rows and columns parameters define the ratio of the width to height, i.e. square if the parameters are the same or rectangular if columns is larger than rows, etc
hexKeys	array		Array to hexKeys describes the physical keys on the PIN pad, it does not include FDKs.
hexKeys.xPos	integer		Specifies the position of the top left corner of the FK relative to the left hand side of the keyboard expressed as a value between 0 and 999, where 0 is the left edge and 999 is the right edge
hexKeys.yPos	integer		Specifies the position of the top left corner of the FK relative to the top of the keyboard expressed as a value between 0 and 999, where 0 is the top edge and 999 is the bottom edge
hexKeys.xSize	integer		Specifies the FK width expressed as a value between 1 and 1000, where 1 is the smallest possible size and 1000 is the full width of the keyboard
hexKeys.ySize	integer		Specifies the FK height expressed as a value between 1and 1000, where 1 is the smallest possible size and 1000 is the full height of the keyboard.
hexKeys.fk	object		Specifies the FK associated with the physical key in non shifted mode, empty if the key is not used.
hexKeys.fk.fk0	boolean	false	
hexKeys.fk.fk1	boolean	false	
hexKeys.fk.fk2	boolean	false	
hexKeys.fk.fk3	boolean	false	
hexKeys.fk.fk4	boolean	false	
hexKeys.fk.fk5	boolean	false	
hexKeys.fk.fk6	boolean	false	
hexKeys.fk.fk7	boolean	false	
hexKeys.fk.fk8	boolean	false	
hexKeys.fk.fk9	boolean	false	
hexKeys.fk.fkA	boolean	false	
hexKeys.fk.fkB	boolean	false	
hexKeys.fk.fkC	boolean	false	
hexKeys.fk.fkD	boolean	false	
hexKeys.fk.fkE	boolean	false	
hexKeys.fk.fkF	boolean	false	
hexKeys.fk.fkEnter	boolean	false	
hexKeys.fk.fkCancel	boolean	false	
hexKeys.fk.fkClear	boolean	false	
hexKeys.fk.fkBackspace	boolean	false	
hexKeys.fk.fkHelp	boolean	false	
hexKeys.fk.fkDecPoint	boolean	false	
hexKeys.fk.fk00	boolean	false	
hexKeys.fk.fk000	boolean	false	
hexKeys.fk.fkShift	boolean	false	
hexKeys.fk.fkRES01	boolean	false	
hexKeys.fk.fkRES02	boolean	false	
hexKeys.fk.fkRES03	boolean	false	
hexKeys.fk.fkRES04	boolean	false	
hexKeys.fk.fkRES05	boolean	false	
hexKeys.fk.fkRES06	boolean	false	
hexKeys.fk.fkRES07	boolean	false	
hexKeys.fk.fkRES08	boolean	false	
hexKeys.fk.fkOEM01	boolean	false	
hexKeys.fk.fkOEM02	boolean	false	

Name	Type	Default	Description
hexKeys.fk.fkOEM03	boolean	false	
hexKeys.fk.fkOEM04	boolean	false	
hexKeys.fk.fkOEM05	boolean	false	
hexKeys.fk.fkOEM06	boolean	false	
hexKeys.shiftFK	object		Specifies the FK code associated with the physical key in shifted mode, empty if the key is not used in shifted mode. This field will always be fkUnused when the keyEntryMode parameter indicates that keyboard does not use a shift mode.
hexKeys.shiftFK.fk0	boolean	false	
hexKeys.shiftFK.fk1	boolean	false	
hexKeys.shiftFK.fk2	boolean	false	
hexKeys.shiftFK.fk3	boolean	false	
hexKeys.shiftFK.fk4	boolean	false	
hexKeys.shiftFK.fk5	boolean	false	
hexKeys.shiftFK.fk6	boolean	false	
hexKeys.shiftFK.fk7	boolean	false	
hexKeys.shiftFK.fk8	boolean	false	
hexKeys.shiftFK.fk9	boolean	false	
hexKeys.shiftFK.fkA	boolean	false	
hexKeys.shiftFK.fkB	boolean	false	
hexKeys.shiftFK.fkC	boolean	false	
hexKeys.shiftFK.fkD	boolean	false	
hexKeys.shiftFK.fkE	boolean	false	
hexKeys.shiftFK.fkF	boolean	false	
hexKeys.shiftFK.fkEnter	boolean	false	
hexKeys.shiftFK.fkCancel	boolean	false	
hexKeys.shiftFK.fkClear	boolean	false	
hexKeys.shiftFK.fkBackspace	boolean	false	
hexKeys.shiftFK.fkHelp	boolean	false	
hexKeys.shiftFK.fkDecPoint	boolean	false	
hexKeys.shiftFK.fk00	boolean	false	
hexKeys.shiftFK.fk000	boolean	false	
hexKeys.shiftFK.fkShift	boolean	false	
hexKeys.shiftFK.fkRES01	boolean	false	
hexKeys.shiftFK.fkRES02	boolean	false	
hexKeys.shiftFK.fkRES03	boolean	false	
hexKeys.shiftFK.fkRES04	boolean	false	
hexKeys.shiftFK.fkRES05	boolean	false	
hexKeys.shiftFK.fkRES06	boolean	false	
hexKeys.shiftFK.fkRES07	boolean	false	
hexKeys.shiftFK.fkRES08	boolean	false	
hexKeys.shiftFK.fkOEM01	boolean	false	
hexKeys.shiftFK.fkOEM02	boolean	false	
hexKeys.shiftFK.fkOEM03	boolean	false	
hexKeys.shiftFK.fkOEM04	boolean	false	
hexKeys.shiftFK.fkOEM05	boolean	false	
hexKeys.shiftFK.fkOEM06	boolean	false	

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "keyEntryMode": {
      "notSupp": null
    },
    "funcKeyDetail": {
      "funcMask": {
        "fk0": false,
        "fk1": false,
        "fk2": false,
        "fk3": false,
        "fk4": false,
        "fk5": false,
        "fk6": false,
        "fk7": false,
        "fk8": false,
        "fk9": false,
        "fkA": false,
        "fkB": false,
        "fkC": false,
        "fkD": false,
        "fkE": false,
        "fkF": false,
        "fkEnter": false,
        "fkCancel": false,
```

```
      "fkClear": false,
      "fkBackspace": false,
      "fkHelp": false,
      "fkDecPoint": false,
      "fk00": false,
      "fk000": false,
      "fkShift": false,
      "fkRES01": false,
      "fkRES02": false,
      "fkRES03": false,
      "fkRES04": false,
      "fkRES05": false,
      "fkRES06": false,
      "fkRES07": false,
      "fkRES08": false,
      "fkOEM01": false,
      "fkOEM02": false,
      "fkOEM03": false,
      "fkOEM04": false,
      "fkOEM05": false,
      "fkOEM06": false
    },
    "fdks": [
      {
        "fdk": "fk_fdk01",
        "xPosition": 0,
        "yPosition": 0
      }
    ]
  },
  "clearFDK": {
    "fdk01": false,
    "fdk02": false,
    "fdk03": false,
    "fdk04": false,
    "fdk05": false,
    "fdk06": false,
    "fdk07": false,
    "fdk08": false,
    "fdk09": false,
    "fdk10": false,
    "fdk11": false,
    "fdk12": false,
    "fdk13": false,
    "fdk14": false,
    "fdk15": false,
    "fdk16": false,
    "fdk17": false,
    "fdk18": false,
    "fdk19": false,
    "fdk20": false,
    "fdk21": false,
    "fdk22": false,
    "fdk23": false,
    "fdk24": false,
    "fdk25": false,
    "fdk26": false,
    "fdk27": false,
    "fdk28": false,
    "fdk29": false,
    "fdk30": false,
    "fdk31": false,
    "fdk32": false
  },
  "cancelFDK": {
    "fdk01": false,
    "fdk02": false,
    "fdk03": false,
    "fdk04": false,
    "fdk05": false,
    "fdk06": false,
    "fdk07": false,
    "fdk08": false,
    "fdk09": false,
    "fdk10": false,
    "fdk11": false,
    "fdk12": false,
    "fdk13": false,
    "fdk14": false,
    "fdk15": false,
    "fdk16": false,
    "fdk17": false,
    "fdk18": false,
    "fdk19": false,
    "fdk20": false,
    "fdk21": false,
    "fdk22": false,
    "fdk23": false,
    "fdk24": false,
```

```
      "fdk25": false,
      "fdk26": false,
      "fdk27": false,
      "fdk28": false,
      "fdk29": false,
      "fdk30": false,
      "fdk31": false,
      "fdk32": false
    },
    "backspaceFDK": {
      "fdk01": false,
      "fdk02": false,
      "fdk03": false,
      "fdk04": false,
      "fdk05": false,
      "fdk06": false,
      "fdk07": false,
      "fdk08": false,
      "fdk09": false,
      "fdk10": false,
      "fdk11": false,
      "fdk12": false,
      "fdk13": false,
      "fdk14": false,
      "fdk15": false,
      "fdk16": false,
      "fdk17": false,
      "fdk18": false,
      "fdk19": false,
      "fdk20": false,
      "fdk21": false,
      "fdk22": false,
      "fdk23": false,
      "fdk24": false,
      "fdk25": false,
      "fdk26": false,
      "fdk27": false,
      "fdk28": false,
      "fdk29": false,
      "fdk30": false,
      "fdk31": false,
      "fdk32": false
    },
    "enterFDK": {
      "fdk01": false,
      "fdk02": false,
      "fdk03": false,
      "fdk04": false,
      "fdk05": false,
      "fdk06": false,
      "fdk07": false,
      "fdk08": false,
      "fdk09": false,
      "fdk10": false,
      "fdk11": false,
      "fdk12": false,
      "fdk13": false,
      "fdk14": false,
      "fdk15": false,
      "fdk16": false,
      "fdk17": false,
      "fdk18": false,
      "fdk19": false,
      "fdk20": false,
      "fdk21": false,
      "fdk22": false,
      "fdk23": false,
      "fdk24": false,
      "fdk25": false,
      "fdk26": false,
      "fdk27": false,
      "fdk28": false,
      "fdk29": false,
      "fdk30": false,
      "fdk31": false,
      "fdk32": false
    },
    "columns": 0,
    "rows": 0,
    "hexKeys": [
      {
        "xPos": 0,
        "yPos": 0,
        "xSize": 0,
        "ySize": 0,
        "fk": {
          "fk0": false,
          "fk1": false,
```

```
          "fk2": false,
          "fk3": false,
          "fk4": false,
          "fk5": false,
          "fk6": false,
          "fk7": false,
          "fk8": false,
          "fk9": false,
          "fkA": false,
          "fkB": false,
          "fkC": false,
          "fkD": false,
          "fkE": false,
          "fkF": false,
          "fkEnter": false,
          "fkCancel": false,
          "fkClear": false,
          "fkBackspace": false,
          "fkHelp": false,
          "fkDecPoint": false,
          "fk00": false,
          "fk000": false,
          "fkShift": false,
          "fkRES01": false,
          "fkRES02": false,
          "fkRES03": false,
          "fkRES04": false,
          "fkRES05": false,
          "fkRES06": false,
          "fkRES07": false,
          "fkRES08": false,
          "fkOEM01": false,
          "fkOEM02": false,
          "fkOEM03": false,
          "fkOEM04": false,
          "fkOEM05": false,
          "fkOEM06": false
        },
        "shiftFK": {
          "fk0": false,
          "fk1": false,
          "fk2": false,
          "fk3": false,
          "fk4": false,
          "fk5": false,
          "fk6": false,
          "fk7": false,
          "fk8": false,
          "fk9": false,
          "fkA": false,
          "fkB": false,
          "fkC": false,
          "fkD": false,
          "fkE": false,
          "fkF": false,
          "fkEnter": false,
          "fkCancel": false,
          "fkClear": false,
          "fkBackspace": false,
          "fkHelp": false,
          "fkDecPoint": false,
          "fk00": false,
          "fk000": false,
          "fkShift": false,
          "fkRES01": false,
          "fkRES02": false,
          "fkRES03": false,
          "fkRES04": false,
          "fkRES05": false,
          "fkRES06": false,
          "fkRES07": false,
          "fkRES08": false,
          "fkOEM01": false,
          "fkOEM02": false,
          "fkOEM03": false,
          "fkOEM04": false,
          "fkOEM05": false,
          "fkOEM06": false
        }
      }
    ]
  }
}
```


Event Messages

Pinpad.QueryLogicalHSMDetail

Description

This command reports the ZKA logical hsm available within the EPP. It also reports which logical HSM is currently active.

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
activeLogicalHSM	integer		Specifies the serial number of the logical hsm that is currently active. This value is the hsm serial number (tag CB in the hsm TDATA) encoded as a normal binary value (i.e. it is not a bcd). If no logical HSMs are present or logical hsm are not supported then this value is zero.
hsmInfo	array		array of hsmInfo(one for each logical HSM). A empty is returned if no logical HSMs are supported/present.
hsmInfo.hsmSerialNumber	integer		Specifies the Serial Number of the Logical HSM (tag CB in the hsm tData). This value is encoded as a normal binary value (i.e. it is not a BCD).
hsmInfo.zkaid	string		A string containing the ZKA ID of the logical HSM (defined by tag CC in the hsm tData). The characters in the string are EBCIDIC characters

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "activeLogicalHSM": 0,
    "hsmInfo": [
      {
        "hsmSerialNumber": 0,
        "zkaid": "string"
      }
    ]
  }
}
```

Event Messages

Pinpad.QueryPCIPTSDeviceId

Description

This command is used to report information in order to verify the PCI Security Standards Council PIN transaction security (PTS) certification held by the PIN device. The command provides detailed information in order to verify the certification level of the device. Support of this command by the Service Provider does not imply in anyway the certification level achieved by 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

Name	Type	Default	Description
errorDescription	string		If error, identified that cause of the error
manufacturerIdentifier	string		Returns an ASCII string containing the manufacturer identifier of the PIN device. This value is NULL if the manufacturer identifier is not available. This field is distinct from the hsm key pair that may be reported in the extra field by the Capabilities command.
modelIdentifier	string		Returns an ASCII string containing the model identifier of the PIN device. This value is NULL if the model identifier is not available.
hardwareIdentifier	string		Returns an ASCII string containing the hardware identifier of the PIN device. This value is NULL if the hardware identifier is not available.
firmwareIdentifier	string		Returns an ASCII string containing the firmware identifier of the PIN device. This value is NULL if the firmware identifier is not available.
applicationIdentifier	string		Returns an ASCII string containing the application identifier of the PIN device. This value is NULL if the application identifier is not available.

Example Message (generated)


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


Event Messages

Pinpad.GetLayout

Description

This command allows an application to retrieve layout information for any PIN device. Either one layout or all defined layouts can be retrieved with a single request of this command. There can be a layout for each of the different types of keyboard entry modes, if the vendor and the hardware support these different methods. The types of keyboard entry modes are (1) Data Entry mode which corresponds to the GetData command, (2) PIN Entry mode which corresponds to the GetPin command, and (3) Secure Key Entry mode which corresponds to the SecureKeyEntry command. The layouts can be preloaded into the device, if the device supports this, or a single layout can be loaded into the device immediately prior to the keyboard command being requested.

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.
entryMode	string		Specifies entry mode to be returned

Example Message (generated)

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

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
entryMode	object		Specifies entry mode to be returned. It can be one of the following flags, or zero to return all supported entry modes
entryMode.data	boolean	false	Specifies that the layout be applied to the GetData entry method.
entryMode.pin	boolean	false	Specifies that the layout be applied to the GetPin entry method.
entryMode.secure	boolean	false	Specifies that the layout be applied to the SecurekeyEntry entry method.
frames	array		There can be one or more frame structures included
frames.xPos	integer		For ETS, specifies the left coordinate of the frame as an offset from the left edge of the screen. For all other device types, this value is ignored
frames.yPos	integer		For ETS, specifies the top coordinate of the frame as an offset from the top edge of the screen. For all other device types, this value is ignored
frames.xSize	integer		For ETS, specifies the width of the frame. For all other device types, this value is ignored
frames.ySize	integer		For ETS, specifies the height of the frame. For all other device types, this value is ignored
frames.floatAction	object		Specifies if the device can float the touch keyboards
frames.floatAction.floatX	boolean	false	Specifies that the PIN device will randomly shift the layout in a horizontal direction
frames.floatAction.floatY	boolean	false	Specifies that the PIN device will randomly shift the layout in a vertical direction
frames.fks	array		Defining details of the keys in the keyboard.
frames.fks.xPos	integer		Specifies the position of the top left corner of the FK relative to the left hand side of the layout. For ETS devices, must be in the range defined in the frame. For non-ETS devices, must be a value between 0 and 999, where 0 is the left edge and 999 is the right edge.
frames.fks.yPos	integer		Specifies the position of the top left corner of the FK relative to the left hand side of the layout. For ETS devices, must be in the range defined in the frame. For non-ETS devices, must be a value between 0 and 999, where 0 is the top edge and 999 is the bottom edge
frames.fks.xSize	integer		Specifies the FK width. For ETS, width is measured in pixels. For non-ETS devices, width is expressed as a value between 1 and 1000, where 1 is the smallest possible size and 1000 is the full width of the layout.
frames.fks.ySize	integer		Specifies the FK height. For ETS, height is measured in pixels. For non-ETS devices, height is expressed as a value between 1 and 1000, where 1 is the smallest possible size and 1000 is the full height of the layout.
frames.fks.keyType	string		Defines the type of XFS key definition value is represented by fk and shiftFK
frames.fks.fk	integer		Specifies the FK code associated with the physical area in non-shifted mode, WFS_PIN_FK_UNUSED if the key is not used.

Name	Type	Default	Description
frames.fks.shiftFK	integer		Specifies the FK code associated with the physical key in shifted mode, WFS_PIN_FK_UNUSED if the key is not used in shifted mode.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "entryMode": {
      "data": false,
      "pin": false,
      "secure": false
    },
    "frames": [
      {
        "xPos": 0,
        "yPos": 0,
        "xSize": 0,
        "ySize": 0,
        "floatAction": {
          "floatX": false,
          "floatY": false
        },
        "fks": [
          {
            "xPos": 0,
            "yPos": 0,
            "xSize": 0,
            "ySize": 0,
            "keyType": "fk",
            "fk": 0,
            "shiftFK": 0
          }
        ]
      }
    ]
  }
}
```


Event Messages

Pinpad.Crypt

Description

The input data is either encrypted or decrypted using the specified or selected encryption mode. The available modes are defined in the Capabilities command. This command can also be used for random number generation. For random number generation, the Crypt command should be used. This command cannot be used with externally encrypted keys, which can be specified using the EncKey parameter of the crypt command. This command can be used for Message Authentication Code generation and verification (i.e. macing). The input data is padded to the necessary length mandated by the encryption algorithm using the padding parameter. This command can be used for asymmetric signature generation and verification. This input data is padded to necessary length mandated by the signature algorithm using padding parameter. Applications can use an alternative padding method by pre-formatting the data passed and combining this with the standard padding method. The start value (or Initialization Vector) can be provided as input data to this command, or it can be imported via TR-31 prior to requesting this command and referenced by name. The start value and start value key are both optional parameters. "

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 stored key.

Name	Type	Default	Description
startValueKey	string		Specifies the name of the stored key used to decrypt the startValue to obtain the initialization vector. If this parameter is an empty string, startValue is used as the initialization vector. This value is ignored, if mode equals random.
startValue	string		DES and Triple DES initialization vector for cbc / cfb encryption and macing. If this parameter is an empty string the default value for cbc / cfb / mac is 16 hex digits 0x0. This value is ignored, if mode equals random.
padding	integer		Specifies the padding character. The padding character is a full byte, e.g. 0xFF. This value is ignored, if mode equals random. The valid range is 0x00 to 0xFF.
compression	boolean		Specifies whether data is to be compressed (blanks removed) before building the mac. If compression is 0x00 no compression is selected, otherwise compression holds the representation of the blank character (e.g. 0x20 in ASCII or 0x40 in EBCDIC). This value is ignored, if mode equals random.
cryptData	string		The data to be encrypted, decrypted, or maced formatted in base64. This value is ignored, if mode equals random.
verifyData	string		If the modeOfUse is 'e', 'd', 'g', or 's', then this parameter must be NULL.
cryptAttributes	object		This parameter specifies the encryption algorithm, cryptographic method, and mode to be used for this command. For a list of valid values see the Attributes capability field. The values specified must be compatible with the key identified by Key.
cryptAttributes.keyUsage	string		Specifies the key usage supported by the crypt command
cryptAttributes.algorithm	string		Specifies the encryption algorithms supported by CRYPT command
cryptAttributes.modeOfUse	string		Specifies the encryption mode supported by CRYPT command.
cryptAttributes.cryptoMethod	string		Specifies the cryptographic method supported by the CRYPT command.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "key": "string",
    "startValueKey": "string",
    "startValue": "string",
    "padding": 0,
    "compression": true,
    "cryptData": "string",
    "verifyData": "string",
    "cryptAttributes": {
      "keyUsage": "d0",
      "algorithm": "a",
      "modeOfUse": "d",
      "cryptoMethod": "ecb"
    }
  }
}
```


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
cryptData	string		The encrypted or decrypted data, mac value or signature. This parameter will be NULL if the cryptAttributes.modeOfUse is ï¿½Vï¿½.

Example Message (generated)

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

Event Messages

Pinpad.GetPin

Description

This function stores the pin entry via the pin pad. From the point this function is invoked, pin digit entries are not passed to the application. For each pin digit, or any other active key entered, an execute notification event keyEvent is sent in order to allow an application to perform the appropriate display action (i.e. when the pin pad has no integrated display). The application is not informed of the value entered. The execute notification only informs that a key has been depressed. The EnterDataEvent will be generated when the PIN pad is ready for the user to start entering data. Some PIN pad devices do not inform the application as each PIN digit is entered, but locally process the PIN entry based upon minimum pin length and maximum PIN length input parameters. When the maximum number of pin digits is entered and the flag autoEnd is true, or a terminating key is pressed after the minimum number of pin digits is entered, the command completes. If the <Cancel> key is a terminator key and is pressed, then the command will complete successfully even if the minimum number of pin digits has not been entered. Terminating FDKs can have the functionality of <Enter> (terminates only if minimum length has been reached) or <Cancel> (can terminate before minimum length is reached). The configuration of this functionality is vendor specific. If maxLen is zero, the Service Provider does not terminate the command unless the application sets terminateKeys or terminateFDKs. In the event that terminateKeys or terminateFDKs are not set and maxLen is zero, the command will not terminate and the application must issue a Cancel command. If active the fkCancel and fkClear keys will cause the PIN buffer to be cleared. The fkBackspace key will cause the last key in the PIN buffer to be removed. Terminating keys have to be active keys to operate. If this command is cancelled by a CancelAsyncRequest the PIN buffer is not cleared. If maxLen has been met and autoEnd is set to False, then all numeric keys will automatically be disabled. If the clear or backspace key is pressed to reduce the number of entered keys, the numeric keys will be re-enabled. If the enter key (or FDK representing the enter key â€“ note that the association of an FDK to enter functionality is vendor specific) is pressed prior to minLen being met, then the enter key or FDK is ignored. In some cases the PIN pad device cannot ignore the enter key then the command will complete normally. To handle these types of devices the application should use the output parameter digits field to check that sufficient digits have been entered. The application should then get the user to re-enter their PIN with the correct number of digits. If the application makes a call to GetPinblock or a local verification command without the minimum PIN digits having been entered, either the command will fail or the PIN verification will fail. It is the responsibility of the application to identify the mapping between the FDK code and the physical location of the FDK.

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.
minLen	integer		Specifies the minimum number of digits which must be entered for the PIN. A value of zero indicates no minimum PIN length verification
maxLen	integer		Specifies the maximum number of digits which can be entered for the PIN. A value of zero indicates no maximum PIN length verification.
autoEnd	boolean		If autoEnd is set to true, the Service Provider terminates the command when the maximum number of digits are entered. Otherwise, the input is terminated by the user using one of the termination keys. autoEnd is ignored when maxLen is set to zero.
echo	integer		Specifies the replace character to be echoed on a local display for the PIN digit.
activeFDKs	object		Specifies a mask of those FDKs which are active during the execution of the command
activeFDKs.fdk01	boolean	false	
activeFDKs.fdk02	boolean	false	
activeFDKs.fdk03	boolean	false	
activeFDKs.fdk04	boolean	false	
activeFDKs.fdk05	boolean	false	
activeFDKs.fdk06	boolean	false	
activeFDKs.fdk07	boolean	false	
activeFDKs.fdk08	boolean	false	

Name	Type	Default	Description
activeFDKs.fdk09	boolean	false	
activeFDKs.fdk10	boolean	false	
activeFDKs.fdk11	boolean	false	
activeFDKs.fdk12	boolean	false	
activeFDKs.fdk13	boolean	false	
activeFDKs.fdk14	boolean	false	
activeFDKs.fdk15	boolean	false	
activeFDKs.fdk16	boolean	false	
activeFDKs.fdk17	boolean	false	
activeFDKs.fdk18	boolean	false	
activeFDKs.fdk19	boolean	false	
activeFDKs.fdk20	boolean	false	
activeFDKs.fdk21	boolean	false	
activeFDKs.fdk22	boolean	false	
activeFDKs.fdk23	boolean	false	
activeFDKs.fdk24	boolean	false	
activeFDKs.fdk25	boolean	false	
activeFDKs.fdk26	boolean	false	
activeFDKs.fdk27	boolean	false	
activeFDKs.fdk28	boolean	false	
activeFDKs.fdk29	boolean	false	
activeFDKs.fdk30	boolean	false	
activeFDKs.fdk31	boolean	false	
activeFDKs.fdk32	boolean	false	
activeKeys	object		Specifies a mask of those (other) Function Keys which are active during the execution of the command
activeKeys.fk0	boolean	false	
activeKeys.fk1	boolean	false	
activeKeys.fk2	boolean	false	
activeKeys.fk3	boolean	false	
activeKeys.fk4	boolean	false	
activeKeys.fk5	boolean	false	
activeKeys.fk6	boolean	false	
activeKeys.fk7	boolean	false	
activeKeys.fk8	boolean	false	
activeKeys.fk9	boolean	false	
activeKeys.fkA	boolean	false	
activeKeys.fkB	boolean	false	
activeKeys.fkC	boolean	false	
activeKeys.fkD	boolean	false	
activeKeys.fkE	boolean	false	
activeKeys.fkF	boolean	false	
activeKeys.fkEnter	boolean	false	
activeKeys.fkCancel	boolean	false	
activeKeys.fkClear	boolean	false	
activeKeys.fkBackspace	boolean	false	
activeKeys.fkHelp	boolean	false	
activeKeys.fkDecPoint	boolean	false	
activeKeys.fk00	boolean	false	
activeKeys.fk000	boolean	false	
activeKeys.fkShift	boolean	false	
activeKeys.fkRES01	boolean	false	
activeKeys.fkRES02	boolean	false	
activeKeys.fkRES03	boolean	false	
activeKeys.fkRES04	boolean	false	
activeKeys.fkRES05	boolean	false	
activeKeys.fkRES06	boolean	false	
activeKeys.fkRES07	boolean	false	
activeKeys.fkRES08	boolean	false	
activeKeys.fkOEM01	boolean	false	
activeKeys.fkOEM02	boolean	false	
activeKeys.fkOEM03	boolean	false	
activeKeys.fkOEM04	boolean	false	
activeKeys.fkOEM05	boolean	false	
activeKeys.fkOEM06	boolean	false	
terminateFDKs	object		Specifies a mask of those FDKs which must terminate the execution of the command
terminateFDKs.fdk01	boolean	false	
terminateFDKs.fdk02	boolean	false	
terminateFDKs.fdk03	boolean	false	
terminateFDKs.fdk04	boolean	false	
terminateFDKs.fdk05	boolean	false	
terminateFDKs.fdk06	boolean	false	
terminateFDKs.fdk07	boolean	false	
terminateFDKs.fdk08	boolean	false	
terminateFDKs.fdk09	boolean	false	
terminateFDKs.fdk10	boolean	false	
terminateFDKs.fdk11	boolean	false	

Name	Type	Default	Description
terminateFDKs.fdk12	boolean	false	
terminateFDKs.fdk13	boolean	false	
terminateFDKs.fdk14	boolean	false	
terminateFDKs.fdk15	boolean	false	
terminateFDKs.fdk16	boolean	false	
terminateFDKs.fdk17	boolean	false	
terminateFDKs.fdk18	boolean	false	
terminateFDKs.fdk19	boolean	false	
terminateFDKs.fdk20	boolean	false	
terminateFDKs.fdk21	boolean	false	
terminateFDKs.fdk22	boolean	false	
terminateFDKs.fdk23	boolean	false	
terminateFDKs.fdk24	boolean	false	
terminateFDKs.fdk25	boolean	false	
terminateFDKs.fdk26	boolean	false	
terminateFDKs.fdk27	boolean	false	
terminateFDKs.fdk28	boolean	false	
terminateFDKs.fdk29	boolean	false	
terminateFDKs.fdk30	boolean	false	
terminateFDKs.fdk31	boolean	false	
terminateFDKs.fdk32	boolean	false	
terminateKeys	object		Specifies a mask of those (other) Function Keys which must terminate the execution of the command
terminateKeys.fk0	boolean	false	
terminateKeys.fk1	boolean	false	
terminateKeys.fk2	boolean	false	
terminateKeys.fk3	boolean	false	
terminateKeys.fk4	boolean	false	
terminateKeys.fk5	boolean	false	
terminateKeys.fk6	boolean	false	
terminateKeys.fk7	boolean	false	
terminateKeys.fk8	boolean	false	
terminateKeys.fk9	boolean	false	
terminateKeys.fkA	boolean	false	
terminateKeys.fkB	boolean	false	
terminateKeys.fkC	boolean	false	
terminateKeys.fkD	boolean	false	
terminateKeys.fkE	boolean	false	
terminateKeys.fkF	boolean	false	
terminateKeys.fkEnter	boolean	false	
terminateKeys.fkCancel	boolean	false	
terminateKeys.fkClear	boolean	false	
terminateKeys.fkBackspace	boolean	false	
terminateKeys.fkHelp	boolean	false	
terminateKeys.fkDecPoint	boolean	false	
terminateKeys.fk00	boolean	false	
terminateKeys.fk000	boolean	false	
terminateKeys.fkShift	boolean	false	
terminateKeys.fkRES01	boolean	false	
terminateKeys.fkRES02	boolean	false	
terminateKeys.fkRES03	boolean	false	
terminateKeys.fkRES04	boolean	false	
terminateKeys.fkRES05	boolean	false	
terminateKeys.fkRES06	boolean	false	
terminateKeys.fkRES07	boolean	false	
terminateKeys.fkRES08	boolean	false	
terminateKeys.fkOEM01	boolean	false	
terminateKeys.fkOEM02	boolean	false	
terminateKeys.fkOEM03	boolean	false	
terminateKeys.fkOEM04	boolean	false	
terminateKeys.fkOEM05	boolean	false	
terminateKeys.fkOEM06	boolean	false	

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "minLen": 0,
    "maxLen": 0,
    "autoEnd": true,
    "echo": 0,
    "activeFDKs": {
      "fdk01": false,
      "fdk02": false,
      "fdk03": false,
```

```
    "fdk04": false,
    "fdk05": false,
    "fdk06": false,
    "fdk07": false,
    "fdk08": false,
    "fdk09": false,
    "fdk10": false,
    "fdk11": false,
    "fdk12": false,
    "fdk13": false,
    "fdk14": false,
    "fdk15": false,
    "fdk16": false,
    "fdk17": false,
    "fdk18": false,
    "fdk19": false,
    "fdk20": false,
    "fdk21": false,
    "fdk22": false,
    "fdk23": false,
    "fdk24": false,
    "fdk25": false,
    "fdk26": false,
    "fdk27": false,
    "fdk28": false,
    "fdk29": false,
    "fdk30": false,
    "fdk31": false,
    "fdk32": false
  },
  "activeKeys": {
    "fk0": false,
    "fk1": false,
    "fk2": false,
    "fk3": false,
    "fk4": false,
    "fk5": false,
    "fk6": false,
    "fk7": false,
    "fk8": false,
    "fk9": false,
    "fkA": false,
    "fkB": false,
    "fkC": false,
    "fkD": false,
    "fkE": false,
    "fkF": false,
    "fkEnter": false,
    "fkCancel": false,
    "fkClear": false,
    "fkBackspace": false,
    "fkHelp": false,
    "fkDecPoint": false,
    "fk00": false,
    "fk000": false,
    "fkShift": false,
    "fkRES01": false,
    "fkRES02": false,
    "fkRES03": false,
    "fkRES04": false,
    "fkRES05": false,
    "fkRES06": false,
    "fkRES07": false,
    "fkRES08": false,
    "fkOEM01": false,
    "fkOEM02": false,
    "fkOEM03": false,
    "fkOEM04": false,
    "fkOEM05": false,
    "fkOEM06": false
  },
  "terminateFDKs": {
    "fdk01": false,
    "fdk02": false,
    "fdk03": false,
    "fdk04": false,
    "fdk05": false,
    "fdk06": false,
    "fdk07": false,
    "fdk08": false,
    "fdk09": false,
    "fdk10": false,
    "fdk11": false,
    "fdk12": false,
    "fdk13": false,
    "fdk14": false,
    "fdk15": false,
    "fdk16": false,
    "fdk17": false
```

```
      "fdk18": false,
      "fdk19": false,
      "fdk20": false,
      "fdk21": false,
      "fdk22": false,
      "fdk23": false,
      "fdk24": false,
      "fdk25": false,
      "fdk26": false,
      "fdk27": false,
      "fdk28": false,
      "fdk29": false,
      "fdk30": false,
      "fdk31": false,
      "fdk32": false
    },
    "terminateKeys": {
      "fk0": false,
      "fk1": false,
      "fk2": false,
      "fk3": false,
      "fk4": false,
      "fk5": false,
      "fk6": false,
      "fk7": false,
      "fk8": false,
      "fk9": false,
      "fkA": false,
      "fkB": false,
      "fkC": false,
      "fkD": false,
      "fkE": false,
      "fkF": false,
      "fkEnter": false,
      "fkCancel": false,
      "fkClear": false,
      "fkBackspace": false,
      "fkHelp": false,
      "fkDecPoint": false,
      "fk00": false,
      "fk000": false,
      "fkShift": false,
      "fkRES01": false,
      "fkRES02": false,
      "fkRES03": false,
      "fkRES04": false,
      "fkRES05": false,
      "fkRES06": false,
      "fkRES07": false,
      "fkRES08": false,
      "fkOEM01": false,
      "fkOEM02": false,
      "fkOEM03": false,
      "fkOEM04": false,
      "fkOEM05": false,
      "fkOEM06": false
    }
  }
}
```

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
digits	integer		Specifies the number of PIN digits entered

Name	Type	Default	Description
completion	string		Specifies the reason for completion of the entry. Unless otherwise specified the following values must not be used in the execute event PinKey or in the array of keys in the completion of GetData

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "digits": 0,
    "completion": "auto"
  }
}
```


Event Messages

- Pinpad.KeyEvent
- Pinpad.EnterDataEvent

Pinpad.GetData

Description

This function takes the account information and a PIN entered by the user to build a formatted PIN. Encrypting this formatted PIN once or twice returns a PIN block which can be written on a magnetic card or sent to a host. The PIN block can be calculated using one of the formats specified in the Capabilities command. This command will clear the pin unless the application has requested that the pin be maintained through the MaintainPin 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.
maxLen	integer		Specifies the maximum number of digits which can be returned to the application in the output parameter.
autoEnd	boolean		If autoEnd is set to true, the Service Provider terminates the command when the maximum number of digits are entered. Otherwise, the input is terminated by the user using one of the termination keys. autoEnd is ignored when maxLen is set to zero
activeFDKs	object		Specifies a mask of those FDKs which are active during the execution of the command.
activeFDKs.fdk01	boolean	false	
activeFDKs.fdk02	boolean	false	
activeFDKs.fdk03	boolean	false	
activeFDKs.fdk04	boolean	false	
activeFDKs.fdk05	boolean	false	
activeFDKs.fdk06	boolean	false	
activeFDKs.fdk07	boolean	false	
activeFDKs.fdk08	boolean	false	
activeFDKs.fdk09	boolean	false	
activeFDKs.fdk10	boolean	false	
activeFDKs.fdk11	boolean	false	
activeFDKs.fdk12	boolean	false	
activeFDKs.fdk13	boolean	false	
activeFDKs.fdk14	boolean	false	
activeFDKs.fdk15	boolean	false	
activeFDKs.fdk16	boolean	false	
activeFDKs.fdk17	boolean	false	
activeFDKs.fdk18	boolean	false	
activeFDKs.fdk19	boolean	false	

Name	Type	Default	Description
activeFDKs.fdk20	boolean	false	
activeFDKs.fdk21	boolean	false	
activeFDKs.fdk22	boolean	false	
activeFDKs.fdk23	boolean	false	
activeFDKs.fdk24	boolean	false	
activeFDKs.fdk25	boolean	false	
activeFDKs.fdk26	boolean	false	
activeFDKs.fdk27	boolean	false	
activeFDKs.fdk28	boolean	false	
activeFDKs.fdk29	boolean	false	
activeFDKs.fdk30	boolean	false	
activeFDKs.fdk31	boolean	false	
activeFDKs.fdk32	boolean	false	
activeKeys	object		Specifies a mask of those (other) Function Keys which are active during the execution of the command.
activeKeys.fk0	boolean	false	
activeKeys.fk1	boolean	false	
activeKeys.fk2	boolean	false	
activeKeys.fk3	boolean	false	
activeKeys.fk4	boolean	false	
activeKeys.fk5	boolean	false	
activeKeys.fk6	boolean	false	
activeKeys.fk7	boolean	false	
activeKeys.fk8	boolean	false	
activeKeys.fk9	boolean	false	
activeKeys.fkA	boolean	false	
activeKeys.fkB	boolean	false	
activeKeys.fkC	boolean	false	
activeKeys.fkD	boolean	false	
activeKeys.fkE	boolean	false	
activeKeys.fkF	boolean	false	
activeKeys.fkEnter	boolean	false	
activeKeys.fkCancel	boolean	false	
activeKeys.fkClear	boolean	false	
activeKeys.fkBackspace	boolean	false	
activeKeys.fkHelp	boolean	false	
activeKeys.fkDecPoint	boolean	false	
activeKeys.fk00	boolean	false	
activeKeys.fk000	boolean	false	
activeKeys.fkShift	boolean	false	
activeKeys.fkRES01	boolean	false	
activeKeys.fkRES02	boolean	false	
activeKeys.fkRES03	boolean	false	
activeKeys.fkRES04	boolean	false	
activeKeys.fkRES05	boolean	false	
activeKeys.fkRES06	boolean	false	
activeKeys.fkRES07	boolean	false	
activeKeys.fkRES08	boolean	false	
activeKeys.fkOEM01	boolean	false	
activeKeys.fkOEM02	boolean	false	
activeKeys.fkOEM03	boolean	false	
activeKeys.fkOEM04	boolean	false	
activeKeys.fkOEM05	boolean	false	
activeKeys.fkOEM06	boolean	false	
terminateFDKs	object		Specifies a mask of those FDKs which must terminate the execution of the command
terminateFDKs.fdk01	boolean	false	
terminateFDKs.fdk02	boolean	false	
terminateFDKs.fdk03	boolean	false	
terminateFDKs.fdk04	boolean	false	
terminateFDKs.fdk05	boolean	false	
terminateFDKs.fdk06	boolean	false	
terminateFDKs.fdk07	boolean	false	
terminateFDKs.fdk08	boolean	false	
terminateFDKs.fdk09	boolean	false	
terminateFDKs.fdk10	boolean	false	
terminateFDKs.fdk11	boolean	false	
terminateFDKs.fdk12	boolean	false	
terminateFDKs.fdk13	boolean	false	
terminateFDKs.fdk14	boolean	false	
terminateFDKs.fdk15	boolean	false	
terminateFDKs.fdk16	boolean	false	
terminateFDKs.fdk17	boolean	false	
terminateFDKs.fdk18	boolean	false	
terminateFDKs.fdk19	boolean	false	
terminateFDKs.fdk20	boolean	false	
terminateFDKs.fdk21	boolean	false	
terminateFDKs.fdk22	boolean	false	

Name	Type	Default	Description
terminateFDKs.fdk23	boolean	false	
terminateFDKs.fdk24	boolean	false	
terminateFDKs.fdk25	boolean	false	
terminateFDKs.fdk26	boolean	false	
terminateFDKs.fdk27	boolean	false	
terminateFDKs.fdk28	boolean	false	
terminateFDKs.fdk29	boolean	false	
terminateFDKs.fdk30	boolean	false	
terminateFDKs.fdk31	boolean	false	
terminateFDKs.fdk32	boolean	false	
terminateKeys	object		Specifies a mask of those (other) Function Keys which must terminate the execution of the command
terminateKeys.fk0	boolean	false	
terminateKeys.fk1	boolean	false	
terminateKeys.fk2	boolean	false	
terminateKeys.fk3	boolean	false	
terminateKeys.fk4	boolean	false	
terminateKeys.fk5	boolean	false	
terminateKeys.fk6	boolean	false	
terminateKeys.fk7	boolean	false	
terminateKeys.fk8	boolean	false	
terminateKeys.fk9	boolean	false	
terminateKeys.fkA	boolean	false	
terminateKeys.fkB	boolean	false	
terminateKeys.fkC	boolean	false	
terminateKeys.fkD	boolean	false	
terminateKeys.fkE	boolean	false	
terminateKeys.fkF	boolean	false	
terminateKeys.fkEnter	boolean	false	
terminateKeys.fkCancel	boolean	false	
terminateKeys.fkClear	boolean	false	
terminateKeys.fkBackspace	boolean	false	
terminateKeys.fkHelp	boolean	false	
terminateKeys.fkDecPoint	boolean	false	
terminateKeys.fk00	boolean	false	
terminateKeys.fk000	boolean	false	
terminateKeys.fkShift	boolean	false	
terminateKeys.fkRES01	boolean	false	
terminateKeys.fkRES02	boolean	false	
terminateKeys.fkRES03	boolean	false	
terminateKeys.fkRES04	boolean	false	
terminateKeys.fkRES05	boolean	false	
terminateKeys.fkRES06	boolean	false	
terminateKeys.fkRES07	boolean	false	
terminateKeys.fkRES08	boolean	false	
terminateKeys.fkOEM01	boolean	false	
terminateKeys.fkOEM02	boolean	false	
terminateKeys.fkOEM03	boolean	false	
terminateKeys.fkOEM04	boolean	false	
terminateKeys.fkOEM05	boolean	false	
terminateKeys.fkOEM06	boolean	false	

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "maxLen": 0,
    "autoEnd": true,
    "activeFDKs": {
      "fdk01": false,
      "fdk02": false,
      "fdk03": false,
      "fdk04": false,
      "fdk05": false,
      "fdk06": false,
      "fdk07": false,
      "fdk08": false,
      "fdk09": false,
      "fdk10": false,
      "fdk11": false,
      "fdk12": false,
      "fdk13": false,
      "fdk14": false,
      "fdk15": false,
      "fdk16": false,
      "fdk17": false,
```

```
      "fdk18": false,
      "fdk19": false,
      "fdk20": false,
      "fdk21": false,
      "fdk22": false,
      "fdk23": false,
      "fdk24": false,
      "fdk25": false,
      "fdk26": false,
      "fdk27": false,
      "fdk28": false,
      "fdk29": false,
      "fdk30": false,
      "fdk31": false,
      "fdk32": false
    },
    "activeKeys": {
      "fk0": false,
      "fk1": false,
      "fk2": false,
      "fk3": false,
      "fk4": false,
      "fk5": false,
      "fk6": false,
      "fk7": false,
      "fk8": false,
      "fk9": false,
      "fkA": false,
      "fkB": false,
      "fkC": false,
      "fkD": false,
      "fkE": false,
      "fkF": false,
      "fkEnter": false,
      "fkCancel": false,
      "fkClear": false,
      "fkBackspace": false,
      "fkHelp": false,
      "fkDecPoint": false,
      "fk00": false,
      "fk000": false,
      "fkShift": false,
      "fkRES01": false,
      "fkRES02": false,
      "fkRES03": false,
      "fkRES04": false,
      "fkRES05": false,
      "fkRES06": false,
      "fkRES07": false,
      "fkRES08": false,
      "fkOEM01": false,
      "fkOEM02": false,
      "fkOEM03": false,
      "fkOEM04": false,
      "fkOEM05": false,
      "fkOEM06": false
    },
    "terminateFDKs": {
      "fdk01": false,
      "fdk02": false,
      "fdk03": false,
      "fdk04": false,
      "fdk05": false,
      "fdk06": false,
      "fdk07": false,
      "fdk08": false,
      "fdk09": false,
      "fdk10": false,
      "fdk11": false,
      "fdk12": false,
      "fdk13": false,
      "fdk14": false,
      "fdk15": false,
      "fdk16": false,
      "fdk17": false,
      "fdk18": false,
      "fdk19": false,
      "fdk20": false,
      "fdk21": false,
      "fdk22": false,
      "fdk23": false,
      "fdk24": false,
      "fdk25": false,
      "fdk26": false,
      "fdk27": false,
      "fdk28": false,
      "fdk29": false,
      "fdk30": false,
      "fdk31": false,
```

```
      "fdk32": false
    },
    "terminateKeys": {
      "fk0": false,
      "fk1": false,
      "fk2": false,
      "fk3": false,
      "fk4": false,
      "fk5": false,
      "fk6": false,
      "fk7": false,
      "fk8": false,
      "fk9": false,
      "fkA": false,
      "fkB": false,
      "fkC": false,
      "fkD": false,
      "fkE": false,
      "fkF": false,
      "fkEnter": false,
      "fkCancel": false,
      "fkClear": false,
      "fkBackspace": false,
      "fkHelp": false,
      "fkDecPoint": false,
      "fk00": false,
      "fk000": false,
      "fkShift": false,
      "fkRES01": false,
      "fkRES02": false,
      "fkRES03": false,
      "fkRES04": false,
      "fkRES05": false,
      "fkRES06": false,
      "fkRES07": false,
      "fkRES08": false,
      "fkOEM01": false,
      "fkOEM02": false,
      "fkOEM03": false,
      "fkOEM04": false,
      "fkOEM05": false,
      "fkOEM06": false
    }
  }
}
```


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
keys	integer		Number of keys entered by the user
pinKeys	array		Array to the pinKey that contain the keys entered by the user
completion	string		Specifies the reason for completion of the entry

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "keys": 0,
    "pinKeys": [
      "auto"
    ],
    "completion": "auto"
  }
}
```


Event Messages

- Pinpad.KeyEvent
- Pinpad.EnterDataEvent

Pinpad.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.
ident	string		The value of the ID key. An empty string if not required.
key	string		The value of the encryption key formatted in base64. An empty string if not 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, but can be NULL 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

Pinpad.LocalDES

Description

The PIN, which was entered with the GetPin command, is combined with the requisite data specified by the DES validation algorithm and locally verified for correctness. The result of the verification is returned to the application. This command will clear the PIN unless the application has requested that the PIN be maintained through the MaintainPin 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

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.
validationData	string		Customer specific data (normally obtained from card track data) used to validate the correctness of the PIN. The validation data should be an ASCII string.
offset	string		ASCII string defining the offset data for the PIN block as an ASCII string; if NULL then no offset is used. The character must be in the ranges '0' to '9', 'a' to 'f' and 'A' to 'F'.
padding	integer		Specifies the padding character for the validation data. If the validation data is less than 16 characters long then it will be padded with this character. If padding is in the range 0x00 to 0x0F, padding is applied after the validation data has been compressed. If the padding character is in the range '0' to '9', 'a' to 'f', or 'A' to 'F', padding is applied before the validation data is compressed.
maxPIN	integer		Maximum number of PIN digits to be used for validation. This parameter corresponds to PINMINL in the IBM 3624 specification
valDigits	integer		Number of Validation digits from the validation data to be used for validation. This is the length of the validationData string.
noLeadingZero	boolean		If set to TRUE and the first digit of result of the modulo 10 addition is a 0x0, it is replaced with 0x1 before performing the verification against the entered PIN. If set to FALSE, a leading zero is allowed in entered PINs
key	string		Name of the key to be used for validation. The key referenced by key must have the function or pinLocal attribute.
keyEncKey	string		If an empty string, key is used directly for PIN validation. Otherwise, key is used to decrypt the encrypted key passed in keyEncKey and the result is used for PIN validation.
decTable	string		ASCII decimalization table (16 character string containing characters '0' to '9'). This table is used to convert the hexadecimal digits (0x0 to 0xF) of the encrypted validation data to decimal digits (0x0 to 0x9).

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "validationData": "string",
    "offset": "string",
    "padding": 0,
    "maxPIN": 0,
    "valDigits": 0,
    "noLeadingZero": true,
    "key": "string",
    "keyEncKey": "string",
    "decTable": "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
result	boolean		boolean value which specifies whether the PIN is correct or not.

Example Message (generated)

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

Event Messages

Pinpad.LocalEuroCheque

Description

The PIN, which was entered with the GetPin command, is combined with the requisite data specified by the Eurocheque validation algorithm and locally verified for correctness. The result of the verification is returned to the application. This command will clear the PIN unless the application has requested that the PIN be maintained through the MaintainPin 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.
euroChequeData	string		Track-3 Eurocheque data
pvv	string		PIN Validation Value from track data.
firstEncDigits	integer		Number of digits to extract after first encryption.
firstEncOffset	integer		Offset of digits to extract after first encryption.
pvvDigits	integer		Number of digits to extract for pvv.
pvvOffset	integer		Offset of digits to extract for pvv.
key	string		Name of the validation key. The key referenced by key must have the function or pinLocal attribute
keyEncKey	string		If an empty string, key is used directly for PIN validation. Otherwise, key is used to decrypt the encrypted key passed in keyEncKey and the result is used for PIN validation.
decTable	string		ASCII decimalization table (16 character string containing characters '0' to '9'). This table is used to convert the hexadecimal digits (0x0 to 0xF) of the encrypted validation data to decimal digits (0x0 to 0x9).

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "euroChequeData": "string",
    "pvv": "string",
    "firstEncDigits": 0,
    "firstEncOffset": 0,
    "pvvDigits": 0,
    "pvvOffset": 0,
    "key": "string",
    "keyEncKey": "string",
    "decTable": "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
result	boolean		Boolean value which specifies whether the PIN is correct or not.

Example Message (generated)


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


Event Messages

Pinpad.LocalVisa

Description

The PIN, which was entered with the GetPin command, is combined with the requisite data specified by the VISA validation algorithm and locally verified for correctness. The result of the verification is returned to the application. This command will clear the PIN unless the application has requested that the PIN be maintained through the MaintainPin 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	Type	Default	Description
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.
pan	string		Primary Account Number from track data, as an ASCII string. PAN should contain the eleven rightmost digits of the PAN (excluding the check digit), followed by the pvki indicator in the 12th byte.
pvv	string		PIN Validation Value from track data, as an ASCII string with characters in the range '0' to '9'. This string should contain 4 digits.
pvvDigits	integer		Number of digits of PVV.
key	string		Name of the validation key. The key referenced by key must have the function or pinLocal attribute
keyEncKey	string		If an empty string, key is used directly for PIN validation. Otherwise, key is used to decrypt the encrypted key passed in keyEncKey and the result is used for PIN validation.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "pan": "string",
    "pvv": "string",
    "pvvDigits": 0,
    "key": "string",
    "keyEncKey": "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
result	boolean		Pointer to a boolean value which specifies whether the PIN is correct or not.

Example Message (generated)


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


Event Messages

Pinpad.CreateOffset

Description

This function is used to generate a pin Offset that is typically written to a card and later used to verify the PIN with the LocalDes command. The PIN offset is computed by combining validation data with the keypad entered PIN. This command will clear the PIN unless the application has requested that the PIN be maintained through the MaintainPin 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.
validationData	string		Validation data. The validation data should be an ASCII string.
padding	integer		Specifies the padding character for validation data. If padding is in the range 0x00 to 0x0F, padding is applied after the validation data has been compressed. If the padding character is in the range '0' to '9', 'a' to 'f', or 'A' to 'F, padding is applied before the validation data is compressed.
maxPin	integer		Maximum number of pin digits to be used for PIN Offset creation. This parameter corresponds to pinMinl in the IBM 3624 specification.
valDigits	integer		Number of validation Data digits to be used for PIN Offset creation. This is the length of the validationData string.
key	string		Name of the validation key. The key referenced by key must have the function or pinLocal attribute.
keyEncKey	string		If an empty string, key is used directly in PIN Offset creation. Otherwise, key is used to decrypt the encrypted key passed in keyEncKey and the result is used in PIN Offset creation.
decTable	string		ASCII decimalization table (16 character string containing characters '0' to '9'). This table is used to convert the hexadecimal digits (0x0 to 0xF) of the encrypted validation data to decimal digits (0x0 to 0x9).

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "validationData": "string",
    "padding": 0,
    "maxPin": 0,
    "valDigits": 0,
    "key": "string",
    "keyEncKey": "string",
    "decTable": "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
offset	string		Computed pin Offset.

Example Message (generated)


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


Event Messages

Pinpad.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
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	integer		Specifies the algorithm that is used for derivation.
key	string		Specifies the name where the derived key will be stored.
keyGenKey	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 parameter is NULL, startValue is used as the Initialization Vector.
startValue	string		des initialization vector for the encryption step within the derivation.
padding	integer		Specifies the padding character for the encryption step within the derivation. The valid range is 0x00 to 0xFF
inputData	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 ident should be NULL. 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

Pinpad.PresentIDC

Description

The PIN, which was entered with the GetPin command, is combined with the requisite data specified by the IDC presentation algorithm and presented to the smartcard contained in the ID card unit. The result of the presentation is returned to the application. This command will clear the PIN unless the application has requested that the PIN be maintained through the MaintainPin 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

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.
presentAlgorithm	string		Specifies the algorithm that is used for presentation. Possible values are: (see command Capabilites).
chipProtocol	string		Identifies the protocol that is used to communicate with the chip. Possible values are: (see command CardReader.Capabilites in the Identification Card Device Class Interface)
chipData	string		Points to the data to be sent to the chip formatted in base64.
algorithmData	object		Contains the data required for the specified presentation algorithm
algorithmData.pinPointer	integer		The byte offset where to start inserting the PIN into chipData. The leftmost byte is numbered zero. See below for an example
algorithmData.pinOffset	integer		The bit offset within the byte specified by pinPointer where to start inserting the PIN. The leftmost bit numbered zero.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "presentAlgorithm": "presentClear",
    "chipProtocol": "string",
    "chipData": "string",
    "algorithmData": {
      "pinPointer": 0,
      "pinOffset": 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
chipProtocol	string		Identifies the protocol that was used to communicate with the chip. This field contains the same value as the corresponding field in the input.
chipData	string		The data responded from the chip.

Example Message (generated)


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


Event Messages

Pinpad.LocalBanksys

Description

The PIN block previously built by the GetPin command is sent to the Banksys security control module using the BankSysIO command. The BANKSYS security control module will return an atmVac code, which is then used in this command to locally validate the PIN. The key referenced by key within the most recent successful GetPinBlock command is reused by the LocalBankSysIO command for the local validation.

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.
atmVac	string		The atmVac code calculated by the Banksys Security Control Module formatted in base64

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "atmVac": "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
result	boolean		Pointer to a boolean value which specifies whether the PIN is correct or not

Example Message (generated)

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

Event Messages

Pinpad.BanksysIo

Description

This command sends a single command to the Banksys Security Control Module.

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.
data	string		The data sent to the Banksys Security Control Module formatted in base64.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "data": "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
data	string		The data responded by the Banksys Security Control Module formatted in base64.

Example Message (generated)


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


Event Messages

Pinpad.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

Pinpad.HSMSetTData

Description

This function allows the application to set the hsm terminal data (except keys, trace number and session key index). The data must be provided as a series of â€œtag/length/valueâ€? items. Terminal data that are set but are not supported by the hardware will be ignored.

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.
tData	string		Specifies which parameter(s) is(are) to be set. Specifies which parameter(s) is(are) to be set. tData is a series of â€œtag/length/valueâ€? items where each item consists of: * One byte tag (see the list of tags below). * One byte specifying the length of the following data as an unsigned binary number. *N bytes data (see the list below for formatting) with no separators. The following tags are supported: Tag Format Length Meaning Read / EPP / (hexl) (bytes) Write HSM C2 BCD 4 Terminal ID R/W EPP ISO BMP 41 C3 BCD 4 Bank code R/W EPP ISO BMP 42 (rightmost 4 bytes) C4 BCD 9 Account data for terminal account R/W EPP ISO BMP 60 (load against other card) C5 BCD 9 Account data for fee account R/W EPP ISO BMP 60 (Laden vom Kartenkonto) C6 EBCDIC 40 Terminal location R/W EPP ISO BMP 43 C7 ASCII 3 Terminal currency R/W EPP C8 BCD 7 Online date and time R/W HSM (YYYYMMDDHHMMSS) ISO BMP 61 C9 BCD 4 Minimum load fee in units of 1/100 of R/W EPP terminal currency, checked against leftmost 4 Bytes of ISO BMP42 CA BCD 4 Maximum load fee in units of 1/100 of R/W EPP terminal currency, checked against leftmost 4 Bytes of ISO BMP42 CB BIN 3 logical HSM binary coded serial R HSM number (starts with 1; 0 means that there are no logical HSMs) CC EBCDIC 16 ZKA ID (is filled during the pre- R HSM initialization of the HSM) CD BIN 1 HSM status R HSM 1 = irreversibly out of order 2 = out of order, K_UR is not loaded 3 = not pre-initialized, K_UR is loaded 4 = pre-initialized, K_INIT is loaded 5 = initialized/personalized, K_PERS is loaded CE EBCDIC variable, HSM-ID (6 byte Manufacturer- ID + R EPP min. 16 min. 10 Byte serial number), as needed for ISO BMP57 of a pre-initialization In the table above, the fifth column indicates if the variable is read only or both read and write. The sixth column indicates if the variable is unique per logical HSM or common across all logical HSMs within an EPP.

Example Message (generated)

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

Pinpad.SecureMsgSend

Description

This function allows the application to set the HSM terminal data (except keys, trace number and session key index). The data must be provided as a series of 'tag/length/value' items. Terminal data that are set but are not supported by the hardware will be ignored.

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.
protocol	string		Specifies the protocol the message belongs to.
msg	string		Specifies the message that was received. This value can be NULL if during a specified time period no response was received from the communication partner (necessary to set the internal state machine to the correct state).

Example Message (generated)

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

Completion 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
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error
protocol	string		Specifies the protocol the message belongs to.
msg	string		Specifies the message that was received. This value can be NULL if during a specified time period no response was received from the communication partner (necessary to set the internal state machine to the correct state).

Example Message (generated)


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


Event Messages

Pinpad.SecureMsgReceive

Description

This command handles all messages that are received through a secure messaging from an authorization system, German 'Ladezentrale', personalization system or the chip. The encryption module checks the security relevant fields. All messages must be presented to the encryptor via this command even if they do not contain security relevant fields in order to keep track of the transaction status in the internal state machine.

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.
protocol	string		Specifies the protocol the message belongs to.
msg	string		Specifies the message that was received. This value can be NULL if during a specified time period no response was received from the communication partner (necessary to set the internal state machine to the correct state).

Example Message (generated)

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

Pinpad.GetJournal

Description

This command is used to get journal data from the encryption module. It retrieves cryptographically secured information about the result of the last transaction that was done with the indicated protocol. When the Service Provider supports journaling (see Capabilities) then it is impossible to do any SecureMsgSend/Receive with this protocol, unless the journal data is retrieved. It is possible - especially after restarting a system - to get the same journal data again.

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.
protocol	string		Specifies the protocol the journal data belong to.

Example Message (generated)


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


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
journalData	string		The journal data formatted in base64.

Example Message (generated)


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


Event Messages

Pinpad.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

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 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. Must be NULL if the key specified by Key is to be deleted.
keyAttributes.keyUsage	string		Specifies the Key usages supported by Import Key command.
keyAttributes.propkeyUsage	integer	0	if keyUsage is set to Numeric, proprietary value is set.
keyAttributes.algorithm	string		Specifies the encryption algorithms supported by the import command.
keyAttributes.propAlgorithm	integer	0	if algorithm is set to Numeric, proprietary value is set.
keyAttributes.modeOfUse	string		Specifies the encryption modes supported by import key.
keyAttributes.propmodeOfUse	integer	0	if modeOfUse is set to Numeric, proprietary value is set.
keyAttributes.cryptoMethod	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	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. Can be NUan empty string 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 PIN 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. Must be an empty string 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.
verificationData	string		Contains the data to be verified before importing. verificationData is an empty string 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. verifyKey is an empty string 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. Must be Null if verificationData is an empty string.
verifyAttributes.keyUsage	string		Specifies the key usages supported by the import command.
verifyAttributes.propkeyUsage	integer	0	if keyUsage is set to Numeric, proprietary value is set.
verifyAttributes.algorithm	string		Specifies the encryption algorithms supported by the import command.
verifyAttributes.propAlgorithm	integer	0	if algorithm is set to Numeric, proprietary value is set.
verifyAttributes.modeOfUse	string		Specifies the encryption modes supported by the import command.
verifyAttributes.propmodeOfUse	integer	0	if modeOfUse is set to Numeric, proprietary value is set.
verifyAttributes.cryptoMethod	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 parameter must be Null.

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 parameter is an empty 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 parameter is NULL if there is no verification data.
verifyAttributes.keyUsage	string		Specifies the key usages supported by the import command.
verifyAttributes.propkeyUsage	integer	0	if keyUsage is set to Numeric, proprietary value is set.
verifyAttributes.algorithm	string		Specifies the encryption algorithms supported by the import command.
verifyAttributes.propAlgorithm	integer	0	if algorithm is set to Numeric, proprietary value is set.
verifyAttributes.modeOfUse	string		Specifies the encryption modes supported by the import command.
verifyAttributes.propmodeOfUse	integer	0	if modeOfUse is set to Numeric, proprietary value is set.
verifyAttributes.cryptoMethod	string		This parameter specifies the cryptographic method that will be used with encryption algorithm.
keyLength	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

Pinpad.Enclo

Description

This command is used to communicate with the encryption module. Transparent data is sent from the application to the encryption module and the response is returned transparently to the application. This command is used to add support for country-specific protocols.

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.
protocol	string		Identifies the protocol that is used to communicate with the encryption module.
ioData	string		A structure containing the data to be sent to the encryption module formatted in base64. This structure depends on the protocol field where each protocol may contain a different structure

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "protocol": "ch",
    "ioData": "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	Type	Default	Description
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
protocol	string		Identifies the protocol that is used to communicate with the encryption module. This field contains the same value as the corresponding field in the input structure.
ioData	string		A structure containing the data responded by the encryption module formatted in base64.

Example Message (generated)


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


Event Messages

Pinpad.HSMInit

Description

This command is used to set the hsm out of order. If multiple logical hsms are configured then the command sets the currently active logical hsm out of order. At the same time the online time can be set to control when the opt online dialog (see Wisops protocol) shall be started to initialize the hsm again. When this time is reached an optRequiredEvent will be sent.

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.
initMode	string		Specifies the init mode
onlineTime	string		Specifies the Online date and time in the format YYYYMMDDHHMMSS like in ISO BMP 61 as BCD packed characters. This parameter is ignored when the init mode equals definite or irreversiblen. If this parameter is an empty or the value is 0x00 0x00 0x00 0x00 0x00 0x00 0x00 the online time will be set to a value in the past.

Example Message (generated)

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

Pinpad.ImportRSAPublicKey

Description

The Public rsa key passed by the application is loaded in the encryption module. The use parameter restricts the cryptographic functions that the imported key can be used for. This command provides similar public key import functionality to that provided with ImportKey. The primary advantage gained through using this function is that the imported key can be verified as having come from a trusted source. If a Signature algorithm is specified that is not supported by the PIN Service Provider, then the request will not be accepted and the command fails.

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
key	string		Specifies the name of key being loaded
value	string		Contains the pkcs #1 formatted rsa Public Key to be loaded formatted in base64, represented in DER encoded ASN.1.
use	string		Specifies the type of access for which the key can be used. If this parameter equals zero, the key is deleted.
sigKey	string		Specifies the name of a previously loaded asymmetric key (i.e. an rsa Public Key) which will be used to verify the signature passed in signature. The default Signature Issuer public key (installed in a secure environment during manufacture) will be used, if sigKey is either an empty string or contains the name of the default Signature issuer
rsaSignatureAlgorithm	string		Defines the algorithm used to generate the Signature specified in signature.
signature	string		Base64 encoded signature data.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "key": "string",
    "value": "string",
    "use": "rsaPublic",
    "sigKey": "string",
    "rsaSignatureAlgorithm": "na",
    "signature": "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	string		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.
keyCheckValue	string		Contains the Signature associated with the key being imported or deleted formatted in base64. The signature is used to validate the key request has been received from a trusted sender. This value is an empty when no key validation is required.

Example Message (generated)


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


Event Messages

Pinpad.ExportRSAIssuerSignedItem

Description

This command is used to export data elements from the PIN 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 PIN device. This value may be used to uniquely identify a PIN 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 PIN 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
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 PIN pad.
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¿½s Security Item, which may be vendor specific.
rsaSignatureAlgorithm	string		Specifies the algorithm used to generate the Signature returned in signature

Name	Type	Default	Description
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

Pinpad.ImportRSASignedDESKey

Description

This command is used to load a Symmetric Key that is either a single-length, double-length or triple-length DES key into the Pinpad. The key passed by the application is loaded in the encryption module, the (optional) signature is used during validation, the key is decrypted using the deviceâ€™s rsa Private Key, and is then stored. The loaded key will be discarded at any stage if any of the above fails. The random number previously obtained from the StartKeyExchange command and sent to the host is included in the signed data. This random number (when present) is verified during the load process. This command ends the Key Exchange process. The use parameter restricts the cryptographic functions that the imported key can be used for. If a signature algorithm is specified that is not supported by the pin Service Provider, then the message will not be decrypted and the command fails.

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 key being loaded.
decryptKey	string		Specifies the name of the rsa private key used to decrypt the symmetric key. See section 8.1.8 (Default Keys and Security Item loaded during manufacture) for a description of the fixed name defined for the default decryption private key. If decryptKey is an empty then the default decryption private key is used.
rsaEncipherAlgorithm	string		Specifies the RSA algorithm that is used, along with the private key, to decipher the imported key
value	string		Specifies the enciphered value of the key to be loaded formatted in base64. value contains the concatenation of the random number (when present) and enciphered key
use	object		Specifies the type of access for which the key can be used.
use.crypt	boolean		Key isused for encryption and decryption.
use.function	boolean		Key is used for pin block creation.
use.macing	boolean		Key is using for macing.
use.keyEncKey	boolean		Key is used as key encryption key
use.pinLocal	boolean		Key is used only for local PIN check.

Name	Type	Default	Description
use.noDuplicate	boolean		Key can be imported only once
use.svEncKey	boolean		Key is used as cbc start Value encryption key.
use.ansTR31master	boolean		Key can be used for importing keys packaged within an ANS TR-31 key block. This key usage can only be combined with construct and secureConstruct
use.pinRemote	boolean		Key is used only for PIN block creation.
use.restrictedKeyEncKey	boolean		Key is used as keyenckey key whose later subsequently derived keys inherit and are restricted to a single use. To express this the restrictedKeyEncKey use must be combined with the use keyEncKey and must additionally be combined with the use that the later subsequently derived keys will have.
use.delete	boolean	false	Delete key and other usage must not be used.
sigKey	string		If sigKey is an empty string then the key signature will not be used for validation and signature is ignored. Otherwise sigKey specifies the name of an Asymmetric Key (i.e. an rsa Public Key) previously loaded which will be used to verify the signature passed in signature.
rsaSignatureAlgorithm	string		Specifies the algorithm used to generate the signature specified in signature
signature	string		Contains the Signature associated with the key being imported formatted in base64. The signature is used to validate the key has been received from a trusted sender. The signature is generated over the contents of the value. The signature signature contains NULL when no key validation is required.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "key": "string",
    "decryptKey": "string",
    "rsaEncipherAlgorithm": "pkcs1V15",
    "value": "string",
    "use": {
      "crypt": true,
      "function": true,
      "macing": true,
      "keyEncKey": true,
      "pinLocal": true,
      "noDuplicate": true,
      "svEncKey": true,
      "ansTR31master": true,
      "pinRemote": true,
      "restrictedKeyEncKey": true,
      "delete": false
    },
    "sigKey": "string",
    "rsaSignatureAlgorithm": "na",
    "signature": "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

Name	Type	Default	Description
keyLength	string		Specifies the length of the key loaded
keyCheckMode	string		Specifies the mode that is used to create the key check value
keyCheckValue	string		The key verification data that can be used for verification of the loaded key formatted in base64, An empty string if device does not have that capability

Example Message (generated)


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


Event Messages

Pinpad.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 pin 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

Pinpad.ExportRSAEPPSignedItem

Description

This command is used to export data elements from the pin 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 pin 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 pin device. This value may be used to uniquely identify a PIN 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 pin 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

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 pin pad
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.
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

Pinpad.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	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	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

Pinpad.ReplaceCertificate

Description

This command is used to replace the existing primary or secondary Certificate Authority certificate already loaded into the Pinpad. 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 Pinpad. 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
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	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

Name	Type	Default	Description
errorDescription	string		If error, identified that cause of the error
newCertificateData	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

Pinpad.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.
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

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 PIN 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

Pinpad.EMVImportPublicKey

Description

The Certification Authority and the chip card rsa public keys needed for EMV are loaded or deleted in/from the encryption module. This command is similar to the ImportKey command, but it is specifically designed to address the key formats and security features defined by emv. Mainly the extensive use of â€œsigned certificateâ€? or â€œemv certificateâ€? (which is a compromise between signature and a pure certificate) to provide the public key is taken in account. The Service Provider is responsible for all EMV public key import validation. Once loaded, the Service Provider is not responsible for key/certificate expiry, this is an application responsibility.

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 key being loaded.
use	string		Specifies the type of access for which the key can be used. If this parameter equals zero, the key is deleted.
importScheme	string		Defines the import scheme used.
importData	string		The importData parameter contains all the necessary data to complete the import using the scheme specified within importScheme.
sigKey	string		This field specifies the name of the previously loaded key used to verify the signature.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "key": "string",
    "use": "public",
    "importScheme": "plainCA",
    "importData": "string",
    "sigKey": "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
expiryDate	string		Contains the expiry date of the certificate in the following format MMYY. If no expiry date applies then expiryDate is an empty string.

Example Message (generated)


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


Event Messages

Pinpad.Digest

Description

This command is used to compute a hash code on a stream of data using the specified hash algorithm. This command can be used to verify emv static and dynamic data.

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.
hashAlgorithm	string		Specifies which hash algorithm should be used to calculate the hash. See the Capabilities section for valid algorithms.
digestInput	string		Contains the length and the data to be hashed formatted in base64.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "hashAlgorithm": "sha1",
    "digestInput": "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
digestOutput	string		Contains the length and the data containing the calculated has.

Example Message (generated)


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


Event Messages

Pinpad.SecureKeyEntry

Description

This command allows a full length symmetric encryption key part to be entered directly into the PIN pad without being exposed outside of the PIN pad. From the point this function is invoked, encryption key digits (fk0 to fk9 and fkA to fkF) are not passed to the application. For each encryption key digit, or any other active key entered (except for shift), an execute notification event eyEvent is sent in order to allow an application to perform the appropriate display action (i.e. when the PIN pad has no integrated display). When an encryption key digit is entered the application is not informed of the value entered, instead zero is returned. The EnterDataEvent will be generated when the PIN pad is ready for the user to start entering data. The keys that can be enabled by this command are defined by the FuncKeyDetail parameter of the SecureKeyDetail command. Function keys which are not associated with an encryption key digit may be enabled but will not contribute to the secure entry buffer (unless they are Cancel, Clear or Backspace) and will not count towards the length of the key entry. The Cancel and Clear keys will cause the encryption key buffer to be cleared. The Backspace key will cause the last encryption key digit in the encryption key buffer to be removed. If autoEnd is TRUE the command will automatically complete when the required number of encryption key digits have been added to the buffer. If autoEnd is FALSE then the command will not automatically complete and Enter, Cancel or any terminating key must be pressed. When keyLen hex encryption key digits have been entered then all encryption key digits keys are disabled. If the Clear or Backspace key is pressed to reduce the number of entered encryption key digits below usKeyLen, the same keys will be reenabled. Terminating keys have to be active keys to operate. If an FDK is associated with

Enter, Cancel, Clear or Backspace then the FDK must be activated to operate. The Enter and Cancel FDKs must also be marked as a terminator if they are to terminate entry. These FDKs are reported as normal FDKs within the KeyEvent, applications must be aware of those FDKs associated with Cancel, Clear, Backspace and Enter and handle any user interaction as required. For example, if the fdk01 is associated with Clear, then the application must include the fk_fdk01 FDK code in the activeFDKs parameter (if the clear functionality is required). In addition when this FDK is pressed the KeyEvent will contain the fk_fdk01 mask value in the digit field. The application must update the user interface to reflect the effect of the clear on the encryption key digits entered so far. On some devices that are configured as either regularUnique or irregularUnique all the function keys on the PIN pad will be associated with hex digits and there may be no FDKs available either. On these devices there may be no way to correct mistakes or cancel the key encryption entry before all the encryption key digits are entered, so the application must set the autoEnd flag to TRUE and wait for the command to auto-complete. Applications should check the KCV to avoid storing an incorrect key component. Encryption key parts entered with this command are stored through either the ImportKey. Each key part can only be stored once after which the secure key buffer will be cleared automatically.

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.
keyLen	integer		Specifies the number of digits which must be entered for the encryption key, 16 for a singlelength key, 32 for a double-length key and 48 for a triple-length key. The only valid values are 16, 32 and 48.
autoEnd	boolean	false	If autoEnd is set to true, the Service Provider terminates the command when the maximum number of encryption key digits are entered. Otherwise, the input is terminated by the user using Enter, Cancel or any terminating key. When keyLen is reached, the Service Provider will disable all keys associated with an encryption key digit.
activeFDKs	object		Specifies those FDKs which are active during the execution of the command. This parameter should include those FDKs mapped to edit functions
activeFDKs.fdk01	boolean	false	
activeFDKs.fdk02	boolean	false	
activeFDKs.fdk03	boolean	false	
activeFDKs.fdk04	boolean	false	
activeFDKs.fdk05	boolean	false	
activeFDKs.fdk06	boolean	false	
activeFDKs.fdk07	boolean	false	
activeFDKs.fdk08	boolean	false	
activeFDKs.fdk09	boolean	false	
activeFDKs.fdk10	boolean	false	
activeFDKs.fdk11	boolean	false	
activeFDKs.fdk12	boolean	false	
activeFDKs.fdk13	boolean	false	
activeFDKs.fdk14	boolean	false	
activeFDKs.fdk15	boolean	false	
activeFDKs.fdk16	boolean	false	
activeFDKs.fdk17	boolean	false	
activeFDKs.fdk18	boolean	false	
activeFDKs.fdk19	boolean	false	
activeFDKs.fdk20	boolean	false	
activeFDKs.fdk21	boolean	false	
activeFDKs.fdk22	boolean	false	
activeFDKs.fdk23	boolean	false	
activeFDKs.fdk24	boolean	false	
activeFDKs.fdk25	boolean	false	
activeFDKs.fdk26	boolean	false	
activeFDKs.fdk27	boolean	false	
activeFDKs.fdk28	boolean	false	
activeFDKs.fdk29	boolean	false	
activeFDKs.fdk30	boolean	false	
activeFDKs.fdk31	boolean	false	
activeFDKs.fdk32	boolean	false	
activeKeys	object		Specifies all Function Keys(not FDKs) which are active during the execution of the command. This should be the complete set or a subset of the keys returned in the FuncKeyDetail parameter of the SecureKeyDetail command.
activeKeys.fk0	boolean	false	
activeKeys.fk1	boolean	false	
activeKeys.fk2	boolean	false	
activeKeys.fk3	boolean	false	
activeKeys.fk4	boolean	false	
activeKeys.fk5	boolean	false	
activeKeys.fk6	boolean	false	
activeKeys.fk7	boolean	false	

Name	Type	Default	Description
activeKeys.fk8	boolean	false	
activeKeys.fk9	boolean	false	
activeKeys.fkA	boolean	false	
activeKeys.fkB	boolean	false	
activeKeys.fkC	boolean	false	
activeKeys.fkD	boolean	false	
activeKeys.fkE	boolean	false	
activeKeys.fkF	boolean	false	
activeKeys.fkEnter	boolean	false	
activeKeys.fkCancel	boolean	false	
activeKeys.fkClear	boolean	false	
activeKeys.fkBackspace	boolean	false	
activeKeys.fkHelp	boolean	false	
activeKeys.fkDecPoint	boolean	false	
activeKeys.fk00	boolean	false	
activeKeys.fk000	boolean	false	
activeKeys.fkShift	boolean	false	
activeKeys.fkRES01	boolean	false	
activeKeys.fkRES02	boolean	false	
activeKeys.fkRES03	boolean	false	
activeKeys.fkRES04	boolean	false	
activeKeys.fkRES05	boolean	false	
activeKeys.fkRES06	boolean	false	
activeKeys.fkRES07	boolean	false	
activeKeys.fkRES08	boolean	false	
activeKeys.fkOEM01	boolean	false	
activeKeys.fkOEM02	boolean	false	
activeKeys.fkOEM03	boolean	false	
activeKeys.fkOEM04	boolean	false	
activeKeys.fkOEM05	boolean	false	
activeKeys.fkOEM06	boolean	false	
terminateFDKs	object		Specifies those FDKs which must terminate the execution of the command. This should include the FDKs associated with Cancel and Enter.
terminateFDKs.fdk01	boolean	false	
terminateFDKs.fdk02	boolean	false	
terminateFDKs.fdk03	boolean	false	
terminateFDKs.fdk04	boolean	false	
terminateFDKs.fdk05	boolean	false	
terminateFDKs.fdk06	boolean	false	
terminateFDKs.fdk07	boolean	false	
terminateFDKs.fdk08	boolean	false	
terminateFDKs.fdk09	boolean	false	
terminateFDKs.fdk10	boolean	false	
terminateFDKs.fdk11	boolean	false	
terminateFDKs.fdk12	boolean	false	
terminateFDKs.fdk13	boolean	false	
terminateFDKs.fdk14	boolean	false	
terminateFDKs.fdk15	boolean	false	
terminateFDKs.fdk16	boolean	false	
terminateFDKs.fdk17	boolean	false	
terminateFDKs.fdk18	boolean	false	
terminateFDKs.fdk19	boolean	false	
terminateFDKs.fdk20	boolean	false	
terminateFDKs.fdk21	boolean	false	
terminateFDKs.fdk22	boolean	false	
terminateFDKs.fdk23	boolean	false	
terminateFDKs.fdk24	boolean	false	
terminateFDKs.fdk25	boolean	false	
terminateFDKs.fdk26	boolean	false	
terminateFDKs.fdk27	boolean	false	
terminateFDKs.fdk28	boolean	false	
terminateFDKs.fdk29	boolean	false	
terminateFDKs.fdk30	boolean	false	
terminateFDKs.fdk31	boolean	false	
terminateFDKs.fdk32	boolean	false	
trerminateKeys	object		Specifies those all Function Keys (not FDKs) which must terminate the execution of the command. This does not include the FDKs associated with Enter or Cancel.
trerminateKeys.fk0	boolean	false	
trerminateKeys.fk1	boolean	false	
trerminateKeys.fk2	boolean	false	
trerminateKeys.fk3	boolean	false	
trerminateKeys.fk4	boolean	false	
trerminateKeys.fk5	boolean	false	
trerminateKeys.fk6	boolean	false	
trerminateKeys.fk7	boolean	false	
trerminateKeys.fk8	boolean	false	
trerminateKeys.fk9	boolean	false	

Name	Type	Default	Description
trerminateKeys.fkA	boolean	false	
trerminateKeys.fkB	boolean	false	
trerminateKeys.fkC	boolean	false	
trerminateKeys.fkD	boolean	false	
trerminateKeys.fkE	boolean	false	
trerminateKeys.fkF	boolean	false	
trerminateKeys.fkEnter	boolean	false	
trerminateKeys.fkCancel	boolean	false	
trerminateKeys.fkClear	boolean	false	
trerminateKeys.fkBackspace	boolean	false	
trerminateKeys.fkHelp	boolean	false	
trerminateKeys.fkDecPoint	boolean	false	
trerminateKeys.fk00	boolean	false	
trerminateKeys.fk000	boolean	false	
trerminateKeys.fkShift	boolean	false	
trerminateKeys.fkRES01	boolean	false	
trerminateKeys.fkRES02	boolean	false	
trerminateKeys.fkRES03	boolean	false	
trerminateKeys.fkRES04	boolean	false	
trerminateKeys.fkRES05	boolean	false	
trerminateKeys.fkRES06	boolean	false	
trerminateKeys.fkRES07	boolean	false	
trerminateKeys.fkRES08	boolean	false	
trerminateKeys.fkOEM01	boolean	false	
trerminateKeys.fkOEM02	boolean	false	
trerminateKeys.fkOEM03	boolean	false	
trerminateKeys.fkOEM04	boolean	false	
trerminateKeys.fkOEM05	boolean	false	
trerminateKeys.fkOEM06	boolean	false	
verificationType	string		Specifies the type of verification to be done on the entered key.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "keyLen": 0,
    "autoEnd": false,
    "activeFDKs": {
      "fdk01": false,
      "fdk02": false,
      "fdk03": false,
      "fdk04": false,
      "fdk05": false,
      "fdk06": false,
      "fdk07": false,
      "fdk08": false,
      "fdk09": false,
      "fdk10": false,
      "fdk11": false,
      "fdk12": false,
      "fdk13": false,
      "fdk14": false,
      "fdk15": false,
      "fdk16": false,
      "fdk17": false,
      "fdk18": false,
      "fdk19": false,
      "fdk20": false,
      "fdk21": false,
      "fdk22": false,
      "fdk23": false,
      "fdk24": false,
      "fdk25": false,
      "fdk26": false,
      "fdk27": false,
      "fdk28": false,
      "fdk29": false,
      "fdk30": false,
      "fdk31": false,
      "fdk32": false
    },
    "activeKeys": {
      "fk0": false,
      "fk1": false,
      "fk2": false,
      "fk3": false,
      "fk4": false,
```

```
    "fk5": false,
    "fk6": false,
    "fk7": false,
    "fk8": false,
    "fk9": false,
    "fkA": false,
    "fkB": false,
    "fkC": false,
    "fkD": false,
    "fkE": false,
    "fkF": false,
    "fkEnter": false,
    "fkCancel": false,
    "fkClear": false,
    "fkBackspace": false,
    "fkHelp": false,
    "fkDecPoint": false,
    "fk00": false,
    "fk000": false,
    "fkShift": false,
    "fkRES01": false,
    "fkRES02": false,
    "fkRES03": false,
    "fkRES04": false,
    "fkRES05": false,
    "fkRES06": false,
    "fkRES07": false,
    "fkRES08": false,
    "fkOEM01": false,
    "fkOEM02": false,
    "fkOEM03": false,
    "fkOEM04": false,
    "fkOEM05": false,
    "fkOEM06": false
  },
  "terminateFDKs": {
    "fdk01": false,
    "fdk02": false,
    "fdk03": false,
    "fdk04": false,
    "fdk05": false,
    "fdk06": false,
    "fdk07": false,
    "fdk08": false,
    "fdk09": false,
    "fdk10": false,
    "fdk11": false,
    "fdk12": false,
    "fdk13": false,
    "fdk14": false,
    "fdk15": false,
    "fdk16": false,
    "fdk17": false,
    "fdk18": false,
    "fdk19": false,
    "fdk20": false,
    "fdk21": false,
    "fdk22": false,
    "fdk23": false,
    "fdk24": false,
    "fdk25": false,
    "fdk26": false,
    "fdk27": false,
    "fdk28": false,
    "fdk29": false,
    "fdk30": false,
    "fdk31": false,
    "fdk32": false
  },
  "trerminateKeys": {
    "fk0": false,
    "fk1": false,
    "fk2": false,
    "fk3": false,
    "fk4": false,
    "fk5": false,
    "fk6": false,
    "fk7": false,
    "fk8": false,
    "fk9": false,
    "fkA": false,
    "fkB": false,
    "fkC": false,
    "fkD": false,
    "fkE": false,
    "fkF": false,
    "fkEnter": false,
    "fkCancel": false,
    "fkClear": false,
```

```
    "fkBackspace": false,
    "fkHelp": false,
    "fkDecPoint": false,
    "fk00": false,
    "fk000": false,
    "fkShift": false,
    "fkRES01": false,
    "fkRES02": false,
    "fkRES03": false,
    "fkRES04": false,
    "fkRES05": false,
    "fkRES06": false,
    "fkRES07": false,
    "fkRES08": false,
    "fkOEM01": false,
    "fkOEM02": false,
    "fkOEM03": false,
    "fkOEM04": false,
    "fkOEM05": false,
    "fkOEM06": false
   },
   "verificationType": "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
digits	integer		Specifies the number of key digits entered. Applications must ensure all required digits have been entered before trying to store the key.
completion	string		Specifies the reason for completion of the entry.
kcv	string		Contains the key check value data that can be used for verification of the entered key formatted in base 64. This parameter is an empty if device does not have this capability, or the key entry was not fully entered, e.g. the entry was terminated by Enter before the required number of digits was entered.

Example Message (generated)

```
{
 "headers": {
   "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
   "type": "command",
   "name": "string"
 },
 "payload": {
   "status": "ok",
   "errorDescription": "string",
   "digits": 0,
   "completion": "auto",
   "kcv": "string"
 }
}
```

Event Messages

- Pinpad.KeyEvent
- Pinpad.EnterDataEvent

Pinpad.GenerateKCV

Description

This command returns the Key Check Value (kcv) for the specified 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
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 key that should be used to generate the kcv.
keyCheckMode	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	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

Pinpad.MaintainPin

Description

This command is used to control if the PIN is maintained after a PIN processing command for subsequent use by other PIN processing commands. This command is also used to clear the PIN buffer when the PIN is no longer required.

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.
maintainPIN	boolean	false	Specifies if the PIN should be maintained after a PIN processing command. Once set, this setting applies until changed through another call to this command

Example Message (generated)

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

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

Pinpad.KeypressBeep

Description

This command is used to enable or disable the PIN device from emitting a beep tone on subsequent key presses of active or in-active keys. This command is valid only on devices which have the capability to support application control of automatic beeping. See Capabilities structure for 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.
mode	object		Specifies whether automatic generation of key press beep tones should be activated for any active or in-active key subsequently pressed on the PIN. mode selectively turns beeping on and off for active, in-active or both types of keys.
mode.active	boolean	false	Specifies that beeping should be enabled for active keys. If this flag is not present then beeping is disabled for active keys.
mode.inactive	boolean	false	Specifies that beeping should be enabled for in-active keys. If this flag is not present then beeping is disabled for in-active keys.

Example Message (generated)


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


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

Pinpad.SetPinblockData

Description

This function should be used for devices which need to know the data for the PIN block before the PIN is entered by the user. GetPin and GetPinblock should be called after this command. For all other devices Unsupported will be returned here. If this command is required and it is not called, the GetPin command will fail with the generic error SequenceError. If the input parameters passed to this command and GetPinblock are not identical, the GetPinblock command will fail with the generic error InvalidData. The data associated with this command will be cleared on a GetPinblock 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.
customerData	string		The customer data should be an ASCII string. Used for ANSI, ISO-0 and ISO-1 algorithm to build the formatted PIN. For ANSI and ISO-0 the PAN (Primary Account Number, without the check number) is supplied, for ISO-1 a ten digit transaction field is required. If not used a NULL is required. Used for DIEBOLD with coordination number, as a two digit coordination number. Used for EMV with challenge number (8 bytes) coming from the chip card. This number is passed as unpacked string, for example: 0123456789ABCDEF = 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x41 0x42 0x43 0x44 0x45 0x46 For AP PIN blocks, the data must be a concatenation of the PAN (18 digits including the check digit), and the CCS (8 digits).
xorData	string		If the formatted PIN is encrypted twice to build the resulting PIN block, this data can be used to modify the result of the first encryption by an XOR-operation. This parameter is a string of hexadecimal data that must be converted by the application, e.g. 0x0123456789ABCDEF must be converted to 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x41 0x42 0x43 0x44 0x45 0x46 and terminated with 0x00. In other words the application would set xorData to “0123456789ABCDEF”. The hex digits 0xA to 0xF can be represented by characters in the ranges ‘a’ to ‘f’ or ‘A’ to ‘F’. If this value is NULL no XOR-operation will be performed. If the formatted PIN is not encrypted twice (i.e. if lpsKeyEncKey is NULL) this parameter is ignored.
padding	integer		Specifies the padding character. The valid range is 0x00 to 0x0F. Only the least significant nibble is used. This field is ignored for PIN block formats with fixed, sequential or random padding.
format	string		Specifies the format of the PIN block. Possible values are: (see command Capabilities)
key	string		Specifies the key used to encrypt the formatted PIN for the first time, NULL if no encryption is required. If this specifies a double-length or triple-length key, triple DES encryption will be performed. The key referenced by lpsKey must have the WFS_PIN_USEFUNCTION or UserPinRemote attribute. If this specifies an RSA key, RSA encryption will be performed
secondEncKey	string		Specifies the key used to format the once encrypted formatted PIN, NULL if no second encryption required. The key referenced by lpsKeyEncKey must have the UseFunction or UsePinRemote attribute. If this specifies a double-length or triple-length key, triple DES encryption will be performed.
pinBlockAttributes			This parameter specifies the encryption algorithm, cryptographic method, and mode to be used for this command. For a list of valid values see the pinBlockAttributes capabilities field. For a list of valid values see the cryptAttributes capability field. The values specified must be compatible with the key identified by key.
pinBlockAttributes.keyUsage	string		Specifies the key usages supported by the PINBLOCK command.
pinBlockAttributes.algorithm	string		Specifies the encryption algorithms supported by the PINBLOCK command as one of the following values

Name	Type	Default	Description
pinBlockAttributes.modeOfUse	string		Specifies the encryption modes supported by the PINBLOCK command as one of the following values
pinBlockAttributes.cryptoMethod	string		This parameter specifies the cryptographic method that will be used with the encryption algorithm specified by Algorithm.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "customerData": "string",
    "xorData": "string",
    "padding": 0,
    "format": 3624,
    "key": "string",
    "secondEncKey": "string",
    "pinBlockAttributes": {
      "keyUsage": "p0",
      "algorithm": "a",
      "modeOfUse": "e",
      "cryptoMethod": "ecb"
    }
  }
}
```


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

Pinpad.SetLogicalHSM

Description

This command allows an application select the logical HSM that should be active. If the device does not support multiple logical hsm this command returns Unsupported. The QuetyLogicalHSMDetail command can be called to determine the current active logical HSM. Once the active logical HSM is set with this command, that logical hsm remains active until this command is used to change the logical hsm or the system is re-started. The selected HSM is not persistent across re-boots, when applications want to address a specific logical HSM they must ensure that the correct logical hsm is set as the active logical hsm. The commands affected by this command are as follows:

- HSMData
- KeyDetail
- HSMSetData

- SecureMgsSend (only affected for the protocols hsmldi and isops)
- SecureMsgReceive (only affected for the protocols hsmldi and isops)
- HSM_Init
- GetJournal (only affected for the protocol isops). If there are multiple applications that manipulate the current logical hsm then applications must co-operate or use the XFS locking facilities to synchronize access to the logical hsm. The current logical hsm is the same for all clients.

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.
hsmSerialNumber	integer		Specifies the serial number of the hsm that should be set as the active hsm. The value passed in this field corresponds to the hsmSerialNumber field reported in the QueryLogicalHSMDetail command output (and hence corresponds to the CB tag in the hsm tData). The hsmSerialNumber value is encoded as a standard binary value

Example Message (generated)


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

Pinpad.ImportKeyBlock

Description

The command imports an encryption key that has been passed by the application within an ANSI X9 TR-31 key block

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 key being loaded
encKey	string		encKey specifies a key name which will be used to verify and decrypt the key block passed in keyBlock. This key must have a key usage defined as ansTR31Master
keyBlock	string		Specifies the complete key block for the key being imported formatted in base64.

Example Message (generated)


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

Pinpad.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	string		Specifies the method to use to load the certificate
signer	string		Specifies the signer of the certificate to be loaded
certificateData	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	Type	Default	Description
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	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 must be 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

Pinpad.ImportRSAEncipheredPKCS7Key

Description

This command is used to load a Key Transport Key that is either a single-length, double-length or triple-length DES key into the Pinpad. The Key Transport Key should be destroyed if the entire process is not completed. In addition, a new Key Transport Key should be generated each time this protocol is executed. This method ends 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.
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.
importRsaKeyIn	string		Pointer to a binary encoded pkcs #7 represented in DER encoded ASN.1 notation. This allows the Host to verify that key was imported correctly and to the correct Pinpad. The message has an outer Signed-data content type with the signerInfo encryptedDigest field containing the HOSTï¿½s signature. The inner content is an Enveloped-data content type. The ATM identifier is included as the issuerAndSerialNumber within the RecipientInfo.
key	string		Specifies the name of the key to be stored.
use	object		Specifies the type of access for which the key can be used
use.crypt	boolean	false	Key isused for encryption and decryption.
use.function	boolean	false	Key is used for Pin block creation.

Name	Type	Default	Description
use.macing	boolean	false	Key is used for macing.
use.keyEnckey	boolean	false	Key is used as key encryption key
use.noDuplicate	boolean	false	Key can be imported only once
use.svEncKey	boolean	false	Key is used as cbc start Value encryption key.
use.ansTR31master	boolean	false	Key can be used for importing keys packaged within an ANS TR-31 key block. This key usage can only be combined with Construct and SecureConstruct
crklLoadOption	string		Specifies the method to use to load the Key Transport Key

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "importRsaKeyIn": "string",
    "key": "string",
    "use": {
      "crypt": false,
      "function": false,
      "macing": false,
      "keyEnckey": false,
      "noDuplicate": false,
      "svEncKey": false,
      "ansTR31master": false
    },
    "crklLoadOption": "noRandom"
  }
}
```


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
keyLength	string		Specifies the length of the key loaded.
rsaKeyCheckMode	string		Defines algorithm/method used to generate the public key check value/t
rsaData	string		If crklLoadOption is random or noRandom, this data is a pointer to a binary encoded pkcs #7, represented in DER encoded ASN.1 notation. The message has an outer Signed-data content type with the signerInfo encryptedDigest field containing the ATM's signature. The random numbers are included as authenticatedAttributes within the signerInfo. The inner content is a data content type, which contains the HOST identifier as an issuerAndSerialNumber sequence. If rsaKeyCheckMode is none, then this field must be an empty.
keyCheckMode	string		Specifies the mode that is used to create the key check value
keyCheckValue	string		Contains the key verification code data that can be used for verification of the loaded key, NULL if device does not have that capability.

Example Message (generated)

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

Event Messages

Pinpad.DefineLayout

Description

This command allows an application to configure a layout for any PIN device. One or more layouts can be defined with a single request of this command. There can be a layout for each of the different types of keyboard entry modes, if the vendor and the hardware supports these different methods. The types of keyboard entry modes are (1) Mouse mode, (2) Data mode which corresponds to the GetData command, (3) PIN mode which corresponds to the GetPin command, and (4) Secure mode which corresponds to the SecureKeyEntry command. One or more layouts can be preloaded into the device, if the device supports this, or a single layout can be loaded into the device immediately prior to the keyboard command being requested. If a GetData, GetPin, or SecureKeyEntry command is already in progress (or queued), then this command is rejected with a command result of SequenceError. Layouts defined with this command are persistent.

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
entryMode	object		Specifies entry mode to be returned. It can be one of the following flags, or zero to return all supported entry modes
entryMode.data	boolean	false	Specifies that the layout be applied to the GetData entry method.
entryMode.pin	boolean	false	Specifies that the layout be applied to the GetPin entry method.
entryMode.secure	boolean	false	Specifies that the layout be applied to the SecurekeyEntry entry method.
frames	array		There can be one or more frame structures included
frames.xPos	integer		For ETS, specifies the left coordinate of the frame as an offset from the left edge of the screen. For all other device types, this value is ignored
frames.yPos	integer		For ETS, specifies the top coordinate of the frame as an offset from the top edge of the screen. For all other device types, this value is ignored
frames.xSize	integer		For ETS, specifies the width of the frame. For all other device types, this value is ignored
frames.ySize	integer		For ETS, specifies the height of the frame. For all other device types, this value is ignored
frames.floatAction	object		Specifies if the device can float the touch keyboards
frames.floatAction.floatX	boolean	false	Specifies that the PIN device will randomly shift the layout in a horizontal direction
frames.floatAction.floatY	boolean	false	Specifies that the PIN device will randomly shift the layout in a vertical direction
frames.fks	array		Defining details of the keys in the keyboard.

Name	Type	Default	Description
frames.fks.xPos	integer		Specifies the position of the top left corner of the FK relative to the left hand side of the layout. For ETS devices, must be in the range defined in the frame. For non-ETS devices, must be a value between 0 and 999, where 0 is the left edge and 999 is the right edge.
frames.fks.yPos	integer		Specifies the position of the top left corner of the FK relative to the left hand side of the layout. For ETS devices, must be in the range defined in the frame. For non-ETS devices, must be a value between 0 and 999, where 0 is the top edge and 999 is the bottom edge
frames.fks.xSize	integer		Specifies the FK width. For ETS, width is measured in pixels. For non-ETS devices, width is expressed as a value between 1 and 1000, where 1 is the smallest possible size and 1000 is the full width of the layout.
frames.fks.ySize	integer		Specifies the FK height. For ETS, height is measured in pixels. For non-ETS devices, height is expressed as a value between 1 and 1000, where 1 is the smallest possible size and 1000 is the full height of the layout.
frames.fks.keyType	string		Defines the type of XFS key definition value is represented by fk and shiftFK
frames.fks.fk	integer		Specifies the FK code associated with the physical area in non-shifted mode, WFS_PIN_FK_UNUSED if the key is not used.
frames.fks.shiftFK	integer		Specifies the FK code associated with the physical key in shifted mode, WFS_PIN_FK_UNUSED if the key is not used in shifted mode.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "entryMode": {
      "data": false,
      "pin": false,
      "secure": false
    },
    "frames": [
      {
        "xPos": 0,
        "yPos": 0,
        "xSize": 0,
        "ySize": 0,
        "floatAction": {
          "floatX": false,
          "floatY": false
        },
        "fks": [
          {
            "xPos": 0,
            "yPos": 0,
            "xSize": 0,
            "ySize": 0,
            "keyType": "fk",
            "fk": 0,
            "shiftFK": 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"

Example Message (generated)

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

Event Messages

Pinpad.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 PIN 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	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
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error

Name	Type	Default	Description
internalCmdResult	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 dataToSign is an empty string and the command result is success.
signers	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

Pinpad.Authenticate

Description

This command can be used to add authentication to any existing PIN command. The functionality of the command specified by commandID will be executed within the context of this command, and the XFS application should not call the command specified by commandID. The signed data is unique for each command request and therefore can be used only once per command. The StartAuthenticate command must be called before this command. If this command is called without first calling the StartAuthenticate command, then this command will fail and SequenceError will be returned. The StartAuthenticate command does not need to immediately precede the Authenticate command. It is acceptable for other commands to be executed between these commands, except for any command that will clear from the PIN device the data that is being saved in order to verify the signed data provided in the Authenticate command. If this occurs, then SequenceError will be returned.

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.
signer	object		Specifies the signer of the data.
signer.none	boolean		Authentication is not required
signer.certhost	boolean		The data is signed by the current Host, using the RSA certificate-based scheme.
signer.sighost	boolean		The data is signed by the current Host, using the RSA signature-based scheme.
signer.ca	boolean		The data is signed by the Certificate Authority (CA).
signer.hl	boolean		The data is signed by the Higher Level (HL) Authority.
signer.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.
signer.cbcmac	boolean		A MAC is calculated over the data using lpsKey and the CBC MAC algorithm.
signer.cmac	boolean		A MAC is calculated over the data using lpsKey and the CMAC algorithm.
signer.reserved_1	boolean		Reserved for a vendor-defined signing method.
signer.reserved_2	boolean		Reserved for a vendor-defined signing method.
signer.reserved_3	boolean		Reserved for a vendor-defined signing method.
sigKey	string		If cbcmac or cmac are specified for signer, then sigKey is the name of a key with the macing usage. If sighost is specified for signer, then sigKey specifies the name of a previously loaded asymmetric key (i.e. an rsa Public Key). The default signature issuer public key (installed in a secure environment during manufacture) will be used, if sigKey is either an empty string or contains the name of the default signature issuer.Otherwise, this parameter must be NULL empty string.
signedData	string		This field contains the signed version of the data that was provided by the PIN device during the previous call to the StartAuthenticate command. The signer specified by signer is used to do the signing. Both the signature and the data that was signed must be verified before the operation is performed. If certhost, CA , or HL are specified for signer, then signedData is a pkcs#7 signedData structure which includes the data that was returned by the StartAuthenticate command. The optional CRL field may or may not be included in the pkcs#7 signedData structure. If the tr34 flag is set, then either the CA or HL flag must also be set. Please refer to the X9 TR34-2012 [Ref. 42] for more details. If sighost is specified for signer, then signedData is a pkcs#7 signedData structure which includes the data that was returned by the StartAuthenticate command. If cbcmac or cmac are specified for signer, then sigKey must refer to a key loaded with the macing usage
commandId	string		The command name to which authentication is being applied.
inputData	object		Pointer to the input data structure of the command referred to by commandID. For details on the contents of the structure pointed to by inputData, refer to the command referenced by commandID.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "signer": {
      "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
    },
    "sigKey": "string",
    "signedData": "string",
    "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
status	string		ok if the commmand was successful otherwise error
errorDescription	string		If error, identified that cause of the error
internalCmdResult	string		Result from the command referenced by commandID. If the authentication was verified but the internal command failed, then hInternalCmdResult will return an error but the result of this command will be ok.
commandID	string		The command name to which authentication was applied.
outputData	object		Pointer to the output data structure of the command referred to by commandID. For details on the contents of the structure pointed to by outputData, refer to the command referenced by commandID.

Example Message (generated)

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

Event Messages

Pinpad.GetPinblock

Description

This function takes the account information and a PIN entered by the user to build a formatted PIN. Encrypting this formatted PIN once or twice returns a PIN block which can be written on a magnetic card or sent to a host. The PIN block can be calculated using one of the algorithms specified in the Capabilities command. This command will clear the PIN unless the application has requested that the PIN be maintained through the MaintainPin 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
customerData	string		The customer data should be an ASCII string. Used for ANSI, ISO-0 and ISO-1 algorithm to build the formatted PIN. For ANSI and ISO-0 the PAN (Primary Account Number, without the check number) is supplied, for ISO-1 a ten digit transaction field is required. If not used a NULL is required. Used for DIEBOLD with coordination number, as a two digit coordination number. Used for EMV with challenge number (8 bytes) coming from the chip card. This number is passed as unpacked string, for example: 0123456789ABCDEF = 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x41 0x42 0x43 0x44 0x45 0x46 For AP PIN blocks, the data must be a concatenation of the PAN (18 digits including the check digit), and the CCS (8 digits).
xorData	string		If the formatted PIN is encrypted twice to build the resulting PIN block, this data can be used to modify the result of the first encryption by an XOR-operation. This parameter is a string of hexadecimal data that must be converted by the application, e.g. 0x0123456789ABCDEF must be converted to 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x41 0x42 0x43 0x44 0x45 0x46 and terminated with 0x00. In other words the application would set xorData to “0123456789ABCDEF”. The hex digits 0xA to 0xF can be represented by characters in the ranges ‘a’ to ‘f’ or ‘A’ to ‘F’. If this value is NULL no XOR-operation will be performed. If the formatted PIN is not encrypted twice (i.e. if lpsKeyEncKey is NULL) this parameter is ignored.
padding	integer		Specifies the padding character. The valid range is 0x00 to 0x0F. Only the least significant nibble is used. This field is ignored for PIN block formats with fixed, sequential or random padding.
format	string		Specifies the format of the PIN block. Possible values are: (see command Capabilities)
key	string		Specifies the key used to encrypt the formatted PIN for the first time, NULL if no encryption is required. If this specifies a double-length or triple-length key, triple DES encryption will be performed. The key referenced by lpsKey must have the WFS_PIN_USEFUNCTION or UserPinRemote attribute. If this specifies an RSA key, RSA encryption will be performed
secondEncKey	string		Specifies the key used to format the once encrypted formatted PIN, NULL if no second encryption required. The key referenced by lpsKeyEncKey must have the UseFunction or UsePinRemote attribute. If this specifies a double-length or triple-length key, triple DES encryption will be performed.
pinBlockAttributes			This parameter specifies the encryption algorithm, cryptographic method, and mode to be used for this command. For a list of valid values see the pinBlockAttributes capabilities field. For a list of valid values see the cryptAttributes capability field. The values specified must be compatible with the key identified by key.
pinBlockAttributes.keyUsage	string		Specifies the key usages supported by the PINBLOCK command.
pinBlockAttributes.algorithm	string		Specifies the encryption algorithms supported by the PINBLOCK command as one of the following values
pinBlockAttributes.modeOfUse	string		Specifies the encryption modes supported by the PINBLOCK command as one of the following values
pinBlockAttributes.cryptoMethod	string		This parameter specifies the cryptographic method that will be used with the encryption algorithm specified by Algorithm.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "customerData": "string",
    "xorData": "string",
    "padding": 0,
    "format": 3624,
    "key": "string",
    "secondEncKey": "string",
    "pinBlockAttributes": {
      "keyUsage": "p0",
      "algorithm": "a",
      "modeOfUse": "e",
      "cryptoMethod": "ecb"
    }
  }
}
```

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
pinBlock	string		The encrypted PIN block formatted in base64

Example Message (generated)

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

Event Messages

Pinpad.LuxLoadAppKey

Description

This command can be used to load an Application Key and to replace the Transport Key. Once the keys are loaded the encryptor will use the keys to do the other commands. The encryptor will use the Application Key to obtain a random encrypted session key needed for the PIN Encryption, the MAC Computation and the Data Encryption/Decryption. The application will use the Transport Key for loading the other keys (mkMac, mkPac and mkEnc) into the encryptor. When this command is used for replacing the Transport Key, the new Transport key is provided encrypted by the existing Transport Key. The generation of the first Transport Key is the responsibility of the Authorization Center in Luxemburg (CETREL). The loading method of the first Transport Key into the encryptor is vendor dependent. Keys loaded through this command are reported through the KeyDetail commands. Keys loaded through this command do not require to be deleted before the application can replace them. To access this command, the object Enclo of the Enclo command has to be defined as required by the Luxembourg protocol (see general definition in the first paragraph). The only definitions specific to this command are the input and output parameters.

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.
command	string		Is set to LuxLoadAppKey.
keyName	string		This field contains the name of the key to be loaded. The Service Provider will right pad the keyName to 20 bytes with char 0x20.
sequenceNumber	string		"This field is defined by the Authorization Center in Luxemburg (CETREL) and contains a 4 bytes key logic number as follows: Least significant 2 bytes represent the Key Generation Most significant 2 bytes represent the Key Version The key logic number will contribute in the MAC calculation, in the PIN block construction and in the Data Encryption/Decryption."
keyData	string		The command name to which authentication is being applied.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "command": "string",
    "keyName": "mkMac",
    "sequenceNumber": 2001,
    "keyData": "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
command	string		Is set to LuxLoadAppKey.
result	string		The command reply codes.

Example Message (generated)

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


Event Messages

Pinpad.LuxGenerateMac

Description

This command is used to generate the CBC-MAC (Message Authentication Code ISO9797-1:1999, Padding Method 1, MAC Algorithm 3). This command returns the generated MAC for the data passed in. To access the LuxGenerateMac command, the payload of the Enclo command has to be defined as required by the Luxembourg protocol (see general definition in the first paragraph). The only definitions specific to this command are the input and output parameters

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		Is set to LuxGenerateMac.
data	string		The data parameter contains the data whose MAC is to be generated formatted in base64. data will be padded according to ISO9797-1:1999, Padding Method 1 if it is not passed in as multiple of 8 bytes.
macLength	integer		Specifies the MAC length. Legal values are: 2, 4, 6 or 8.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "command": "string",
    "data": "string",
    "macLength": 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
command	string		Is set to LuxGenerateMac.
result	string		The Command reply codes
mac	string		The mac parameter contains the generated mac formatted in base64.
random	string		The random parameter contains the random value used to work out th session key.

Example Message (generated)


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


Event Messages

Pinpad.LuxCheckMac

Description

This command verifies the CBC-MAC (Message Authentication Code ISO9797-1:1999, Padding Method 1, MAC Algorithm 3). This command generates a MAC for the data passed in and compares it with the provided MAC value. To access the LuxCheckMac command, the payload of the Enclo command has to be defined as required by the Luxembourg protocol (see general definition in the first paragraph). The only definitions specific to this command are the input and output parameters.

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		Is set to LuxCheckMac.
data	string		The data parameter contains the data whose mac is to be checked formatted in base64. Data will be padded according to ISO9797-1:1999, Padding Method 1 if it is not passed in as multiple of 8 bytes.
mac	string		The mac parameter contains the mac that is to be checked formatted in base64. Legal values for the mac length are: 2, 4, 6 or 8.
random	string		The random parameter contains the random value used to work out the session key formatted in based64.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "command": "string",
    "data": "string",
    "mac": "string",
    "random": "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
command	string		Is set to LuxCheckMac.
result	string		The command reply codes (see general definition in the first paragraph)

Example Message (generated)

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

Event Messages

Pinpad.LuxBuildPinBlock

Description

This command is used to construct the PIN blocks described below for remote PIN check. For PIN block format see comment section below. To access the LuxBuildPinBlock command, the payload of the Enclo command has to be defined as required by the Luxembourg protocol (see general definition in the first paragraph). The only definitions specific to this command are the input and output parameters.

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		Is set to LuxBuildPinBlock.
format	string		Specifies the format of the PIN block.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "command": "string",
    "format": "luxFormIso1"
  }
}
```


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
command	string		Is set to LuxBuildPinBlock.
result	string		The Command reply codes
pinBlock	string		The pinBlock parameter contains the constructed PIN block formatted in base64.
random	string		The random parameter contains the random value used to work out th session key formatted in base64.

Example Message (generated)


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


Event Messages

Pinpad.LuxDecryptTDES

Description

This command is used to decrypt the data according to triple DES algorithm. To access the LuxDecryptTDES command, the payload of the Enclo command has to be defined as required by the Luxembourg protocol (see general definition in the first paragraph). The only definitions specific to this command are the input and output parameters.

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		Is set to LuxDecryptTDES.
type	string		An integer word specifying the type of triple DES decryption
data	string		The data parameter contains the data to be decrypted. data must be multiple of 8-byte blocks formatted in base64.
iv	string		If type is luxTriDesCbc then this field contains the 8 bytes of data containing the Initial Value needed for decryption in CBC mode formatted in base64. Otherwise this field is ignored.
random	string		The random parameter contains the random value used to calculate the session key formatted in base64.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "command": "string",
    "type": "luxTriDesEcb",
    "data": "string",
    "iv": "string",
    "random": "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
command	string		Is set to LuxDecryptTDES.
result	string		The Command reply codes
data	string		The data parameter contains the decrypted data formatted in base64.

Example Message (generated)

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

Event Messages

Pinpad.LuxEncryptTDES

Description

This command is used to encrypt the data according to triple DES algorithm. To access the LuxEncryptTDES command, the payload of the Enclo command has to be defined as required by the Luxembourg protocol (see general definition in the first paragraph). The only definitions specific to this command are the input and output parameters.

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		Is set to LuxEncryptTDES.
type	string		An integer word specifying the type of triple DES encryption
data	string		The data parameter contains the data to be decrypted formatted in base64. data must be multiple of 8-byte blocks.Application must fill the end of the data with 0x00 if the data does not contain a multiple of 8-byte blocks.
iv	string		If type is luxTriDesCbc then this field contains the 8 bytes of data containing the Initial Value needed for decryption in CBC mode formatted in base64. Otherwise this field is ignored.

Example Message (generated)

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

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
command	string		Is set to LuxEncryptTDES.
result	string		The Command reply codes
data	string		The data parameter contains the decrypted data formatted in base64.
random	string		The random parameter contains the random value used to calculate the session key formatted in base64.

Example Message (generated)


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


Event Messages

Pinpad.CHNDigest

Description

This command is used to compute a hash code on a stream of data using the specified SM3 hash algorithm. This command can be used to verify PBOC static and dynamic data.

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		Is set to CHNDigest.
hashAlgorithm	string		Specifies which hash algorithm should be used to calculate the hash.
digestInput	string		Contains the data to be hashed.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "command": "string",
    "hashAlgorithm": "sm3Digest",
    "digestInput": "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
command	string		Is set to CHNDigest.
result	string		The Command reply codes
digestOutput	string		Contains the data containing the calculated hash.

Example Message (generated)

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

Event Messages

Pinpad.CHNSetSm2Param

Description

This command is used to set SM2 algorithm parameter. The SM2 algorithm is based on elliptic curves. Six parameters need to be set before using to calculate. There are defined in Password industry standard of the People's Republic of China GM/T 0003.5-2012 [Ref. 43].

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	Type	Default	Description
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		Is set to CHNSetSm2Param.
p	string		Prime number p. It should be greater than 3. It is used to define prime number field Fp It is defined in Password industry standard of the People's Republic of China GM/T 0003.5-2012 [Ref. 43]. string formatted in base64
a	string		An element a in prime number field Fp. They are used to define elliptic curveâ€™s equation: y2 = x3 + a*x + b. It is defined in Password industry standard of the People's Republic of China GM/T 0003.5-2012 [Ref. 43]. string formatted in base64
b	string		An element b in prime number field Fp. They are used to define elliptic curveâ€™s equation: y2 = x3 + a*x + b. It is defined in Password industry standard of the People's Republic of China GM/T 0003.5-2012 [Ref. 43]. string formatted in base64
n	string		The number of base points on the elliptic curve. It should be greater than 2191, and greater than 4*p1/2. It is defined in Password industry standard of the People's Republic of China GM/T 0003.5-2012 [Ref. 43]. string formatted in base64
xG	string		The X coordinate of one base point G= (xG, yG) on the elliptic curve. The base point G should be in the set of prime number field Fp. It is defined in Password industry standard of the People's Republic of China GM/T 0003.5-2012 [Ref. 43]. string formatted in base64
yG	string		The Y coordinate of one base point G= (xG, yG) on the elliptic curve. The base point G should be in the set of prime number field Fp. It is defined in Password industry standard of the People's Republic of China GM/T 0003.5-2012 [Ref. 43]. string formatted in base64

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "command": "string",
    "p": "string",
    "a": "string",
    "b": "string",
    "n": "string",
    "xG": "string",
    "yG": "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
command	string		Is set to CHNSetSm2Param.
result	string		The command reply codes

Example Message (generated)

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

Event Messages

Pinpad.CHNImportSM2PublicKey

Description

This command is used to set sm2 algorithm parameter. The sm2 algorithm is based on elliptic curves. Six parameters need to be set before using to calculate. There are defined in Password industry standard of the People's Republic of China GM/T 0003.5-2012 [Ref. 43].

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		Is set to CHNImportSM2PublicKey.
key	string		Specifies the name of key being loaded
value	string		Contains the GM/T 2012 SM2 Public Key to be loaded.
use	string		Specifies the type of access for which the key can be used. If this parameter equals delete, the key is deleted. If use equals zero the specified key is deleted.When no signature is required to authenticate the deletion of a public key, all parameters but Key are ignored. In addition, CHNImportSm2PublicKey and CHNImportSm2SignedSm4Key can be used to delete a key that has been imported with this command. When a signature is required to authenticate the deletion of the public key, all parameters in the command are used. value must contain the concatenation of the Security Item which uniquely identifies the PIN device (see the command CHNExportSm2IssuerSignedItem) and the GM/T 2012 SM2 public key to be deleted. signature contains the signature generated from value using the private key component of the public key being deleted. The equivalent commands in the certificate scheme must not be used to delete a key imported through the signature scheme.
sigKey	string		SigKey specifies the name of a previously loaded asymmetric key (i.e. a SM2 Public Key) which will be used to verify the signature passed in signature. The default signature Issuer public key (installed in a secure environment during manufacture) will be used, if sigKey is either an empty string or contains the name of the default signature issuer.
sm2SignatureAlgorithm	string		Defines the algorithm used to generate the signature specified in signature.
signature	string		Contains the signature associated with the key being imported or deleted. The signature is used to validate the key request has been received from a trusted sender. This value can be an empty string when no key validation is required.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "command": "string",
    "key": "string",
    "value": "string",
    "use": "usesm2public",
    "sigKey": "string",
    "sm2SignatureAlgorithm": "pinSignNa",
    "signature": "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
command	string		Is set to CHNImportSM2PublicKey.
result	string		The Command reply codes
sm2KeyCheckMode	string		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.
keyCheckValue	string		Contains the public key check value as defined by the sm2KeyCheckMode flag.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "status": "ok",
    "errorDescription": "string",
    "command": "string",
    "result": "success",
    "sm2KeyCheckMode": "none",
    "keyCheckValue": "string"
  }
}
```


Event Messages

Pinpad.CHNSign

Description

This command is used to sign sm2 algorithm data.

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.
command	string		Is set to CHNSign.
key	string		Specifies the name of the stored key.
signerId	string		Specifies the signer's ID.
plaintTextData	string		Pointer to the data that need to be signed.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "command": "string",
    "key": "string",
    "signerId": "string",
    "plaintTextData": "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
command	string		Is set to CHNSign.
result	string		The Command reply codes
signData	string		signature.

Example Message (generated)

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

Event Messages

Pinpad.CHNVerify

Description

This command is used to verify sm2 algorithm signature data.

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		Is set to CHNVerify.
key	string		Specifies the name of the stored key.
cipherData	string		User's Plain text Data formatted in base64.
signData	string		Signature data signed by CHNSign formatted in base64.

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "command": "string",
    "key": "string",
    "cipherData": "string",
    "signData": "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	Type	Default	Description
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
command	string		Is set to CHNVerify.
result	string		The Command reply codes
result.signatureError	string		Signature data is wrong

Example Message (generated)

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

Event Messages

Pinpad.CHNExportSm2IssuerSignedItem

Description

This command is used to export data elements from the PIN 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: â€¢ The Security Item which uniquely identifies the PIN device. This value may be used to uniquely identify a PIN device and therefore confer trust upon any key or data obtained from this device. â€¢ The SM2 Public key component of a public/private key pair that exists within the PIN 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
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		Is set to CHNExportSm2IssuerSignedItem.
exportItemType	string		Defines the type of data item to be exported from the PIN.
name	string		Specifies the name of the public key to be exported. The private/public key pair was installed during manufacture. If name is empty, 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",
    "command": "string",
    "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
command	string		Is set to CHNExportSm2IssuerSignedItem.
result	string		The command reply codes
value	string		If a public key was requested then value contains the GM/T 2012 SM2 Public Key. If the security item was requested then Value contains the PINâ€™s Security Item, which may be vendor specific.
sm2SignatureAlgorithm	string		Specifies the algorithm used to generate the signature returned in signature.
signature	string		Specifies the SM2 signature of the data item exported. an empty string 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",
    "command": "string",
    "result": "success",
    "value": "string",
    "sm2SignatureAlgorithm": "na",
    "signature": "string"
  }
}
```


Event Messages

Pinpad.CHNGenerateSm2KeyPair

Description

This command will generate a new SM2 key pair. The public key generated as a result of this command can subsequently be obtained by calling CHNExportSm2EPPSignedItem command. The newly generated key pair can only be used for the use defined in the Use 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.
command	string		Is set to CHNGenerateSm2KeyPair.
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 Sm2PrivateSign 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.

Example Message (generated)

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

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
command	string		Is set to CHNGenerateSm2KeyPair.
result	string		The command reply codes

Example Message (generated)

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

Event Messages

Pinpad.CHNExportSm2EPPSignedItem

Description

This command is used to export data elements from the PIN device that have been signed by a private key within the EPP. This command is used in place of the CHNExportSm2EPPSignedItem command, when a private key generated within the PIN 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: â€¢ The Security Item which uniquely identifies the PIN device. This value may be used to uniquely identify a PIN device and therefore confer trust upon any key or data obtained from this device. â€¢ The SM2 Public key component of a public/private key pair that exists within the PIN device. The public/private key pairs exported by this command are either installed during manufacture or generated through the CHNGenerateSm2KeyPair command. The KeyDetail command can be used to determine the valid uses for the exported public 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
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		Is set to CHNExportSm2EPPSignedItem.
exportItemType	string		Defines the type of data item to be exported from the PIN.
name	string		Specifies the name of the public key to be exported. This can either be the name of a key-pair generated through CHNGenerateSm2KeyPair 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 retured in both the selfSignature and signature fields.

Example Message (generated)

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

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
command	string		Is set to CHNExportSm2EPPSignedItem.
result	string		The Command reply codes
value	string		If a public key was requested then value contains the GM/T 2012 SM2 Public Key. If the security item was requested then lValue contains the PINâ€™s Security Item, which may be vendor specific.
selfSignature	string		If a public key was requested then selfSignature contains the SM2 signature of the public key exported, generated with the key-pairâ€™s private component. NULL can be returned when key selfSignature are not supported/required.
signature	string		Specifies the SM2 signature of the data item exported. NULL can be returned when signature are not supported/required.

Example Message (generated)


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


Event Messages

Pinpad.CHNImportSm2SignedSm4Key

Description

This command is used to load a Symmetric Key that is a SM4 key into the encryptor. The key passed by the application is loaded in the encryption module, the (optional) signature is used during validation, the key is decrypted using the deviceâ€™s sm2 Private Key, and is then stored. The loaded key will be discarded at any stage if any of the above fails. The use parameter restricts the cryptographic functions that the imported key can be used for. If a Signature algorithm is specified that is not supported by the PIN Service Provider, then the message will not be decrypted and the command fails.

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		Is set to CHNImportSm2SignedSm4Key.
key	string		Specifies the name of key being loaded.
decryptKey	string		Specifies the name of the RSA private key used to decrypt the symmetric key. If decryptKey is an empty then the default decryption private key is used.
sm2EncipherAlgorithm	string		Specifies the RSA algorithm that is used, along with the private key, to decipher the imported key.

Name	Type	Default	Description
value	string		Specifies the enciphered value of the key to be loaded. value contains the concatenation of the random number (when present) and enciphered key.
use	object		Specifies the type of access for which the key can be used. If this parameter equals zero, the key is deleted.If use doesn't have set any of possible variables the specified key is deleted. In that case all parameters but key are ignored. CHNImportSM2PublicKey and CHNImportSm2GinedSm4Key can be used to delete a key that has been imported with this command. The equivalent commands in the certificate scheme must not be used to delete a key imported through the signature scheme.
use.crypt	boolean		Key is used for encryption.
use.function	boolean		Key is used for PIN block creation
use.macing	boolean		Key is used for MACing
use.keyEncKey	boolean		Key is used as key encryption key.
use.pinLocal	boolean		Key is used for local PIN check.
sigKey	string		If SigKey is NULL then the key signature will not be used for validation and Signature is ignored. Otherwise SigKey specifies the name of an Asymmetric Key (i.e. an SM2 Public Key) previously loaded which will be used to verify the signature passed in Signature.
sm2SignatureAlgorithm	string		Specifies the algorithm used to generate the Signature specified in Signature.
signature	string		Contains the Signature associated with the key being imported. The Signature is used to validate the key has been received from a trusted sender. The signature is generated over the contents of the Value. The Signature signature contains NULL when no key validation is required.

Example Message (generated)


```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "timeout": "5000",
    "command": "string",
    "key": "string",
    "decryptKey": "string",
    "sm2EncipherAlgorithm": "sm2GmT2012",
    "value": "string",
    "use": {
      "crypt": true,
      "function": true,
      "macing": true,
      "keyEncKey": true,
      "pinLocal": true
    },
    "sigKey": "string",
    "sm2SignatureAlgorithm": "na",
    "signature": "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
command	string		Is set to CHNImportSm2SignedSm4Key.

Name	Type	Default	Description
result	string		The command reply codes
keyCheckMode	string		Specifies the mode that is used to create the key check value.
keyCheckValue	string		The key verification data that can be used for verification of the loaded key, NULL if device does not have that capability.

Example Message (generated)


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


Event Messages

Unsolicited Events

Pinpad.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"
  }
}
```


Pinpad.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
  }
}
```


Pinpad.StatusChangeEvent

Description

This service event specifies that the status 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
encryptionState	string		Specifies the state of the encryption module
autoBeepMode	string		Specifies whether automatic beep tone on key press is active or not. Active and in-active key beeping is reported independently. autoBeepMode can take a combination of the following values, if the flag is not set auto beeping is not activated (or not supported) for that key type (i.e. active or in-active keys)
certificateState	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": {
    "encryptionState": "ready",
    "autoBeepMode": "active",
    "certificateState": "unknown"
  }
}
```


Pinpad.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. an empty string if not required
key	string		The value of the encryption key formatted in base 64. an empty string if not required.

Example Message (generated)

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

Pinpad.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	string		Specifies the name of the key that caused the error.
errorCode	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"
  }
}
```

Pinpad.OPTRequiredEvent

Description

This event indicates that the online date/time stored in a HSM has been reached.

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
hsmSerialNumber	integer		Specifies the serial number of the logical hsm where the online time has been reached. If logical hsms are not supported then optRequired is NULL. The hsmSerialNumber value is encoded as a standard binary value (i.e. it is not BCD).

Example Message (generated)

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

Pinpad.HSMTDataChangedEvent

Description

This event indicates that one of the values of the terminal data has changed (these are the data that can be set using HSMSetTData). I.e. this event will be sent especially when the online time or the hsm status is changed because of a hsmInit command or an OPT online dialog (SecureMsgSend/Receive with PROTISOPS). On configurations with multiple logical HSMs, the serial number tag must be included within the data so that the logical HSM that has changed can be identified.

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
tData	string		Contains the parameter settings as a series of â€œtag/length/valueâ€? items. See command HSMSetTData for the tags supported. Binary data formatted in base 64

Example Message (generated)

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

Pinpad.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	string		Specifies change of the certificate state inside of the Pinpad.

Example Message (generated)

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

Pinpad.HSMChangedEvent

Description

This event indicates that the currently active logical HSM has been changed. This event will be triggered when an application changes the current HSM through the SetLogicalHSM command. This event is not generated if the HSM is not 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
hsmSerialNumber	integer		Specifies the serial number of the logical hsm that has been made active. The hsmSerialNumber value is encoded as a standard binary value (i.e. it is not BCD).

Example Message (generated)

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

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
  }
}
```

Pinpad.KeyEvent

Description

This event specifies that any active key has been pressed at the PIN pad. It is used if the device has no internal display unit and the application has to manage the display of the entered digits. It is the responsibility of the application to identify the mapping between the FDK code and the physical location of the FDK.

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
completion	string		Specifies the reason for completion of the entry.
digit.	object		Specifies the function keys available for this physical device.
digit.fk0	boolean	false	
digit.fk1	boolean	false	
digit.fk2	boolean	false	
digit.fk3	boolean	false	
digit.fk4	boolean	false	
digit.fk5	boolean	false	
digit.fk6	boolean	false	
digit.fk7	boolean	false	
digit.fk8	boolean	false	
digit.fk9	boolean	false	
digit.fkA	boolean	false	
digit.fkB	boolean	false	
digit.fkC	boolean	false	
digit.fkD	boolean	false	
digit.fkE	boolean	false	

Name	Type	Default	Description
digit.fkF	boolean	false	
digit.fkEnter	boolean	false	
digit.fkCancel	boolean	false	
digit.fkClear	boolean	false	
digit.fkBackspace	boolean	false	
digit.fkHelp	boolean	false	
digit.fkDecPoint	boolean	false	
digit.fk00	boolean	false	
digit.fk000	boolean	false	
digit.fkShift	boolean	false	
digit.fkRES01	boolean	false	
digit.fkRES02	boolean	false	
digit.fkRES03	boolean	false	
digit.fkRES04	boolean	false	
digit.fkRES05	boolean	false	
digit.fkRES06	boolean	false	
digit.fkRES07	boolean	false	
digit.fkRES08	boolean	false	
digit.fkOEM01	boolean	false	
digit.fkOEM02	boolean	false	
digit.fkOEM03	boolean	false	
digit.fkOEM04	boolean	false	
digit.fkOEM05	boolean	false	
digit.fkOEM06	boolean	false	
digit.	object		Specifies the FDK keys available for this physical device.
digit.fdk01	boolean	false	
digit.fdk02	boolean	false	
digit.fdk03	boolean	false	
digit.fdk04	boolean	false	
digit.fdk05	boolean	false	
digit.fdk06	boolean	false	
digit.fdk07	boolean	false	
digit.fdk08	boolean	false	
digit.fdk09	boolean	false	
digit.fdk10	boolean	false	
digit.fdk11	boolean	false	
digit.fdk12	boolean	false	
digit.fdk13	boolean	false	
digit.fdk14	boolean	false	
digit.fdk15	boolean	false	
digit.fdk16	boolean	false	
digit.fdk17	boolean	false	
digit.fdk18	boolean	false	
digit.fdk19	boolean	false	
digit.fdk20	boolean	false	
digit.fdk21	boolean	false	
digit.fdk22	boolean	false	
digit.fdk23	boolean	false	
digit.fdk24	boolean	false	
digit.fdk25	boolean	false	
digit.fdk26	boolean	false	
digit.fdk27	boolean	false	
digit.fdk28	boolean	false	
digit.fdk29	boolean	false	
digit.fdk30	boolean	false	
digit.fdk31	boolean	false	
digit.fdk32	boolean	false	

Example Message (generated)

```
{
  "headers": {
    "requestId": "b34800d0-9dd2-4d50-89ea-92d1b13df54b",
    "type": "command",
    "name": "string"
  },
  "payload": {
    "completion": "auto",
    "digit": {
      "fk0": false,
      "fk1": false,
      "fk2": false,
      "fk3": false,
      "fk4": false,
      "fk5": false,
      "fk6": false,
      "fk7": false,
      "fk8": false,
      "fk9": false,
      "fkA": false,
      "fkB": false,
      "fkC": false,
      "fkD": false,
      "fkE": false,
      "fkF": false,
      "fkEnter": false,
      "fkCancel": false,
      "fkClear": false,
      "fkBackspace": false,
      "fkHelp": false,
      "fkDecPoint": false,
      "fk00": false,
      "fk000": false,
      "fkShift": false,
      "fkRES01": false,
      "fkRES02": false,
      "fkRES03": false,
      "fkRES04": false,
      "fkRES05": false,
      "fkRES06": false,
      "fkRES07": false,
      "fkRES08": false,
      "fkOEM01": false,
      "fkOEM02": false,
      "fkOEM03": false,
      "fkOEM04": false,
      "fkOEM05": false,
      "fkOEM06": false,
      "fdk01": false,
      "fdk02": false,
      "fdk03": false,
      "fdk04": false,
      "fdk05": false,
      "fdk06": false,
      "fdk07": false,
      "fdk08": false,
      "fdk09": false,
      "fdk10": false,
      "fdk11": false,
      "fdk12": false,
      "fdk13": false,
      "fdk14": false,
      "fdk15": false,
      "fdk16": false,
      "fdk17": false,
      "fdk18": false,
      "fdk19": false,
      "fdk20": false,
      "fdk21": false,
      "fdk22": false,
      "fdk23": false,
      "fdk24": false,
      "fdk25": false,
      "fdk26": false,
      "fdk27": false,
      "fdk28": false,
      "fdk29": false,
      "fdk30": false,
      "fdk31": false,
      "fdk32": false
    }
  }
}
```


Pinpad.EnterDataEvent

Description

This mandatory event notifies the application when the device is ready for the user to start entering data.

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"

Example Message (generated)

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